European IT Certification Curriculum Self-Learning Preparatory Materials

EITC/WD/HCF

HTML and CSS Fundamentals

This document constitutes European IT Certification curriculum self-learning preparatory material for the EITC/WD/HCF HTML and CSS Fundamentals programme.

This self-learning preparatory material covers requirements of the corresponding EITC certification programme examination. It is intended to facilitate certification programme's participant learning and preparation towards the EITC/WD/HCF HTML and CSS Fundamentals programme examination. The knowledge contained within the material is sufficient to pass the corresponding EITC certification examination in regard to relevant curriculum parts. The document specifies the knowledge and skills that participants of the EITC/WD/HCF HTML and CSS Fundamentals certification programme should have in order to attain the corresponding EITC certificate.

TABLE OF CONTENTS

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: INTRODUCTION
TOPIC: HOW TO GET STARTED WITH HTML AND CSS

HTML and CSS are fundamental languages used in web development to create complete websites. HTML, which stands for hypertext markup language, is used to structure the content of a website. It allows us to mark up specific sections of a website, such as the main content or the menu, and inform the browser about their purpose.

On the other hand, CSS, which stands for cascading style sheet, is responsible for styling the content inside the website. While HTML focuses on the structure, CSS is used to make the website visually appealing by determining how the content should look. For example, CSS is used to define the size, color, and position of elements like menus.

To illustrate how HTML looks like, here is a basic example:

```html
<!DOCTYPE html>
<html>
<head>
    <title>My First Website</title>
</head>
<body>
    <h1>Welcome to My Website</h1>
    <p>This is some sample text.</p>
</body>
</html>
```

Creating websites requires a computer and a text editor. Two popular text editors for web development are Sublime Text and Notepad++. Sublime Text is a free text editor, while Notepad++ is also free and recommended for beginners. These text editors provide features like syntax highlighting and error indicators to help with coding.

When creating a website, you can work offline by storing all the website files, including documents and images, in a folder on your computer. Later, you can upload this folder to an online server to make the website accessible on the internet.

HTML and CSS are essential languages for web development. HTML is used to structure the content of a website, while CSS is used to style the content and make it visually appealing. With a computer and a text editor like Sublime Text or Notepad++, you can create websites by writing HTML and CSS code.

HTML and CSS are fundamental technologies used in web development. In this didactic material, we will discuss how to get started with HTML and CSS.

To begin, you can use any text editor to create HTML files. Sublime Text is one example, but you can also use editors like Notepad++ or any other text editor of your choice. The text editor itself does not create specific files or documents that can only be used within the program. Once you create an HTML file, you can open it in any text editor.

Next, let's address a common question: how long does it take to learn HTML? The answer depends on your dedication and learning pace. If you are willing to spend extended periods learning and practicing, you could potentially create a website within a day. However, for more complex websites, it may require additional time and learning beyond a single day.

Now, let's explore what you can do with HTML and CSS. As an example, let's consider the YouTube website. On this page, you will find various elements such as a logo, navigation bar, search bar, profile tools, videos, and more. Using HTML and CSS, you can create the visible parts of a website, including these elements. You can design the appearance of the elements, insert images, and create links between pages.

However, it's important to note that HTML and CSS alone cannot create all the functionalities of a website. For example, you cannot create features like a subscription system, login functionality, or interactive search bars without using additional programming languages. Even the videos on the YouTube website are loaded from a database, which requires another language to interact with the database.

HTML and CSS allow you to create the visual aspects of a website, but not the functionalities. You can program everything you see on a website, but you would need to hard code it. This means that if you need to make changes, you would have to modify the code manually each time.

Before we conclude, it's worth mentioning that the color scheme in Sublime Text may differ from what you see on your screen. Each user can customize the color scheme in their text editor. So, if the code you write in your editor doesn't have the same colors as mine, it doesn't indicate any errors. It's simply a result of using a different color scheme.

In the next episode, we will learn how to get started on our first website. We will cover creating a root folder and the first document, which will serve as the front page of our website.

If you're interested in more tutorials, feel free to check out my channel and subscribe for more content. I hope to see you in the next episode!

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: GETTING STARTED
TOPIC: CREATING HTML PROJECT AND DOCUMENT

Today, we will learn how to set up a new project for creating a website and how to create our first page within the website. To avoid any confusion, we will be using a different editor called Adam for these tutorials. Adam is a free editor that is nearly identical to Sublime Text, but without the annoying pop-ups that come with the free version of Sublime Text. It is important that everyone can follow these lessons and use an editor that is accessible to all.

To create a new web project, we need to create a new folder on our computer. In this case, we will create a folder on the desktop. Right-click on the desktop, go to "New," and then select "Folder." You can name this folder anything you like. In this case, we will call it the root folder, as that is the technical term for a website folder. Inside this folder, we will place all the documents, images, videos, or any other media we want to include in our website.

Next, we will open up the text editor, Adam. When we open Adam for the first time, we may see some windows that we don't need. We can close these windows to have only the untitled page inside the editor. If the untitled page is accidentally closed, we can create a new one by going to "File" and selecting "New File."

Inside this file, we will insert the code that will become the front page of our website. Before we proceed, it is important to save the file. Press Ctrl + S or go to "File" and select "Save As." Save the file inside the root folder on the desktop. Name the file "index.html." It is crucial to name it "index.html" as this is how the server will recognize it as the front page of the website. Naming it something else, like "page.html" or "home.html," will not work.

After saving the file, we will see a project window on the left side of the editor, and the file will still be open on the right side. The project window displays the documents within the root folder, allowing us to open them directly from there. To close the project window, click the arrow next to it. To open it again, click the arrow once more.

Now, let's focus on the index file. The first thing we need to do is inform the browser that this is an HTML document. Although the file name contains "HTML," we still need to include HTML tags within the document for the browser to recognize it as an HTML file. For example, an HTML tag consists of an opening tag and a closing tag. The name inside the tags determines the type of content we will write within.

HTML and CSS Fundamentals - Getting started - Creating HTML project and document

HTML (Hypertext Markup Language) is the standard language used to create web pages. It allows us to structure the content of our web pages using tags. Tags are enclosed in angle brackets (<>) and are used to define different elements and their attributes.

To create an HTML project and document, we need to start by informing the browser that we are working with an HTML file. We do this by using the <!DOCTYPE html> declaration at the very beginning of the document. This declaration tells the browser that we are using HTML5, the latest version of HTML.

Next, we need to define the start and end of the HTML document. This is done by enclosing all the content of the page between the <html> and </html> tags. Anything we put inside these tags will be considered part of the HTML file.

Inside the HTML document, there are two important

To create the <head> and <body> tags quickly in text editors like Atom or Sublime Text, we can use shortcuts like typing "head" and then pressing the tab key. This will generate the opening and closing tags automatically.

As a convention, when creating our first application, it is common to write "Hello world" as a way to test our code. Inside the <body> tag, we can simply write "Hello world" to display this message on the webpage.

Inside the <head> tag, we need to include some essential information for the website to function properly. One important element is the meta tag. The meta tag provides specific information about the website, such as the character set used. We can include the meta tag by typing "<meta charset="UTF-8">". This tells the browser to use the UTF-8 character set, which covers all the characters typically used in websites.

Another important tag to include inside the <head> tag is the <title> tag. The <title> tag is used to define the title of the webpage, which is displayed in the browser's title bar or tab. The <title> tag has an opening and closing tag, unlike the meta tag. For example, we can include "<title>My Webpage</title>" to set the title of our webpage as "My Webpage".

By following these steps, we can create a basic HTML project and document. Remember to include all the necessary tags, such as the <!DOCTYPE html> declaration, <html> and </html> tags to define the start and end of the HTML document, the <head> and <body> tags to structure the content, and the meta and title tags to provide essential information to the browser.

To create an HTML project and document, we need to follow a few steps. First, we need to include a title tag. This can be done by using the opening and closing title tags. Inside these tags, we can write the desired title for our website. For example, we can use "Hello World" as the title.

Next, we need to add the body tag. The body tag is where we include the content of our webpage. Inside the body tag, we can add text, images, links, and other elements that we want to display on our webpage.

To test our webpage, we can open the index file in a web browser. Right-click on the index file, select "Open with," and choose a web browser of your choice. For example, we can use Google Chrome.

When we open the webpage in the browser, we will see the content we added inside the body tag. In our case, it will display the text "Hello World." Additionally, the title we set inside the title tag will appear in the tab of the browser.

This is just the beginning of creating a website. In the next episode, we will explore different tags that we can use in HTML to enhance our webpage. These tags will help us add structure, formatting, and interactivity to our website.

To create an HTML project and document, we need to include a title tag and a body tag. The title tag sets the title of the webpage, which appears in the browser tab. The body tag contains the content of the webpage. By opening the index file in a web browser, we can see the webpage and its content.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: GETTING STARTED
TOPIC: HTML ELEMENTS AND ATTRIBUTES

HTML and CSS Fundamentals - Getting started - HTML elements and attributes

In this lesson, we will discuss HTML elements and attributes. HTML elements are also referred to as tags. There are two types of HTML elements: regular HTML elements and empty elements. Regular HTML elements have opening and closing tags, while empty elements do not have a closing tag.

In the previous lesson, we learned about opening and closing tags, which are examples of regular HTML elements. We also mentioned the meta tag, which is an example of an empty element. The meta tag does not have a closing tag and is used to provide information about the website, such as character encoding.

To determine whether an HTML element requires a closing tag or not, it is something you will learn as you gain experience in programming HTML. There is no definitive rule that can be taught beforehand.

In the previous lesson, we also discussed the tags used in the front page of a website, known as the index.html file. We used the HTML tags to define the content of the HTML document. It is recommended to move the content outside of the HTML tags to better visualize the hierarchy of the website.

Similarly, we moved the title tag and the meta tag outside of the head tags in the previous lesson. The head tags contain information that is not directly visible on the webpage, such as the title of the website and meta information.

The body tag defines the visible content of the webpage. Anything placed between the opening and closing body tags will be displayed on the webpage. In the previous lesson, we added the text "Hello World" inside the body tags, and it was displayed on the webpage.

HTML offers a wide range of elements, but we will only cover a few in this lesson. Attributes are additional pieces of information that can be added to HTML tags. In the previous lesson, we used the charset attribute inside the meta tag to specify the character encoding.

Let's create another tag called a paragraph tag. The paragraph tag is used to insert text into the webpage. To create a paragraph tag, we simply write <p> and then press the tab key. Inside the opening and closing paragraph tags, we can add text such as "Hello World" or "This is text." When we refresh the webpage, we can see the text displayed.

By default, web browsers apply default styles to certain tags, such as paragraph tags. If we want to customize the styling of the paragraph tag, we can add attributes to the opening tag. For example, we can add a title attribute by writing <p title="Some text">. This attribute can be used to provide additional information or functionality to the tag.

HTML elements and attributes are essential components of web development. Attributes provide additional information or functionality within a website. Let's consider an example to understand this concept better.

Suppose we have a paragraph of text that we want to style. We can use the "style" attribute inside the paragraph tag to achieve this. By using the "style" attribute, we can add CSS code to modify various properties of the text. For instance, if we want to change the color of the text to red, we can set the "color" property to "red" within the style attribute.

After saving the changes and refreshing the browser, we can see that the text now appears in red. This demonstrates how attributes can be used to modify the appearance of HTML elements.

In this episode, we have discussed a few examples of attributes in HTML. While there are numerous attributes available in HTML, we will focus on two specific ones in this episode. In future lessons, we will explore additional attributes and HTML elements that are commonly used in websites, such as menus and titles.

By understanding and utilizing attributes effectively, you will be able to enhance the functionality and design of your websites. In the next episode, we will delve further into inserting text within a website.

We hope you found this episode informative. Stay tuned for more valuable insights in the upcoming episodes.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: GETTING STARTED
TOPIC: CREATING TITLES AND TEXT USING HTML

HTML and CSS Fundamentals - Getting started - Creating titles and text using HTML

In web development, we use HTML to insert text inside a website. There are two main tags we can use for this purpose: the paragraph tag and the header tag. The paragraph tag, represented by `<p>`, is used to create regular text. By placing text between the opening and closing `<p>` tags, we can display it on the website. For example, `<p>Hello world</p>` will display the text "Hello world" in the browser.

On the other hand, the header tag, represented by `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, and `<h6>`, is used to indicate the importance of the text. The number in the header tag determines its importance, with `<h1>` being the most important and `<h6>` being the least important. These tags are used to tell the browser and search engines like Google which text is the most significant on the website.

For instance, if we have a section on our website that contains articles, we can use the `<h1>` tag to indicate the title of the section, such as "Latest Articles." Then, we can use the `<h2>` tag to label the individual articles within that section. By nesting the text inside the appropriate header tags, we can create a structured and organized website.

It's important to note that we should only use one `<h1>` tag per page. This tag should describe the main purpose or topic of the page. Using multiple `<h1>` tags can be seen as spam by search engines. However, we can use multiple `<h2>`, `<h3>`, and so on, tags to label different sections or subsections of the website.

To illustrate this concept, let's consider an example. We can use the `<section>` tag to group related content together. Within the `<section>` tag, we can use the `<h2>` tag to indicate the title of the section, such as "Latest Posts." Then, we can use the `<h3>` tag to provide a subheader, like "Here you can see all the latest posts." Finally, we can use the `<article>` tag to represent each individual post within the section.

By structuring our website using appropriate header tags, we can convey the importance and hierarchy of the text to both browsers and search engines. This helps improve the accessibility and search engine optimization of our website.

HTML and CSS Fundamentals - Getting started - Creating titles and text using HTML

In HTML, we can create titles and text using different tags. One important tag is the article tag, which groups related content together. It doesn't mean a newspaper article, but rather that all the content inside the article tags is related. For example, we can use the h4 tag inside the article tag to create a post title. The actual post content can be placed inside a paragraph tag. Additionally, we can include the author of the post. The hierarchy of these tags determines their importance on the website.

There is no specific rule for the order of tags, but it is important to choose what makes sense for the content. Once we save the code and refresh the browser, we can see the text inside the website. The latest post will be displayed as the main heading, followed by the sub header, post title, post content, and the author. If we add a second article, it will appear as a second post on the website.

To format the content, we can use the break tag. This tag is an example of an empty element in HTML, as it does not have a closing tag. By adding a break tag inside a paragraph, we can shift the content to the next line, creating a line break. This is useful when we want to avoid having all the content on a single line.

HTML allows us to create titles and text using different tags. The article tag groups related content together, while the h4 tag can be used for post titles and the paragraph tag for post content. The hierarchy of these tags determines their importance on the website. By using the break tag, we can create line breaks to format the content.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: GETTING STARTED
TOPIC: USING BOXES IN WEBSITES

Containers play a crucial role in web development, as they allow us to organize and structure the content of a website. In this lesson, we will explore the concept of containers and how they are used in HTML and CSS to create boxes on websites.

When we visit a website, we can observe that the content is divided into boxes. These boxes help us group related content together and make the website more visually appealing. Let's take a look at a few examples to better understand containers and their significance.

The first website we will examine is called "Night People." On this website, we can see that the content is divided into several boxes. There is a box for the logo, a box for the navigation, a box for the login section, and another box for the main content of the page. Each of these boxes contains specific elements that are related to each other.

Another example is a website with a header at the top. In this case, the header is contained within a white box that encompasses the logo, navigation, and login button. As we scroll down, we encounter a video with accompanying content, which is also enclosed within a box. This pattern continues throughout the website, with various elements being placed inside boxes.

To create boxes in our own websites, we can use HTML elements. For example, the "article" element is designed to contain related content. By using this element, we can group our content together and create a box-like structure. Additionally, we can use the "div" element, which is a generic container that allows us to group content without any specific purpose. This element is particularly useful when we want to group content together for styling purposes.

Styling our containers is done using CSS, which we will cover in the next lesson. CSS allows us to manipulate the appearance and layout of our HTML elements, including the containers we create. By applying CSS styles to our containers, we can control their position, size, color, and other visual properties.

Containers are essential in web development as they enable us to organize and structure the content on our websites. By using HTML elements like "article" and "div," we can create boxes to group related content together. CSS then allows us to style and customize these containers to achieve the desired visual effect.

When developing websites, it is important to understand how to structure and organize content. One way to achieve this is by using boxes, which can group and organize different elements within a webpage. In this didactic material, we will explore how to use boxes in websites using HTML and CSS.

To begin, we can use the `<div>` tag in HTML to create a box. By enclosing elements such as headings (`<h2>`) and paragraphs (`<p>`) within the `<div>` tag, we can group them together inside the box. This allows us to easily manipulate and style the content within the box using CSS.

To better understand how boxes are used in websites, let's take a look at an example. Suppose we have a website with content on the left side and the right side. By inspecting the website's code, we can see that the content is organized into boxes. To inspect the code of a website, you can right-click on the desired section, select "Inspect," and a window will appear displaying the code.

Inside the code, you will notice that each box is represented by a `<div>` tag. By hovering over different sections of the code, you can see that the corresponding content is highlighted. This allows us to identify the different boxes within the code.

For instance, let's focus on a specific section within the code. Inside this section, we can see that everything is contained within one box. However, upon further inspection, we can observe that this box is actually divided into two smaller boxes: a left box and a right box. This division helps to further organize the content within the larger box.

As we progress in this course, we will learn how to utilize these boxes, such as the `<div>` tag, to effectively structure and organize content within our websites. By utilizing different HTML elements and CSS styling, we can create visually appealing and well-structured webpages.

Understanding how to use boxes in websites is crucial for organizing and structuring content. By using the `<div>` tag in HTML and applying CSS styling, we can group and manipulate elements within a webpage. This allows for better organization and presentation of content, resulting in a more visually appealing and user-friendly website.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: ADVANCING IN HTML AND CSS
TOPIC: INCLUDING CSS IN HTML

In this didactic material, we will learn how to use CSS to style websites. CSS, or Cascading Style Sheets, is used to change the appearance of HTML elements in a browser. In the previous episode, we discussed the basics of HTML and CSS. To recap, HTML is used to structure the content of a website, while CSS is used to style that content.

When we view a website, the content is displayed in the browser. We can modify the appearance of this content by adding styling using CSS. In this episode, we will cover two main topics: how to add CSS directly to an HTML file and how to create a separate style sheet.

To add CSS directly to an HTML file, we can use the `<style>` tag. This tag is placed inside the `<head>` tag of the HTML file. Within the `<style>` tag, we can write CSS code to style specific elements. For example, if we want to style an `<h2>` tag, we can write `h2 { ... }` and specify the desired styles inside the curly brackets.

CSS code follows a specific syntax. Each style declaration starts with a keyword, such as `color` to change the text color. After the keyword, we use a colon to indicate what we want to do with the styling. For example, `color: red;` sets the text color to red. It is important to end each style declaration with a semicolon.

In real projects, it is common to use a separate style sheet instead of styling directly in the HTML file. A style sheet is an external CSS file that is linked to the HTML file using the `<link>` tag. This allows for better organization and reusability of styles throughout the website.

Another important concept to consider is the browser's default styling. Browsers like Chrome, Firefox, Edge, and Internet Explorer have default styles applied to HTML elements. These default styles can affect the appearance of our website. To ensure consistent styling, we can use a CSS reset style sheet. This resets the default styles and allows us to start with a clean slate.

In future episodes, we will explore more advanced CSS techniques and best practices for web development. CSS is a powerful tool that allows us to transform the look and feel of our websites, and learning CSS is essential for any web developer.

To include CSS in HTML, we need to follow a few guidelines. Firstly, it is important to add a semicolon at the end of each line of code. This ensures that the code is properly structured and avoids any errors. Secondly, we can save our code and refresh the browser to see the changes.

To change the font size, we can add the code "font-size: 70px;" below the color code. This will make the font size quite large inside the browser. Again, don't forget to add a semicolon at the end of the line. After saving the code and refreshing the browser, we will notice a much bigger font size.

We can also manually remove the margin by adding the code "margin: 0px;" below the font size code. This will eliminate the spacing above and below the text. After saving the code and refreshing the browser, the spacing will disappear.

To add a background color to the text, we can use the code "background-color: blue;". This will give the text a blue background color. After saving the code and refreshing the browser, the text will have a blue background color.

To center the text inside the website, we can use the code "text-align: center;". This will align the text in the center of the website. After saving the code and refreshing the browser, the text will be centered.

In addition to the h2 tag, we can also style the paragraph tag. To do this, we add the code for the paragraph tag inside curly brackets. For example, to change the color of the paragraph text, we can use the code "color: #999999;". This will give the text a dark gray color. After saving the code and refreshing the browser, the text will appear in the specified color.

We can also use different color formats, such as RGB. For example, to give the background color a black color, we can use the code "background-color: RGB(0, 0, 0);". After saving the code and refreshing the browser, the background color will be black. By changing the values of red, green, and blue, we can create different colors.

Another way to add styling directly to HTML elements is by using the "style" attribute. For example, inside the h2 tag, we can add the attribute "style" and specify the CSS code within double quotes. This allows us to directly add CSS styling to specific elements.

In this lesson, we have learned how to include CSS in HTML by adding code to the style tags and using the style attribute. We have seen how to change font size, remove margin, add background color, center text, and style different HTML elements. In future lessons, we will explore more CSS code to further customize our websites.

In web development, it is common to use HTML and CSS to create and style websites. In this lesson, we will explore how to include CSS in HTML to apply styles to different elements on a webpage.

To begin, we can add styles directly inside HTML elements. For example, we can set the background color of a text to a light gray by specifying the hex color code in the style attribute. After saving the changes and refreshing the browser, the text will appear with a light gray background.

However, it is not typical to add styles directly inside HTML elements. Instead, it is recommended to create a separate CSS file that contains all the CSS code. To do this, we can create a new file in the root folder and save it as "style.css". Although the file can be named differently, it is a good practice to name it "style.css" if you plan to use a CMS system like WordPress in the future.

Multiple style sheets can be used in a website, but it is usually not necessary. Having multiple documents open can be confusing. Therefore, it is common to have all the styling inside a single style sheet.

To apply styles from the style sheet, we need to link it to the HTML file. This can be done by adding a link element in the head section of the HTML file. The link element has two attributes: rel, which should be set to "stylesheet", and href, which should specify the path to the style sheet. After saving the changes and refreshing the browser, the styles from the style sheet will be applied to the HTML elements.

In the style sheet, we can define styles for different elements. For example, we can target the h2 tag and give it a different color. By specifying the color property and its value, such as "color: red;", the h2 tag will appear in red. Other elements like the paragraph tag can also be styled in a similar manner.

When targeting specific elements, it is important to consider the hierarchy of the HTML structure. For instance, if we have multiple h2 tags on a webpage, we may not want all of them to have the same style. To specify a specific h2 tag, we can use the hierarchy of elements. By using a div tag as a parent element and an h2 tag inside it, we can apply styles only to the h2 tag inside the div. This can be achieved by adding a space and the h2 selector inside the div selector in the style sheet. By specifying the desired styles, such as "color: green;", the targeted h2 tag will appear in green.

It is important to note that targeting specific elements each time is not the optimal way to apply styles. Instead, we can use classes and IDs to target specific elements in our HTML code. Classes and IDs provide a more efficient and organized way to apply styles to elements.

Including CSS in HTML allows us to style different elements on a webpage. By creating a separate style sheet and linking it to the HTML file, we can define styles for various elements. It is important to consider the hierarchy of elements and use classes and IDs to target specific elements for styling.

In web development, it is important to understand how to include CSS (Cascading Style Sheets) in HTML (Hypertext Markup Language) files. By using classes and IDs, we can target specific elements and apply styling to them.

To demonstrate this, let's consider an example. Inside an HTML file, we can target specific elements using classes and IDs. For instance, we can delete a div tag and go inside an h2 tag. We can then add an attribute called "class" with a specific name, such as "title". This creates an h2 tag with the class "title".

To apply styling to this specific h2 tag, we need to define it in our stylesheet. We can target the tag with the class "title" by using the CSS selector ".title". By specifying the desired styling properties, such as color or font size, we can customize the appearance of the h2 tag with the class "title".

It is worth noting that in this episode, we are only introducing the concept of classes and IDs. We will delve deeper into their usage and impact on websites in future episodes.

Before concluding this episode, let's discuss the concept of a reset stylesheet. When we view a website in different browsers, there may be default styling applied, such as spacing or font sizes. This can result in inconsistencies across browsers. To ensure consistent styling, we need to reset the default browser styling.

To create a reset stylesheet, we can search for a pre-existing code snippet. One popular source is the HTML5 Doctor website, where a person named Ritter Clark has provided a reset code. We can copy this code and paste it at the top of our stylesheet. It is important to place the reset code before any other styling code in order for it to take effect.

By applying the reset stylesheet, we eliminate the default styling from the browser and start with a clean slate. This ensures that our website looks consistent across different browsers.

In this episode, we learned about including CSS in HTML files using classes and IDs. We saw how to target specific elements and apply styling to them. Additionally, we discussed the importance of using a reset stylesheet to eliminate default browser styling and ensure consistency across different browsers.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: ADVANCING IN HTML AND CSS
TOPIC: CREATING HTML AND CSS COMMENTS

Comments in HTML and CSS are essential for organizing and understanding code in web development projects. They provide a way to add explanatory notes that are not rendered on the webpage but are visible in the code. This helps developers and designers keep track of the purpose and functionality of different sections of code.

In HTML, comments are created using the "<!-- -->" syntax. To add a comment, simply place the opening comment tag "<!--" before the code you want to comment, and the closing comment tag "-->" after it. Anything between these tags will be treated as a comment and will not be displayed on the webpage.

For example, if we have an HTML file with an h2 tag and a paragraph tag, we can add a comment to describe the purpose of the code. By placing "<!-- This is the main content -->" after the h2 tag and before the paragraph tag, we create a comment that clarifies the role of the code. The comment is closed with "-->". It is important to note that the comment tags themselves are not part of the comment.

In CSS, comments are created using the "/* */" syntax. To add a comment, place the opening comment tag "/*" before the code and the closing comment tag "*/" after it. Similar to HTML, anything between these tags will be treated as a comment and will not affect the styling of the webpage.

In a CSS file, comments can be used to indicate where specific sections of code begin or to provide additional information about the purpose of the code. For example, if we have a CSS file with a reset style sheet at the top, we can add a comment to mark the start of our custom code. By placing "/* This is my code */" after the reset style sheet and before our custom code, we create a comment that clearly distinguishes our code from the rest. The comment is closed with "*/".

Using comments in HTML and CSS is crucial for maintaining readability and understanding in web development projects. By providing explanations and clarifications within the code, developers can easily navigate and modify complex websites. It is highly recommended to add comments when working on larger projects to ensure code comprehension and collaboration among team members.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: ADVANCING IN HTML AND CSS
TOPIC: INTRODUCTION TO CLASSES AND IDS IN HTML

Classes and IDs are important concepts in HTML and CSS that allow us to target specific elements on our webpages for styling purposes. In previous episodes, we learned how to style our HTML by targeting elements directly in our stylesheet. However, using classes and IDs provides a more efficient way to style individual elements.

When we target elements directly in our stylesheet, such as using "h2" or "paragraph", we apply the styling to every instance of that element on our website. This means that all h2 tags or paragraph tags will have the same styling. If we want to style only a specific element, we can use classes or IDs.

To use classes, we add the "class" attribute to the HTML element we want to target. Inside the attribute, we give it a name, such as "index-h2". Then, in our stylesheet, we can target the class by using ".index-h2". This way, we create a separate class for the element and apply the desired styling only to that specific element. We can also add multiple classes to one element, allowing us to combine different styles.

Another way to use classes is by creating a container element, such as a div, and adding a class to it. Inside the container, we can have other elements, like paragraphs, that we want to style. By targeting the class of the container in our stylesheet, we can apply the desired styling to all elements inside that container. This way, we don't need to add classes to every individual element.

IDs are similar to classes but have some differences. We can add the "id" attribute to an HTML element and give it a name, just like with classes. In our stylesheet, we can target the ID by using "#index-h2". IDs are unique, meaning that they can only be used once on a webpage. Therefore, we usually use IDs when we want to target a specific element that is unique, like a header or a footer.

Classes and IDs are useful tools in HTML and CSS that allow us to target specific elements for styling purposes. Classes are used to target multiple elements or groups of elements, while IDs are used to target unique elements. By using classes and IDs, we can apply styling to specific elements without affecting other elements on our webpage.

In HTML and CSS, classes and IDs are used to style elements and provide unique identifiers for specific sections or elements within a webpage. Classes are used to group elements together and apply the same styles to multiple elements, while IDs are used to uniquely identify a specific element.

When using classes, you can apply styles to multiple elements by assigning the same class to each element. For example, if you have a class called "highlight" that sets the color to green, you can assign this class to multiple elements and they will all have the green color. However, when using IDs, you can only have one ID per HTML element. This means that each ID should be unique within the webpage.

It is important to note that using the same ID name for different elements or using multiple IDs for a single element is not recommended. While some browsers may not show an error message for this, it is best practice to avoid such situations.

So why use IDs when classes can achieve the same styling? IDs have some advantages over classes. One advantage is that IDs can be used for linking to specific sections within a webpage. By creating an ID for a section and using it in a link, clicking on the link will take the user directly to that section of the webpage. This is useful for creating navigation within a single page website.

Another advantage of IDs is their use in JavaScript programming. JavaScript often requires the use of IDs to target specific elements and manipulate their behavior. While this is beyond the scope of this HTML and CSS course, it is important to note that IDs play a crucial role in JavaScript functionality.

To summarize, classes and IDs are important in HTML and CSS for styling elements and providing unique identifiers. Classes are used to group elements and apply the same styles to multiple elements, while IDs are used to uniquely identify specific elements or sections within a webpage. IDs have additional advantages such as linking to specific sections and supporting JavaScript functionality.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: ADVANCING IN HTML AND CSS
TOPIC: STYLING TEXT WITH CSS

Today, we will be learning about styling text within a website using CSS. Text styling is an essential skill in web development, as it allows us to customize the appearance of text to suit our needs. In this lesson, we will explore various examples of text styling to demonstrate how we can change the look of text on a website.

To begin, let's take a look at the basic setup of our website. In our index file, we have a simple structure with an h1 tag and a paragraph tag. Additionally, we have created a link using the anchor tag, which has default styling applied to it. The href attribute within the anchor tag is set to "#" for demonstration purposes.

Now, let's move on to styling these elements. We will start by targeting the h1 tag using CSS selectors. By using the h1 selector followed by curly brackets, we can apply styling properties specifically to the h1 tag.

When it comes to styling text, there are various properties we can use within the curly brackets. Let's focus on properties that include "font-" in their name. The first property we will explore is "font-family". This property allows us to choose the font we want to use for our text. For example, setting the font-family to "Arial" will change the font of the h1 tag to Arial. If the browser does not have Arial installed, we can specify an alternate font using a comma-separated list. For instance, we can set the font-family to "Arial, Times New Roman" to use Times New Roman as a fallback font.

Next, let's look at the "font-size" property. This property allows us to specify the size of the font. We can use different units of measurement, such as pixels, em, or percentages. In our example, we will set the font-size to 22 pixels.

Lastly, we have the "font-style" property, which enables us to make the text italic, normal, or oblique. By setting the font-style property to "italic", we can make the text appear in an italicized format.

By applying these styling properties to the h1 tag, we can see the changes take effect when we refresh the website in our browser. The h1 tag will now have a different font, font size, and font style compared to the default styling of the paragraph and link.

It's important to note that if we don't specify any styling properties for an element, the browser will apply default styling based on its own rules.

Understanding how to style text using CSS is crucial for web developers. By utilizing properties such as font-family, font-size, and font-style, we can customize the appearance of text within a website to create visually appealing designs.

To style text using CSS, there are various properties that can be used. One of these properties is "font-style", which allows us to apply different styles to the text. For example, setting it to "italic" will make the text appear slanted. Another property is "font-weight", which determines the thickness of the text. By adjusting the value between 100 and 900, we can make the text appear thinner or thicker.

To align the text, we can use the "text-align" property. Setting it to "left" will align the text to the left side of the container, while "center" will center the text, and "right" will align it to the right side. There is also an option called "justify", which evenly spaces the text to fill the entire width of the container.

The "text-decoration" property is used to add or remove decorations from the text. By setting it to "none", we can remove any underlines or other decorations. Alternatively, we can use values like "underline", "line-through", or "overline" to apply specific decorations.

The "text-indent" property allows us to indent the first line of a paragraph or a block of text. By specifying a value, such as 30 pixels, the first line will be moved out slightly.

The "text-transform" property is used to change the capitalization of the text. Options like "capitalize" will capitalize the first letter of each word, "uppercase" will make all letters uppercase, and "lowercase" will make all letters lowercase.

To change the color of the text, we can use the "color" property. We can specify a color using a hex code (e.g., #FF0000 for red) or using RGB values.

These CSS properties can be applied to different HTML elements, such as headers, paragraphs, or links, to customize the appearance of the text on a webpage.

When styling text with CSS, there are several properties that can be used to change the appearance of the text. One of the most common properties is the color property, which allows us to change the color of the text. In CSS, colors can be specified using different formats, such as hexadecimal (hex) or RGB values.

To use a hex color, we can use a six-digit code that represents the amount of red, green, and blue in the color. For example, the hex code for red is #FF0000, where FF represents the maximum value for red and 00 represents the minimum value for green and blue. It's important to note that if all the digits in the hex code are the same, we can use just three digits instead of six. For example, #000000 can be written as #000.

Another way to specify colors is by using RGB values. RGB stands for red, green, and blue, and each color component can have a value between 0 and 255. For example, the RGB values for yellow are (255, 244, 96), where 255 represents the maximum value for red, 244 represents the value for green, and 96 represents the value for blue. To use RGB colors in CSS, we can use the rgb() function and provide the values for each color component.

Additionally, we can make the text transparent by using RGBA colors. RGBA stands for red, green, blue, and alpha, where alpha represents the transparency of the color. The alpha value can range from 0 (completely transparent) to 1 (completely opaque). To use RGBA colors in CSS, we can use the rgba() function and provide the values for each color component and the alpha value.

In addition to changing the color of the text, we can also adjust the spacing between letters and words. The letter-spacing property allows us to increase or decrease the space between letters. We can specify the spacing using different units, such as pixels. For example, setting the letter-spacing to 10 pixels will increase the space between each letter. Similarly, we can use the word-spacing property to adjust the space between words.

Lastly, the line-height property is used to control the height of each line of text. By increasing or decreasing the line-height, we can change the spacing between lines in a paragraph. This property is important for improving the readability and aesthetics of the text.

When styling text with CSS, we can use properties like color, letter-spacing, word-spacing, and line-height to change the appearance of the text. By using different color formats like hex or RGB, we can specify the color of the text. Additionally, we can adjust the spacing between letters and words using the letter-spacing and word-spacing properties. Lastly, the line-height property allows us to control the height of each line of text.

To style text with CSS, there are various properties that can be used. One important property is the line-height property, which determines the amount of space between lines of text. This property can be set to a specific value, such as pixels, to control the spacing.

To change the line-height property, you need to select the element you want to style. In this case, we will focus on styling paragraphs. To do this, you can use the paragraph selector in CSS. For example, to change the line-height to 40 pixels, you would use the following code:

```
p {
line-height: 40px;
}
```

Additionally, you can also change the font-size property to adjust the size of the text. For example, to set the font size to 20 pixels, you would use the following code:

```
p {
font-size: 20px;
}
```

By combining these two properties, you can create text with larger size and increased spacing between lines. This can improve the readability of your website.

It's worth noting that the line-height property is not limited to pixels. You can use other units of measurement as well, such as em or percentage. The choice of unit depends on your specific requirements and design preferences.

To see the changes in action, you need to refresh your browser after saving the CSS file. This will update the styling of the paragraphs on your website.

In addition to adjusting the line height and font size, you can also use other CSS properties to style text, such as text-indent to indent the first line of a paragraph. These properties allow you to customize the appearance of text and make your website more visually appealing.

Importing new fonts into your website is another topic that will be covered in the next episode. This is useful when you want to use special fonts that are not commonly available on standard computers. Importing fonts allows you to use them in your website's design.

Styling text with CSS involves using properties like line-height and font-size to control the spacing and size of text. By adjusting these properties, you can enhance the readability and visual appeal of your website. Importing new fonts will be discussed in the next episode.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: ADVANCING IN HTML AND CSS
TOPIC: IMPORTING NEW FONTS

To import new fonts into a website, there are two methods that can be used. The first method involves downloading a font file from an online source and including it in the website's root folder. The font file is then linked to the website using a stylesheet. This method allows the font to be used even without an internet connection. However, it may slightly increase the loading time of the website.

The second method is to link to a font hosted online in a font library. This method allows for faster loading of the font on the website and is also more convenient for developers. However, it requires an internet connection to access the font.

To demonstrate the second method, we will be using Google Fonts, a library provided by Google that offers a wide range of fonts for free use in websites. To start, go to the Google Fonts website and use the search bar to find a font of your choice. Once you have selected a font, click on the plus icon to add it to your collection. At the bottom of the page, you will find a link that needs to be included in your code.

Copy the link and paste it at the top of your stylesheet, ensuring that it is placed above any other style rules. This is important as the font needs to be loaded before it can be used. Next, copy the font family code provided below the link and replace the existing font family in your stylesheet.

After saving the changes, refresh your website and you will see the new font being applied. It is worth mentioning that you can explore the Google Fonts website for a variety of fonts that do not require importing into your website.

By following these steps, you can easily import new fonts into your website and give it a unique and personalized look.

To import new fonts into a website, there are two methods that can be used. The first method involves using Google Fonts. In this method, you can select a font from the Google Fonts library and then use the provided link to import it into your website. To do this, you need to go to the Google Fonts website and choose a font that you like. Once you have selected a font, you can customize it by selecting different weights, such as regular, bold, or italic. After customizing the font, you can click on the "Embed" tab to get the link code. Copy the link code and paste it into your website's index page. You can then go to your stylesheet and select the new font family by copying and pasting it. By refreshing the browser, you will be able to see the new font applied to your website.

The second method involves downloading a font and manually adding it to your website. To do this, you can go to a website that offers free fonts, such as DaFont or FontSquirrel. Choose a font that you like and download it to your computer. Once downloaded, extract the font files and copy them into a new folder called "fonts" in your website's root directory. In your stylesheet, you need to use the @font-face rule to link to the font. Inside the @font-face rule, include the link to the font file using the source URL. Additionally, specify the name that you want to refer to the font as. Finally, you can use the font-family property in your stylesheet to select the font and apply it to the desired elements in your website.

By following these methods, you can import new fonts into your website, allowing you to customize the typography and enhance the visual appeal of your web pages.

When working on web development projects, it is important to have access to a wide variety of fonts to enhance the visual appeal of your website. In this material, we will explore how to import new fonts into your HTML and CSS code.

To begin, it is crucial to find a reliable source for free fonts. There are numerous websites available where you can download fonts for your projects. One popular option is [Website Name], which offers a vast collection of fonts that are free to use.

Once you have selected a font from the website, you will need to import it into your HTML and CSS code. To do this, you will need to obtain the font files, which are typically provided in formats such as .ttf or .otf. These files contain the necessary data for the browser to render the font correctly.

To import the font into your CSS code, you will need to use the `@font-face` rule. This rule allows you to specify the font family name, the location of the font file, and any additional font properties, such as font weight or style.

Here is an example of how to import a font using the `@font-face` rule:

```
@font-face {
  font-family: 'CustomFont';
  src: url('path/to/font-file.ttf');
}
```

In the above example, we have specified the font family name as 'CustomFont' and provided the path to the font file. Make sure to replace 'path/to/font-file.ttf' with the actual path to your font file.

Once you have imported the font, you can use it in your CSS code by specifying the font family. For example:

```
body {
  font-family: 'CustomFont', sans-serif;
}
```

In the above code snippet, we have set the font family of the body element to 'CustomFont'. If the font is not available, the browser will fallback to a generic sans-serif font.

Remember to include the imported font files along with your project files so that they can be accessed by the browser when rendering your web page.

By importing new fonts into your HTML and CSS code, you can enhance the typography of your website and create a more visually appealing user experience.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: ADVANCING IN HTML AND CSS
TOPIC: CREATING SUB PAGES IN HTML

In this lesson, we will learn how to add more pages to a website. So far, we have been using our front page to learn HTML and CSS. However, most websites have more than just one page, so it is important to learn how to create subpages.

To start, we need to go above our root folder and create a new file inside it. The front page of our website should be named "index.html" as we discussed in a previous episode. This is because when we upload our website to a server, the server looks for a file called "index.html" as the front page. However, for subpages, we can name them whatever we like. It is recommended to use lowercase letters and avoid using symbols or anything that may cause issues with the file name. For example, if we want to create a contact page, we can name it "contact.html".

Once we have created the new page, we need to add content to it, just like we did with the front page. We can copy the content from the front page and paste it into the contact page. It is important to include certain tags that are required in any new document. If you are unsure about which tags to include, refer back to the episode where we discussed the necessary tags for creating a new document.

After copying the content, we can make some changes to customize the contact page. For example, we can change the heading to "Contact Page" and modify the paragraph to provide contact information such as name, email, and telephone number.

To view the new page, we can access it by changing the URL in the browser. We can see that the front page and the contact page have different content. However, we haven't created a link to the contact page yet, so in the next episode, we will learn how to create links to navigate between pages without manually changing the URL.

Our goal in this course is to create a portfolio website together using the skills we learn in these lessons. We will gradually build a website that looks professional and includes more than just basic text on a white background.

In this didactic material, we will explore the concept of creating subpages in HTML. Subpages are an essential component of website development, allowing users to navigate through different sections of a website. By the end of this material, you will have a solid understanding of how to create links within your website.

To create a subpage, we need to understand the basic structure of an HTML document. HTML stands for Hypertext Markup Language and is the standard language used for creating web pages. It consists of various elements that define the structure and content of a webpage.

One of the fundamental elements in HTML is the anchor tag, represented by the <a> tag. The anchor tag is used to create hyperlinks, allowing users to navigate to different web pages. To create a link to a subpage, we need to specify the URL or file path of the subpage within the href attribute of the anchor tag.

For example, let's say we have a main webpage called "index.html" and we want to create a subpage called "about.html". To create a link from the main webpage to the subpage, we would use the following code:

```
<a href="about.html">About</a>
```

In this code, "about.html" represents the file path or URL of the subpage. The text "About" within the anchor tag serves as the clickable link that users will see on the main webpage.

It is important to note that the file path or URL should be accurate and accessible. If the subpage is located in a different directory or folder, the file path should reflect that.

Additionally, we can also create links within the same webpage, allowing users to navigate to different sections of the page. To achieve this, we can use the id attribute to identify specific sections or elements within the webpage. We can then create links that point to these sections using the href attribute.

For example, let's say we have a webpage with multiple sections, and we want to create a link that navigates to a specific section called "services". We would first assign an id to the "services" section, like this:

```
<div id="services">
<!-- Section content goes here -->
</div>
```

Then, we can create a link that points to this section using the following code:

```
<a href="#services">Services</a>
```

In this code, the "#" symbol followed by the id "services" represents the link to the specific section within the same webpage.

By understanding how to create links within a website, you can effectively navigate users to different subpages or sections, providing a seamless browsing experience. Remember to ensure accurate file paths or URLs and use the appropriate anchor tag attributes to create these links.

In the next material, we will delve deeper into HTML and CSS concepts, expanding our knowledge in web development. Stay tuned for more valuable lessons!

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: ADVANCING IN HTML AND CSS
TOPIC: CREATING LINKS IN HTML

In this lesson, we will learn how to create links inside a website to navigate between different pages. Links allow users to visit specific pages, such as the "contact.html" page we created in the previous material. In this episode, we will focus on explaining what a link is and how to use it.

To create a link, we need to use an anchor tag. An anchor tag is a pair of text that wraps around the content we want to link to. Inside our front page, we want to include a link that takes the user to the contact page within our website. Currently, we can only access the contact page by manually typing its URL. So, we need to provide a way for users to easily navigate to the contact page without modifying the URL.

To create the link, we will place the anchor tag below the paragraph that says "Welcome to my personal portfolio." The anchor tag is written as "<a></a>". Inside the starting tag, we need to include the "href" attribute, which stands for hyper reference. The "href" attribute contains the link to the page we want to take the user to. In this case, we want to link to the contact page, so we will set the "href" attribute to "contact.html" since the contact page is located in the same folder as the front page.

If the contact page was located in a subfolder, we would need to modify the "href" attribute accordingly. For example, if we created a folder called "pages" and placed the contact page inside it, the "href" attribute would be set to "pages/contact.html". The path to the page we want to link to must be relative to the current document's location.

After setting the "href" attribute, we can insert the text that will serve as the clickable link. For example, we can use the text "Visit my contact page!" between the opening and closing anchor tags. Once we refresh the website, we will see the link. Clicking on it will take us to the contact page.

To provide a way to navigate back to the front page from the contact page, we can copy the anchor tag from the front page and include it at the bottom of the contact page. This time, we will set the "href" attribute to "index.html" to link back to the front page. We can also change the text to "Visit my front page".

Additionally, we can create a link using only a portion of the text. For example, we can copy the opening anchor tag, delete it, and place it before the word "here". This will make the word "here" the only part of the text that links back to the front page.

Lastly, it's important to note that links can also be used to direct users to external websites. We can copy a link from a website, such as "thefont.com", and paste it into the "href" attribute. This will create a link that takes the user to the specified external website.

By using anchor tags, we can create links within our website to navigate between different pages and even link to external websites. This enhances the user experience and makes it easier for visitors to explore the content of our website.

In HTML, we can create links using anchor tags. These anchor tags are represented by the <a> element. To create a link, we wrap the content we want to link to inside the opening and closing anchor tags. For example, if we want to create a link to a website called "font.com", we would write <a href="https://www.font.com">font.com</a>.

By default, when we click on a link, it opens in the same window or tab. However, if we want the link to open in a new window or tab, we can use the "target" attribute. By setting the "target" attribute to "_blank", the link will open in a new tab. For example, <a href="https://www.font.com" target="_blank">font.com</a>.

We can also use external links inside our anchor tags. This means that we can link to websites outside of our own website. When we click on an external link, it is generally a good practice to open it in a new tab, so that users can easily return to our website.

It is not necessary to use text inside a link. We can use other elements, such as a div, as the content of the link.

For example, we can wrap a pair of anchor tags around a div and style it using CSS. This allows us to create a clickable element that behaves like a link. Anything that we insert inside our HTML document between the opening and closing anchor tags can be turned into a link.

In addition to linking to external websites or using other elements as links, we can also use anchor tags as bookmarks. By adding an "id" attribute to an element, we can create a bookmark. We can then create a link that points to this bookmark by using the name of the "id" as the value of the "href" attribute. This allows users to quickly navigate to specific sections of a webpage.

We can create links in HTML by using anchor tags. We can link to external websites, use other elements as links, and create bookmarks within our webpage. Links are created by wrapping the content we want to link to inside the opening and closing anchor tags. By using the "target" attribute, we can control whether the link opens in the same window or in a new tab.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: ADVANCING IN HTML AND CSS
TOPIC: CREATING MENUS IN HTML

In this lesson, we will focus on creating a navigation menu for our website. A navigation menu allows users to easily navigate and access different pages or sections of a website. This is an important element to include in a website, especially if we have learned how to create links and subpages in previous lessons.

To begin, let's understand the purpose of a navigation menu. A navigation menu serves as a roadmap for users, helping them find the information they are looking for quickly and efficiently. It typically appears at the top or side of a website and contains links to various pages or sections.

Now, let's discuss how to create a basic navigation menu in HTML. The most common approach is to use an unordered list (`<ul>`) and list items (`<li>`) to represent each menu item. Each list item will contain a hyperlink (`<a>`) that points to the desired page or section.

Here is an example of a basic navigation menu structure:

```html
<ul>
  <li><a href="home.html">Home</a></li>
  <li><a href="about.html">About</a></li>
  <li><a href="services.html">Services</a></li>
  <li><a href="contact.html">Contact</a></li>
</ul>
```

In this example, we have four menu items: Home, About, Services, and Contact. Each menu item is represented by a list item (`<li>`), and the hyperlink (`<a>`) within the list item specifies the destination page or section using the `href` attribute.

To style the navigation menu, we can use CSS. We can apply different styles to the `<ul>`, `<li>`, and `<a>` elements to achieve the desired visual appearance. For example, we can change the background color, font, and spacing.

Here is an example of CSS code to style the navigation menu:

```css
ul {
  list-style-type: none;
  background-color: #f2f2f2;
  padding: 0;
  margin: 0;
}

li {
  display: inline-block;
  margin-right: 10px;
}

a {
  display: block;
  padding: 10px;
  text-decoration: none;
  color: #333;
}

a:hover {
  background-color: #333;
  color: #fff;
}
```

In this CSS code, we set the `list-style-type` of the `<ul>` element to `none` to remove the default bullet points. We also set the background color, padding, and margin to achieve the desired spacing and appearance.

The `display: inline-block` property is used for the `<li>` elements to make them appear horizontally. The `<a>` elements are styled with padding, text decoration, and color. The `:hover` selector is used to change the background color and text color when hovering over a menu item.

By combining the HTML structure and CSS styling, we can create a functional and visually appealing navigation menu for our website.

A navigation menu is an essential component of a website that allows users to easily navigate and access different pages or sections. By using HTML and CSS, we can create a basic navigation menu structure and style it to match our desired design.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: ADVANCING IN HTML AND CSS
TOPIC: CREATING WRAPPERS IN HTML

A wrapper is a way to group content inside a website. It helps to organize and structure the elements on a webpage. In this lesson, we will learn how to create a wrapper in HTML.

To understand the concept of a wrapper, let's look at some examples. If we visit YouTube and go to a channel page, we can see that all the content below the banner follows a certain line on the left and right side. The content does not extend beyond this line. Similarly, if we visit a personal website, we may notice that the wrapper on the homepage is wider, allowing the content to stretch almost to the left and right sides. However, on subpages like tutorials or portfolio, the wrapper is narrower and follows a specific line.

To create a wrapper, we use the \`<div>\` tag in HTML. We can enclose the content we want to be within the wrapper inside this tag. For example, if we want to create a wrapper for the content on the front page, we can add a \`<div>\` tag before and after the desired content.

```
<div>
  <!-- Content to be within the wrapper -->
</div>
```

By enclosing the content within the \`<div>\` tags, we can apply styles and control the layout of the content inside the wrapper. For instance, we can set a fixed width for the wrapper, which will keep the content within a specific line. The height of the wrapper can be left undefined, allowing it to expand vertically to accommodate the content.

To demonstrate this, let's consider an example where we have a navigation menu on top of the page. Below the navigation, we want to display a heading that indicates the current page. We can add the following code inside the wrapper:

```
<div>
  <nav>
    <!-- Navigation menu -->
  </nav>

  <h1>Front Page</h1>

  <!-- Other content goes here -->
</div>
```

In this example, we have added an \`<h1>\` tag to display the text "Front Page" within the wrapper. Similarly, we can add the same code to other pages, replacing the text accordingly.

After implementing the wrapper, we may notice that the content aligns with the left side of the screen, while the navigation menu has some spacing on the left. This spacing is due to the margin applied to the navigation in the CSS stylesheet. By adjusting the margin value, we can control the spacing between the navigation and the wrapper.

Creating a wrapper allows us to keep the content within a specific line and maintain consistency across different pages of a website. It helps in organizing and structuring the layout of a webpage.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: MULTIMEDIA IN HTML AND CSS
TOPIC: INSERTING IMAGES USING HTML AND CSS

HTML and CSS Fundamentals - Multimedia in HTML and CSS - Inserting images using HTML and CSS

In web development, it is common to include images in websites to enhance the visual appeal and convey information. In this lesson, we will learn how to insert images using HTML and CSS.

To insert an image using HTML, we need to use the "img" tag. This tag is a self-closing tag, meaning it does not require a closing tag. The "img" tag has several attributes, but the most important one is the "src" attribute, which specifies the source or location of the image file.

For example, to insert an image file named "example.jpg" located in the same directory as our HTML file, we would use the following code:

```
<img src="example.jpg" alt="Description of the image">
```

The "alt" attribute is used to provide a text description of the image. This is important for accessibility purposes, as it allows screen readers to describe the image to visually impaired users. It is recommended to always include the "alt" attribute.

In addition to the "src" and "alt" attributes, we can also use other attributes to specify the width, height, and alignment of the image. For example:

```
<img src="example.jpg" alt="Description of the image" width="300" height="200" align="left">
```

In this example, the "width" and "height" attributes specify the dimensions of the image in pixels, while the "align" attribute aligns the image to the left side of the page.

To style the image using CSS, we can use the "style" attribute or an external CSS file. With CSS, we can change the size, position, and other properties of the image. For example:

```
<img src="example.jpg" alt="Description of the image" style="width: 300px; height: 200px; float: left;">
```

In this example, the "style" attribute is used to set the width and height of the image to 300 pixels and 200 pixels, respectively. The "float" property is set to "left" to align the image to the left side of the page.

It is important to note that when inserting images, it is recommended to use descriptive file names and provide meaningful alt text for accessibility purposes. Additionally, it is good practice to optimize images for web by reducing their file size without compromising quality.

By using HTML and CSS, we can easily insert and style images in our web pages, enhancing the overall user experience and visual appeal.

HTML and CSS Fundamentals - Multimedia in HTML and CSS - Inserting images using HTML and CSS

In web development, it is common to include images in a webpage to enhance its visual appeal and provide additional information. In this guide, we will explore how to insert images using HTML and CSS.

To insert an image in HTML, we use the <img> tag. The <img> tag is a self-closing tag that does not require a closing tag. It has several attributes that we can use to specify the source, alt text, width, height, and other properties of the image.

The most important attribute of the <img> tag is the src attribute, which specifies the source URL of the image. The source URL can be a local file or a remote URL. We can also specify alternative text for the image using the alt attribute. The alt text is displayed if the image fails to load or if the user is using a screen reader.

Here is an example of how to insert an image in HTML:

<img src="image.jpg" alt="A beautiful sunset">

In the example above, we have specified the source URL as "image.jpg" and the alt text as "A beautiful sunset". The image file should be in the same directory as the HTML file, or you can provide the full path to the image file if it is located elsewhere.

We can also use CSS to style and position the inserted images. By applying CSS properties to the <img> tag or its parent elements, we can control the size, alignment, borders, and other visual aspects of the image.

For example, we can use the width and height properties to specify the dimensions of the image:

<img src="image.jpg" alt="A beautiful sunset" style="width: 300px; height: 200px;">

In the example above, we have set the width to 300 pixels and the height to 200 pixels. This ensures that the image is displayed at the specified dimensions.

We can also use CSS classes or IDs to target specific images and apply styles to them. By assigning a class or ID to the <img> tag, we can define CSS rules that only affect that particular image.

Here is an example of how to apply a CSS class to an image:

<img src="image.jpg" alt="A beautiful sunset" class="image-style">

In the example above, we have added the class "image-style" to the <img> tag. We can then define CSS rules for the "image-style" class in our CSS file.

Inserting images using HTML and CSS is a fundamental skill in web development. By using the <img> tag and CSS properties, we can easily add and style images in our webpages. Remember to provide alternative text for images to ensure accessibility for all users.

Images are an important component of web development, as they help to enhance the visual appeal and overall user experience of a website. In HTML and CSS, there are various ways to insert images into a webpage.

One common method is by using the `<img>` tag in HTML. This tag allows us to specify the source (URL) of the image, as well as additional attributes such as alt text, width, and height. The alt text is important for accessibility purposes, as it provides a description of the image for users who are visually impaired or unable to view the image.

Here is an example of how to use the `<img>` tag to insert an image into a webpage:

```
<img src="image.jpg" alt="Description of the image" width="500" height="300">
```

In the above example, "image.jpg" is the URL of the image file. The alt text should be replaced with a brief and descriptive text that conveys the content or purpose of the image. The width and height attributes can be adjusted to control the size of the image on the webpage.

Another method of inserting images is by using CSS background images. This technique allows us to set an image as the background of an HTML element. We can specify the image source, position, repeat behavior, and other properties using CSS.

Here is an example of how to use CSS to insert a background image:

```
<style>
    .container {
        background-image: url('image.jpg');
        background-repeat: no-repeat;
        background-size: cover;
    }
</style>

<div class="container">
    <!-- Content of the webpage -->
</div>
```

In the above example, "image.jpg" is the URL of the image file. The `.container` class is applied to the HTML element that we want to have the background image. The `background-repeat` property specifies whether the image should be repeated or not, and the `background-size` property controls how the image should be sized to fit the container.

It is worth noting that when using images in web development, it is important to consider the file size and format. Large image files can slow down the loading speed of a webpage, so it is recommended to optimize images for the web by compressing them or using appropriate file formats such as JPEG or PNG.

Inserting images into a webpage using HTML and CSS is a fundamental skill in web development. Whether it is using the `<img>` tag or CSS background images, understanding how to properly insert and style images can greatly enhance the visual appeal and user experience of a website.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: MULTIMEDIA IN HTML AND CSS
TOPIC: INSERTING HTML5 VIDEOS AND EMBEDDING EXTERNAL VIDEOS

In this lesson, we will learn how to insert videos into a website using two different methods. The first method involves including a video file in the root folder of the website. The second method is by linking to a video online using a technique called embedding.

To start, we will focus on including a video file in the root folder. This method is slightly more challenging when it comes to resizing the video within the website, but it provides more control. In our example, we have a basic index page with a wrapper inside the body tags. The wrapper ensures that all the content stays within a specific width in the middle of the website.

Inside the wrapper, we currently have an h1 tag saying "Welcome to my website." Additionally, we have a style sheet that styles the wrapper and removes any margins or padding inside the website. If we take a look inside the root folder, we can see two files: index.html and style.css. We also have a folder called "video" containing a video file named "tutorial.mp4."

When it comes to browser support, all modern browsers should support mp4 videos. While there are other video formats available, mp4 is widely supported, making it a reliable choice for this lesson. If you have a video and are unsure about the format, mp4 is a safe option.

Now let's proceed to the index page. We will include a video tag, which is a feature introduced in HTML5. Before HTML5, native support for videos in browsers was limited, often requiring the use of Flash. However, with HTML5, we can now include videos directly in the browser without the need for Flash.

To include the video, we add a video tag to the index page. Inside the video tag, we need to specify the source attribute, which will contain the path to the video file. In our case, the video file is located in the "video" folder within the root folder. Therefore, the source attribute should point to "video/tutorial.mp4."

There are additional attributes we can use within the video tag. For example, the autoplay attribute allows the video to play automatically when the user enters the page. If you prefer the user to manually play the video, you can remove the autoplay attribute. Additionally, the poster attribute specifies the thumbnail image for the video. If you want to use a custom thumbnail, you can replace the default thumbnail with your own image.

To enable video controls such as play, pause, and fast-forward, we need to include the controls attribute within the video tag. By adding the controls attribute, the video player will display these controls.

When we refresh the page in the browser, we can now see the video playing with the controls available. The video will also autoplay if we included the autoplay attribute.

That concludes the process of inserting a video file into a website's root folder. In the next lesson, we will explore the second method, which involves embedding external videos.

To insert videos in HTML and CSS, there are two methods: linking to a video file in the root folder or embedding a video from an online platform like YouTube or Vimeo.

To link to a video file in the root folder, we use the `<video>` tag. Inside this tag, we can include a `<source>` tag with the attributes "src" and "type". The "src" attribute specifies the path to the video file, and the "type" attribute tells the browser the video format. For example, if the video is in mp4 format, we set the "type" attribute to "video/mp4". If the browser does not support the video format, we can display an error message using the `<video>` tag.

We can also customize the width of the video by adding a class or targeting the `<video>` tag in the stylesheet. By setting the width to 100%, the video will adjust its size to fit within its container.

To embed a video from an online platform like YouTube or Vimeo, we use the embedding feature provided by these platforms. On YouTube, for example, we can go to the video we want to embed, click on the "Share" button, select "Embed", and customize the options like showing suggested videos, player controls, and video title. After customizing, we can copy the iframe code provided and paste it into our HTML file instead of the `<video>` tag.

Linking to videos online has advantages such as faster loading times and saving server space. By embedding videos, we don't need to upload the video file to our server, reducing the space usage.

To insert videos in HTML and CSS, we can either link to a video file in the root folder using the `<video>` tag or embed a video from online platforms like YouTube or Vimeo using the iframe code provided.

To insert HTML5 videos and embed external videos in a website, we can use iframes. However, if we want the video to be a hundred percent width of the container, we need to do it in a slightly different way. By adding a width attribute to the iframe in the CSS stylesheet, we can achieve a hundred percent width. However, the height may be cut off. To solve this issue, we can wrap the iframe inside a div container.

To do this, we create a basic div and wrap it around the iframe. We give the div a class name, such as "video wrapper". In the CSS stylesheet, we delete the iframe styling and instead style the wrapping div. The purpose of creating a container around the video is to make it scale the height to the width inside the browser.

To achieve this, we set the position of the div to relative. This allows us to move the elements inside the website without using margins and paddings. We then set a padding-bottom to a percentage, such as 36.25%, and a padding-top to a fixed value, such as 25 pixels. Finally, we set the height to 0 pixels.

By creating this container and styling it accordingly, we ensure that the video iframe has the exact same width and height as a normal video would have inside YouTube. This allows the video to scale properly within the website.

To insert HTML5 videos and embed external videos in web development, we need to follow certain steps. First, we need to create a video wrapper that will contain the video. This wrapper should have the same size as the video itself. To achieve this, we set the width and height of the video to 100% of the wrapper.

To do this, we can use CSS. We create a video wrapper and an iframe inside it. The iframe will be styled using absolute positioning. It is important to note that the video wrapper should have a position of relative, as it will serve as the reference point for the absolute positioning of the iframe.

The iframe's positioning is determined by the left, top, right, and bottom properties. To make the video fill the entire wrapper, we set these properties to 0 pixels. This ensures that the video is positioned at the left, top, right, and bottom edges of the wrapper.

To make the video responsive, we set the width and height of the video to 100%. This allows the video to scale proportionally with the width of the wrapper. By doing this, the video will adjust its height accordingly, maintaining its aspect ratio.

For embedding external videos, such as from YouTube or Vimeo, we can use their embed codes. For example, in the case of Vimeo, we can find the embed code by clicking on the "Share" button and selecting the "Embed" option. This will provide us with an iframe code that we can copy and paste into our HTML code.

It is important to note that this method is necessary because, as of the creation of this material, there is no direct support for scaling the height of videos in HTML5. Therefore, this workaround is required to achieve a responsive video.

In the next episode, we will delve into responsive design, which is essential for creating websites that adapt to different screen sizes. This will ensure that our videos and other content are displayed properly on mobile phones and tablets.

In web development, it is crucial to understand how to insert videos into HTML and CSS. This allows us to enhance the multimedia experience on our websites. In this lesson, we will focus on inserting HTML5 videos and embedding external videos.

To insert an HTML5 video, we can use the <video> element. Within this element, we can specify the source of the video using the "src" attribute. Additionally, we can add controls, such as play, pause, and volume, by including the "controls" attribute. It is also possible to set the width and height of the video using the "width" and "height" attributes.

When embedding external videos, we can use iframes. An iframe is an HTML element that allows us to embed external content within our website. To embed a video, we need to obtain the embed code from the video hosting platform, such as YouTube or Vimeo. By pasting this code within the iframe element, the video will be displayed on our website.

To make the video responsive, we can use CSS. By setting the width of the video's container to a percentage value instead of a fixed pixel value, the video will adjust its size according to the browser window. This is known as responsive design. It allows the video to adapt to different screen sizes, ensuring a consistent viewing experience across devices.

Responsive design is essential in today's mobile-dominated world. With the increasing use of smartphones and tablets, it is crucial to have websites that can be easily accessed and viewed on these devices. By making our websites responsive, we can ensure that our content, including videos, is accessible and enjoyable for all users.

In the next lesson, we will delve further into responsive design and explore techniques to optimize websites for different devices. It is important to understand these concepts to create user-friendly and engaging web experiences.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: RESPONSIVE WEBSITES
TOPIC: INTRODUCTION TO RESPONSIVE WEBSITES

Responsive websites are a crucial aspect of modern web development. With the increasing use of mobile devices to access the internet, it is essential to ensure that websites are designed to adapt to different screen sizes and resolutions. In this material, we will introduce the concept of responsive websites and discuss the fundamentals of HTML and CSS that enable us to create responsive designs.

Responsive websites are designed to provide an optimal viewing experience across a wide range of devices, from desktop computers to smartphones and tablets. The goal is to ensure that the content and layout of a website automatically adjust to fit the screen size, without compromising usability or readability.

To achieve responsiveness, we rely on HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets). HTML is the standard markup language used to structure the content of web pages, while CSS is used to control the presentation and layout. By applying CSS media queries, we can define different styles for different devices, allowing us to create responsive designs.

Media queries are CSS rules that apply specific styles based on the characteristics of the device, such as screen width or orientation. For example, we can use media queries to specify that certain elements should have a larger font size on mobile devices, or that a navigation menu should be displayed as a dropdown on smaller screens.

In addition to media queries, we can use other CSS techniques to create responsive layouts. Flexbox and CSS Grid are powerful tools that allow us to create flexible and adaptive grid-based layouts. These techniques enable us to easily rearrange and resize elements based on the available space, providing a seamless user experience across devices.

When developing responsive websites, it is important to consider performance as well. Optimizing images and minimizing the use of unnecessary resources can help improve loading times and reduce bandwidth usage, especially on mobile devices with limited connectivity.

Responsive websites are designed to adapt to different screen sizes and resolutions, providing an optimal user experience across devices. By utilizing HTML and CSS, we can create responsive designs that automatically adjust the layout and presentation based on the characteristics of the device. Media queries, flexbox, and CSS grid are some of the techniques that enable us to achieve responsiveness. Considering performance is also crucial for delivering a fast and efficient experience to users.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: RESPONSIVE WEBSITES
TOPIC: CREATING A RESPONSIVE WEBSITE USING HTML AND CSS

In this didactic material, we will learn about creating a responsive website using HTML and CSS. We will focus on building the front page of a website, both for desktop and mobile versions.

To start, let's understand the basic design we will be working with. The design consists of a header with a navigation menu and a logo. Above the banner, there are a few links that lead to different subpages. Below the banner, we have a footer.

The subpages include a "Cases" page, where you can showcase different projects or portfolios, and a "Contact" page, which includes a phone number and email address. Clicking on one of the subpages will take us to a case page, where specific projects can be displayed along with a short description.

Before we begin coding, it's important to note that we should always design for mobile first. In this case, we will focus on the mobile version initially and then expand to the desktop version using responsive design techniques with CSS media queries.

Although a tablet version is not included in this design, it is recommended to create one as there is a significant difference between mobile and desktop designs.

Now, let's proceed to the coding part. But before that, let's take a look at the resources we have. These include a video that will be used on the case page and some images that will be used throughout the website.

It's worth mentioning that the navigation menu has a slight difference between the desktop and mobile versions. Therefore, we will focus on the desktop version when it comes to creating the navigation menu.

With this understanding, we are now ready to start coding the responsive website using HTML and CSS.

In web development, creating responsive websites using HTML and CSS is essential to ensure that a website looks and functions well across different devices and screen sizes. In this didactic material, we will focus on the process of creating a responsive website using HTML and CSS.

To begin, it is important to have an index page and a style sheet. The style sheet contains the necessary code to reset the browser's default styling, ensuring consistent appearance across different browsers. This can be easily obtained from the internet.

Next, we will start setting up the design, starting with the mobile version. In the mobile version, it is common to have a navigation menu that is accessed by clicking a button. However, since we are not covering JavaScript or other coding languages in this material, we will create a basic navigation using buttons. This navigation should remain visible as the user scrolls down the page.

To implement this, we will start by creating the necessary tags in the HTML code. Within the body tags, we will create a header tag, a main tag for the main content, and a footer tag. Inside the header tag, we will include the logo of the website. In this example, the logo is called "mmm toots," but you can use any name you prefer. Additionally, we will include a navigation menu with links.

In the desktop version of the website, the header includes a logo, a navigation bar, and an additional link. We will focus on the desktop version for the header design. The header has a white background, a bar that spans the entire width, a logo, a navigation menu, and a link on the right side. To achieve this, we can use CSS to style the header accordingly.

It is worth noting that different fonts can be used to enhance the design. In this example, the fonts "catamaran" and "cormorants garamond" are used. These fonts can be obtained from Google Fonts.

To create the logo, we will use a paragraph tag and add the text "mmm toots." To make the logo clickable and link it to the front page, we will change the paragraph tag to an anchor tag and set the hyperlink reference to "index.html."

Next, we will create the navigation menu using a nav tag. Inside the nav tag, we will create an unordered list (ul) with list items (li) for each menu item. In this example, the menu items are "portfolio," "about me," and "contacts." You can customize these menu items as needed.

Once the HTML structure is in place, we can use CSS to style the header, including the background color, the bar, the logo, and the navigation menu.

By following these steps, we can create a responsive website with a header that includes a logo, a navigation menu, and a link. The website will adapt to different screen sizes and devices, ensuring a consistent and user-friendly experience.

In this lesson, we will learn how to create a responsive website using HTML and CSS. We will start by changing the placeholder link in our HTML code to a specific link, such as "portfolio.html" for the portfolio section and "about.html" for the about me section. It is important to choose meaningful names for our files to make them more organized.

Next, we will include the navigation links in our HTML code. We have a link on the right side of our header, which will be a basic individual link. We will link it to a "cases.html" file. To differentiate the styling for this link, we will add a class to it. The class for the logo link can be named "header-brand" and the class for the cases link can be named "header-cases".

To see the progress of our website, we will focus on designing for the mobile version. We can do this by inspecting the website in the browser and toggling the device toolbar to a mobile device, such as an iPhone. This will allow us to see how the website looks on a mobile screen and design accordingly.

In our CSS stylesheet, we will start by setting a background color for the body of the website. This will help us visualize the white background color we will add to the header. We will set the body's background color to a slightly gray color, specifically "#f3f3f3".

Next, we will style the header section. We will set the background color of the header to white, the width to 100% so that it spans the entire width of the screen, and the height to 100 pixels.

It is important to include the meta viewport tag in the head section of our HTML code. This tag ensures that our website is displayed properly on different devices and supports responsive design.

Finally, we will start styling the brand inside the header. This can be done by adding additional CSS code to our stylesheet.

By following these steps, we can create a responsive website using HTML and CSS. We will continue to design and style the website in the upcoming lessons.

To create a responsive website using HTML and CSS, we will start by styling the logo. Inside the header, we will create a class called "header-brands" and apply styling to the fonts of the logo. We will set the font-family to "Arial", font-size to 24 pixels, and the color to a slightly off-black color (hex code #111111). Additionally, we will set the font weight to 900 to make the text slightly bold.

To make the text uppercase, we can either modify the text directly in the HTML code or use CSS to transform the text. In this case, we will use CSS by adding the "text-transform: uppercase;" property to the logo styling.

Next, we will remove the underline from all links on the website. We can achieve this by targeting all links and setting the "text-decoration" property to "none".

To center the link in the mobile version of the website, we will display the link as a block element and use margin: 0 auto; to center it horizontally. Additionally, we will set the text-align property to center to ensure the text is centered within the link.

To create spacing at the top of the website, we will add padding to the header. We will set the padding-top and padding-bottom to 20 pixels to create spacing at the top and bottom of the header. We will also add padding-left and padding-right to create spacing from the left and right edges of the header.

Moving on to styling the navigation, we will target the header with the nav tag inside it, followed by the unordered list (ul) and list items (li). Inside the list items, we will target the links (a) and apply the necessary styling.

To display the links next to each other, we will set the display property of the list items to "inline-block". Additionally, we will set the float property to "left" to ensure the links are aligned horizontally. Using float in this case helps eliminate any gaps between the links.

By applying these CSS styles, we can create a responsive website with a styled logo and navigation.

To create a responsive website using HTML and CSS, there are several important steps to follow. One of the key aspects of a responsive website is the ability to adjust its layout and design based on the device or screen size it is being viewed on. In this didactic material, we will focus on creating a responsive navigation menu and a banner with text.

To start, we need to remove the bullet points from the navigation menu. This can be done by setting the list style to none. Additionally, we want to adjust the font weight and size of the links to match the rest of the website. By removing the font weight and setting the font size to 16 pixels, we can achieve the desired result.

Next, we want to remove a specific link called "cases" from the mobile version of the website. This can be done by adding a class to the header element and setting its display property to none. This will hide the link from the mobile version.

To style the navigation menu, we need to create some spacing between the links. This can be achieved by setting the margin of the list items. By setting the top and bottom margin to zero pixels and the left and right margin to 16 pixels, we can create the desired spacing.

To center the navigation menu, we need to set the unordered list to have a margin of 0 auto. Additionally, we need to add the display property with a value of block to ensure proper centering.

However, we may encounter an issue where the navigation menu takes up the full width of the header. To fix this, we need to set the width of the unordered list to fit its content. By setting the width to "fit-content", the navigation menu will adjust its width to fit the content inside it.

Moving on to the banner with text, we need to add a background image instead of an actual image. This can be achieved by adding a div or a section element. In this case, we will use a section element. Inside the section, we can include the text that needs to be displayed on the banner.

In the design, there are two texts on the banner. The top text, "I'm a freelance web developer," will be included as an h2 tag. The bottom text will be included as an h1 tag. By copying and pasting the text into the respective tags, we can display the desired text on the banner.

Remember to use appropriate HTML tags and attributes to structure the content correctly. Additionally, use CSS to style the elements as needed, including adjusting font properties, margins, and widths.

By following these steps, you can create a responsive website using HTML and CSS, with a customized navigation menu and a banner with text.

To create a responsive website using HTML and CSS, we need to follow certain steps. In this tutorial, we will focus on styling the index page and creating a responsive banner with a background image.

First, we need to add a class to the section tag in our HTML code. We can name it "index-banner" to indicate that it is specific to the index page. This will help us target this section in our CSS stylesheet.

In the CSS stylesheet, we can define the styles for the "index-banner" class. We want the banner to have a width of 100% and a height equal to the device height minus the header height. We can achieve this by using the "height: 100vh" property, where "vh" stands for view height. This ensures that the banner will always fill the entire height of the browser window.

To add a background image to the banner, we can use the "background-image" property and specify the URL of the image file. In this example, let's assume we have an image folder in our root directory, and the image file is named "banner.jpg". We can set the background image by using the following code:

```
background-image: url('image/banner.jpg');
```

To ensure the background image is displayed correctly, we can set the "background-repeat" property to "no-repeat" to prevent the image from repeating, and set the "background-position" property to "center" to center the image within the banner. Additionally, setting the "background-size" property to "cover" ensures that the image will cover the entire banner, regardless of its size.

After applying these styles, we can preview the changes in the browser. The banner should now have a responsive background image that fills the entire height of the browser window.

One thing to note is that when using the "100vh" height property, the banner might extend slightly below the visible area of the website if there is a header above it. To fix this, we can use the "calc" function in CSS to subtract the height of the header from the banner's height. In this example, let's assume the header has a height of 100 pixels. We can calculate the new height using the following code:

```
height: calc(100vh - 100px);
```

By using the "calc" function, the banner will now adjust its height to fit the remaining space after subtracting the header's height. This ensures that the entire banner is visible within the browser window, without any unnecessary scrolling.

Finally, we can style the text inside the banner. Assuming we have an "h2" tag inside the "index-banner" class, we can apply styles to it by targeting the "index-banner h2" selector. For example, we can set the font size to 60 pixels and choose a specific font family, such as "Catamaran".

By following these steps, we can create a responsive website with a banner that adapts to different screen sizes and displays a background image. This approach allows us to create visually appealing and user-friendly websites using HTML and CSS.

To create a responsive website using HTML and CSS, we need to follow certain steps. In this guide, we will cover the process of creating a responsive website and discuss the important HTML and CSS fundamentals involved.

First, let's start by understanding the concept of responsive design. A responsive website is designed to adapt and display properly on different devices and screen sizes. This ensures that the website looks good and functions well on desktops, laptops, tablets, and mobile phones.

To begin, we need to define the basic structure of our website using HTML. HTML (Hypertext Markup Language) is the standard markup language for creating web pages. It provides a structure for organizing content on a webpage.

Next, we will focus on the CSS (Cascading Style Sheets) aspect of our website. CSS is used to style and format the HTML elements. It allows us to control the layout, colors, fonts, and other visual aspects of our website.

One important aspect of creating a responsive website is ensuring that the content adapts to different screen sizes. This can be achieved by using media queries in CSS. Media queries allow us to apply different styles based on the characteristics of the device or screen.

To demonstrate this, let's consider an example where we want to create a responsive heading for our website. We want the heading to be 60 pixels in size, bold, and centered. We also want to add a text shadow to make it stand out.

To achieve this, we can write the following CSS code:

```
h1 {
  color: white;
  font-size: 60px;
  font-weight: 900;
  text-align: center;
  text-shadow: 2px 2px 8px #111;
}
```

In the above code, we target the `h1` element and apply the desired styles. The `color` property sets the text color to white. The `font-size` property sets the font size to 60 pixels. The `font-weight` property sets the font weight to 900, making it bold. The `text-align` property centers the text. Finally, the `text-shadow` property adds a shadow to the text.

By using media queries, we can further enhance the responsiveness of our website. For example, we can change the font size and spacing for smaller screens to improve readability.

In addition to the heading, we can apply similar techniques to other elements on our website to make them responsive as well. By using HTML and CSS together, we can create a fully responsive website that adapts to different devices and screen sizes.

Remember, responsive design is an important aspect of modern web development. It ensures that our websites are accessible and user-friendly across various devices. By following the HTML and CSS fundamentals discussed in this guide, you can create your own responsive websites.

To create a responsive website using HTML and CSS, we can use various techniques. One of these techniques involves using tables and table cells to center the content.

To center the text inside a container, we can set the container's display property to "table" and the text's display property to "table-cell". We can also use the vertical-align property to align the text vertically in the middle.

In the example given, the index banner is set to display: table, and the container inside it is set to display: table-cell with vertical-align: middle. This effectively centers the text inside the browser.

It's important to note that the technical details of how this centering is achieved are not discussed in this material. However, a future episode will cover table cells in more depth.

Moving on to the next section of the website, the transcript mentions the need to include different links and a footer. To create this section, we can add a new section below the index banner section in the HTML code.

Inside this new section, we can create div elements to represent the different boxes that will link to other pages. In the given example, there are six boxes mentioned. Each box can be assigned a class name for styling purposes.

Inside each box, we can add an h3 tag to display the title of the link. The transcript lists the titles of the links as "cases", "portfolio", "mm toots", "youtube channel", "about", and "contact".

To create multiple boxes with similar styling, we can use the copy-paste method. The class names and titles can be modified accordingly for each box.

Once the HTML structure is in place, we can move on to styling the boxes in CSS. In the given example, a class name is used to target the first box, and the styling rules are applied inside curly brackets.

The styling includes setting a margin of 10 pixels, a width of 100% minus 20 pixels (to account for the margin on both sides), a height of 100 pixels, and a background color of #f2f2f2 (a light grey color).

It's important to note that the transcript does not provide complete styling information, so this is just an example to demonstrate the concept.

To create a responsive website using HTML and CSS, we can use tables and table cells to center the content. We can also create sections with boxes that link to other pages. The styling of these elements can be done using CSS.

To create a responsive website using HTML and CSS, we need to follow certain steps. Let's go through them.

First, we need to ensure that our website has enough spacing. We can do this by adding margins to our elements. In our stylesheet, we can set the margin to a specific value, such as 20 pixels. However, it's important to test and adjust this value to achieve the desired spacing. Refreshing the browser will show the changes.

Next, we need to add spacing between different buttons. To do this, we can modify the margin property. By setting the top and bottom margins to 16 pixels and the left and right margins to zero, we can achieve the desired spacing.

Now, let's address the use of different class names for each box. Although it may seem unnecessary, it actually serves a purpose. In the desktop version of our website, we have boxes of different sizes. By assigning different class names to these boxes, we can differentiate them in our CSS and apply specific styles accordingly.

To simplify this, we can reduce the number of class names. By examining our design, we can identify which boxes have similar sizes. We can then assign the same class name to these boxes. This minimizes our code and makes it easier to manage.

Additionally, we should ensure that all sections have appropriate class names. By copying the class name from a previous section and applying it to a new section, we can ensure consistent styling throughout our website.

To style the link names, we can copy the CSS properties used for the banner heading and apply them to the link headings. This includes setting the font weight, font style, font type, color, and text decoration.

To avoid repeating the color property for each paragraph or heading, we can create a general styling for paragraphs using the "p" selector. By setting the color property to a specific value, such as #111, we can apply it to all paragraphs without the need to specify the color individually.

Remember, it's important to test and adjust the styles as needed to achieve the desired look and feel for your responsive website.

To create a responsive website using HTML and CSS, we need to follow a few steps. First, we need to ensure that the text is styled correctly. We can make it uppercase by using the CSS property "text-transform: uppercase". Additionally, we can adjust the line height to 100 pixels to match the height of the box.

Next, we should make the text thicker by changing the font weight. We can experiment with different values, such as 300 or 600, until we achieve the desired thickness.

To make the links clickable, we need to wrap them in anchor tags. We can do this by adding an opening and closing tag around each div element containing the boxes. We can then copy the anchor tags and paste them inside each box div. By changing the href attribute of the anchor tags, we can specify the destination of each link.

After completing the links, we can move on to the footer section. We can add the necessary content to the footer by going to the index page and locating the footer section. We can add menus and links by using unordered lists and list items. For the desktop version, we can hide the second menu using CSS, similar to how we hid the header menu.

Inside the footer, we can add different parts such as menus and text. We can copy and paste the necessary HTML code from the design file and modify it as needed. For example, we can add home, cases, about me, and contact links to the first menu. We can also add a "Latest Cases" title for the second menu.

Finally, we can add the text and images to the footer. We can copy and paste the text from the design file and delete any unnecessary parts. We can also include the images by referencing their file paths.

By following these steps, we can create a responsive website using HTML and CSS. We can adjust the styling, add links, and include content in the footer to make the website functional and visually appealing.

To create a responsive website using HTML and CSS, there are several steps involved. First, you need to gather the necessary images for your website. In this case, we will be using Facebook, Twitter, and YouTube icons. Once you have downloaded these icons, you can proceed to code and link them to your index page.

To do this, open your index page and locate the section where you want to add the icons. You can create a div box named "footer - social media" to contain the icons. Inside this div box, you will use anchor tags to create links and image tags to display the icons. The source attribute of the image tag should point to the image file path and name. Additionally, you should include an alt tag to provide a description of the image for accessibility and search engine optimization purposes.

For example, you can use the following code to add the YouTube, Facebook, and Twitter icons:

```
<div class="footer-social-media">
  <a href="[YouTube URL]"><img src="images/youtube.png" alt="YouTube logo"></a>
  <a href="[Facebook URL]"><img src="images/facebook.png" alt="Facebook logo"></a>
  <a href="[Twitter URL]"><img src="images/twitter.png" alt="Twitter logo"></a>
</div>
```

Make sure to replace [YouTube URL], [Facebook URL], and [Twitter URL] with the actual URLs of your social media accounts.

To style the footer section, you can add CSS rules to your stylesheet. Set the width of the footer to 100% and use padding instead of height to allow for flexible sizing based on the content. For example, you can use the following CSS code:

```
.footer {
  width: 100%;
  padding: 40px 0;
  background-color: #111;
  margin-top: -20px;
}
```

This code sets the background color of the footer to a dark color (#111) and adds some spacing between the footer and the content above it.

To style the menu sections, you can use CSS selectors to target specific elements. For example, if you want to style the footer menu, you can use the following CSS code:

```
.footer ul {
  /* CSS rules for the footer menu */
}
```

By using CSS, you can customize the appearance of your website and make it responsive to different screen sizes and devices.

To create a responsive website using HTML and CSS, you need to gather the necessary images, code and link them to your index page, and style the different sections of your website using CSS.

To create a responsive website using HTML and CSS, there are several steps you need to follow. First, let's start with the navigation menu. In order to make it responsive, we need to create a media query that will adjust the layout for smaller screens. Inside the media query, we set the width of the menu to 100% so that it takes up the full width of the screen. We also add a padding to the left of the menu to create some space.

Next, we need to style the list items and anchor text. We can do this by targeting the "li" and "a" elements. We can copy and paste the styling from the first navigation menu and apply it to the second one as well. We remove the float property, set the list style to none, and adjust the padding and font size. We can also change the color of the text to white.

After refreshing the browser, you will see two menus in the footer. However, we need to make some adjustments. First, we need to add more spacing between the links. To do this, we add a line-height property to the link styling and set it to a desired value, such as 20 pixels.

Additionally, we need to remove the second menu. We can achieve this by applying a class to the different menus in the index page. In the stylesheet, we set the display property of the second menu class to "none". After refreshing the browser, you will notice that the second menu is no longer visible.

Now, let's take care of the images in the footer. We can create a div box with a class of "footer-img". Inside the div, we set the width to the desired size, such as 60 pixels, and float it to the right. We then style the images inside the div by setting the width to 100% and removing the float property. We can also add some spacing below the images by setting a margin-bottom or padding property.

After refreshing the browser, you may notice some layout issues where the content inside the footer is not recognized properly. To fix this, we add an "overflow: hidden" property to the container, which in this case is the footer. This will ensure that the container recognizes the content inside it.

Finally, we can make further adjustments to the image sizes by modifying the width property of the "footer-img" class. After refreshing the browser, you should see the images displayed in the desired size.

To create a responsive website using HTML and CSS, we need to make sure that the website looks good and functions properly on different devices, such as mobile phones, tablets, and desktop computers. In this didactic material, we will focus on creating a responsive website by making changes to the code.

First, let's start with the mobile version of the website. We notice that there is too much spacing underneath the menu. To fix this, we can change the line height to 40 pixels. After refreshing the website, we can see that it looks much better.

Next, we want to make the website fit for a desktop version. Currently, if we view the website on a desktop, it still looks like the mobile version. To change this, we need to add a media query to the code. A media query allows us to apply different styles based on the device's screen size.

We will add the media query after the header section. Inside the media query, we will set a width for when the contents inside the website start changing. In this case, we will set the width to around a thousand pixels, which is typically when the website transitions from mobile to desktop view.

Once inside the media query, we need to rearrange the styling to make everything look correct. For example, the logo should be on the left side, and the navigation should be on the right side of the logo. We also need to adjust the padding and alignment.

To center the logo inside the header, we can set the padding to zero on the left side and a specific value on the right side. We can also add a border on the right side of the logo to create a visual effect.

To ensure that the logo and navigation are aligned correctly, we can use the float property. By floating the logo to the left, we can fix the issue of the border being misaligned.

After making these changes, we can refresh the website and see that the logo and navigation are now properly aligned and styled for the desktop version.

By designing for mobile first and then making adjustments for desktop, we ensure that the website remains responsive and looks good on different devices.

To create a responsive website using HTML and CSS, there are several steps we need to follow. First, we need to adjust the positioning and spacing of elements on the page. We can do this by modifying the padding, margins, and line height properties.

To start, let's focus on the logo at the top of the page. We want to center it vertically and horizontally. To achieve this, we can set the padding on the top and bottom to zero and the left padding to 40 pixels. This will create some space on the left side of the logo. Next, we can adjust the height of the border using the line height property. Setting it to 30 pixels will make the border slightly longer. We can then calculate the amount we need to push down the logo by subtracting the height of the border from the total height of the header. In this case, it is 100 pixels - 38 pixels = 62 pixels. We can divide this by 2 to get 31 pixels, which we can set as the top margin of the logo. This will center the logo vertically.

Moving on to the navigation, we want to center it horizontally. We can copy the navigation code and paste it inside a media query for smaller screens. We can then set the margin to zero to remove the centering effect and set the float property to left to position it on the right side of the logo. To push it down, we can set the line height of the anchor tag to 60 pixels. Similar to the logo, we can calculate the remaining space in the header and divide it equally between the top and bottom margins of the navigation. In this case, it is 100% - 60 pixels = 40 pixels. We can set the top and bottom margins of the unordered list to 20 pixels to achieve this.

To add spacing between the portfolio and the line, we can set the left margin to 20 pixels. Additionally, we can make the text uppercase by setting the text-transform property to uppercase in the mobile version styling.

Finally, we need to make the link on the right side of the header visible. To do this, we can set the display property of the header cases to block in the mobile version styling.

By following these steps, we can create a responsive website using HTML and CSS. The adjustments made to the padding, margins, line height, and display properties ensure that the elements are positioned correctly and spaced appropriately.

When creating a responsive website using HTML and CSS, there are several key elements to consider. One important aspect is the styling of the anchor tag. In this case, we don't need to make any changes to the font family, size, or color. However, we do need to keep the line height consistent.

To add styling to the anchor tag, we can copy the existing styling from the navigation section in the mobile version and paste it here. This will ensure that the anchor tag appears correctly within the website. Additionally, we can add a border box around the anchor tag to match the design.

To achieve this, we can set a border with a thickness of one pixel and a solid color. Initially, we can set the border color to #111. However, upon checking the appearance, we notice that it disrupts the layout. To fix this, we can give the anchor tag a float property with a value of "right". This will push it to the right side of the browser.

Next, we want to adjust the height and width of the anchor tag. To push it from the right side, we can set a margin-right of 40 pixels, similar to the spacing we used for the logo from the left side. Additionally, we can change the width by adding padding. A padding of 0 pixels on the top and bottom and 20 pixels on the left and right should work well.

To ensure that the text is always centered inside the button, we can set the width accordingly. However, upon refreshing the browser, we notice that there is an issue with the height. To address this, we can set the line height to 38 pixels, matching the height of the line near the logo.

To further adjust the positioning of the button, we need to calculate the margin-top value. By subtracting 38 pixels from the height of the header (which is 100 pixels), dividing the result by 2, and adding it to the top and bottom margins, we can center the button vertically. Additionally, we need to consider that adding a border will increase the height by 1 pixel at the top and bottom. Therefore, we need to adjust the margin-top value to 30 pixels.

With these changes, the button should now be centered and properly positioned. When we resize the browser, we can see that the menu becomes responsive at a width of 1000 pixels, as defined in the media query.

Moving on, we want to change the height of the banner inside the website. To achieve this, we can set the height to 500 pixels, as specified in the design. We can add a media query specifically for the index section to ensure that this change only applies to that section.

Finally, we can adjust the width of the bottom text in the banner to fit on two lines instead of one. To do this, we can set a max-width value to limit the width before the text wraps to the next line.

By implementing these changes, we can create a responsive website using HTML and CSS, with properly styled anchor tags and adjusted heights and widths for various elements.

To create a responsive website using HTML and CSS, there are several steps to follow. First, you need to style the text for the header. Set the display property to block and the width property to 600 pixels. Adjust the width as needed to achieve the desired layout. To center the text, set the margin property to auto.

Next, you need to style the banner section. Adjust the height and delete any unnecessary styling. For the links section, use classes to apply styling. Copy the class and styling for the box, then modify the margins as desired.

To add a wrapper around the content, create a div with the class "wrapper" and place it after the main tag. Close the wrapper tag after the footer. Style the wrapper with a width of 1000 pixels and a margin of 0 auto to center it.

Finally, adjust the width of the divs inside the links section. Copy the width property from the original divs and insert it into the square styling. Change the width to 25% to create four equal-sized squares.

Remember to remove any unnecessary styling and test the changes in the browser.

To create a responsive website using HTML and CSS, we need to make some adjustments to the styling. First, we need to overwrite the previous styling inside the div element by using the "!important" declaration. This will ensure that the new styling takes precedence over any other styles applied to the element. Next, we need to adjust the width and height of the div element to create a square shape. We can do this by changing the values in the CSS code. Additionally, we need to apply different styling to the rectangles, which should take up twice as much space as the squares. We can achieve this by modifying the width property for the rectangles. To align the elements next to each other, we can use the "float: left" property. This will position the elements side by side. However, we may notice different spacings between the elements. To fix this, we can adjust the margin property to create equal spacing between the elements. Finally, we can add some spacing between the footer and the links by setting the "overflow" property to "hidden". This will ensure that the links are pushed away from the footer. Additionally, we can modify the line height of the text inside the containers to center them vertically. Lastly, we can style the footer links and the "latest cases" title by applying appropriate CSS styling.

In this didactic material, we will learn how to create a responsive website using HTML and CSS. Responsive websites are designed to adapt and display properly on different devices and screen sizes. We will cover the steps to create a responsive front page, including adding styling, spacing, and organizing the content.

First, let's start by changing the text styling. We can make the text uppercase by adding the "text-transform: uppercase" property to the relevant element.

Next, we need to add some spacing between the menus. We can achieve this by setting a padding-right property for the unordered list. A value of 30 pixels should work well.

Now, let's remove any unnecessary styling that we don't need for our website.

Moving on, we notice that the footer is not part of the wrapper. To fix this, we need to ensure that the footer is inside the main tag, just like the wrapper. We will create a second wrapper for the footer and place it after the main tag. This will ensure that the footer is included in the wrapper.

After making these changes, we can see that the website is now responsive. When we resize the screen, the content adjusts accordingly.

Please note that due to background noise interference, the recording had to be split into two episodes. The first episode covers the creation of a mobile-responsive front page. The second episode will focus on creating two sub-pages, as we have already created the header, footer, and mobile responsiveness in the first episode.

We hope you enjoyed this project and found it helpful. Thank you for your support on YouTube, and a special thanks to our Patreon supporters. If you would like to support us or access the lesson materials, please visit our Patreon page using the link in the video description.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: RESPONSIVE WEBSITES
TOPIC: CREATING A RESPONSIVE CASES WEBSITE EXAMPLE

In the previous episode, we learned how to create a front page using HTML and CSS. In this episode, we will build additional pages for a cases page and a case page. These pages will display different cases in a portfolio, with the case page showing a specific case. Today, we will insert a video with text into the case page as an example.

Before we start coding, I would like to address a couple of questions that were asked in the comments of the previous episode. The first question was why we did not use Bootstrap. Bootstrap is an HTML framework that makes website development easier and faster, and it also provides responsive design out of the box. However, since this is an HTML course and we have not covered HTML completely yet, we decided not to use any frameworks. We may cover frameworks in the future, but for now, we will focus on building a website using core HTML and CSS.

The second question was why we did not use Flexbox in the previous episode. In the previous episode, we used float to position elements next to each other in the website. Flexbox is a newer and easier way to achieve this, but since we have not covered it in this course, we did not use it in the previous episode.

Now, let's move on to the actual website. As you can see, this is what we created in the previous episode: a header, a banner, a few links, and a footer. In this episode, we want the header to stay fixed at the top of the page even when scrolling. To achieve this, we will add some CSS code to the header section.

Inside the coding document, go to the header section and add the following CSS code: set the position property to fixed, the top property to 0, and the left property to 0. This will position the header fixed at the top left corner of the browser. If you go back to the website, you will notice that the content below the header jumps up behind it when scrolling. We need to fix this.

Scroll back to the styling section and find the content section below the header. In the HTML file, you will see that all the content is inside a main tag. Copy the main tag and paste it in the styling section above the header section. Add the CSS code: set the padding-top property to the height of the header (which is currently 100 pixels). This will push the content below the header, making it look normal again. Now, when you scroll, the header will stay fixed at the top of the page.

That's it for this episode. We have learned how to make the header stay fixed while scrolling and how to adjust the content below it. In the next episode, we will continue building the responsive cases website example.

In this lesson, we will learn about creating a responsive cases website example using HTML and CSS fundamentals. We will start by creating a cases page within our document. We will save this page as "cases.html" directly inside the main directory of our root folder.

To create the cases page, we will copy everything from the index page and paste it into the cases page. This means that the cases page will have the same content as the front page.

However, there is an important note to consider. If we make a change in the navigation menu, such as adding a new menu point like "About Us," we would need to make this change in all pages of our website. HTML alone does not provide a way to change it in one place and have it reflected everywhere. To achieve this, we would need to use other programming languages like PHP, but that is beyond the scope of this lesson.

Next, we will delete all the content inside the main tag of the cases page, except for the header and footer. After saving the changes, we can refresh the website and navigate to the cases page. Here, we will see a page with no content except for the header and footer.

To design the cases page, we will open our design and start building the required sections. We will focus on the cases header tag, eight links below it, and a contact me banner.

Inside the main tags of our coding document, we will create an h2 tag with the text "Cases" as the header.

Below the h2 tag, we will create a div box to contain the content. We could use other HTML5 tags to specify the content, but for simplicity, we will use a div in this case.

To improve code reusability, we can insert everything inside a section tag. This section tag can have a class name, such as "cases-links," to help us separate the content inside it. We don't need to include a class inside the div tag since we want to reuse the code.

By using a section as a container for the div boxes, we can avoid duplicating styling code. Instead, we only need to create one set of styles for the section and apply them to all the div boxes inside it.

We have learned how to create a responsive cases website example by creating a cases page, copying the content from the index page, and making necessary changes. We have also discussed the importance of code reusability and using appropriate HTML tags for content organization.

To create a responsive cases website example, we need to add the necessary information inside the tip box. In the design, we have the text "case one," "case two," "case three," and so on. We can create a paragraph for each case and name it accordingly, such as using a relevant name or the name of the company it was made for. After adding the information, we save it and copy/paste it to create eight cases.

To style the cases page, we need to go to the bottom of the style sheet and add a comment that says "cases" to indicate that we are styling the cases page. Inside the cases - links class, we want to target the divs inside it. We set the width to 240 pixels for each box, considering the wrapper width of 1000 pixels and the margin spaces of 5 pixels on each side. We divide the remaining width by four since there are four boxes. The height can be set to the same value or adjusted as needed. We can set the background color to a light gray shade (#iiiiii) and add a float property to position the boxes next to each other.

After refreshing the browser, we can see that all six boxes are displayed next to each other. To fix this, we need to wrap the div boxes inside a wrapper. We add a div with the class "wrapper" after the h2 tag inside the cases page and move all the tip boxes inside it. After saving the changes and refreshing the website, the boxes will be displayed correctly, with four boxes in one line and the remaining two in the next line.

Inside our CSS file, we have specified that if there are any div boxes inside the cases links, they should have a certain styling applied to them. However, we have noticed that our wrapper is also a div box inside our code, and it is also inside the cases links. As a result, all the boxes in this section are getting the unintended styling. To fix this, we will rename the cases links div to cases-link (in singular form) and add this class to all the elements inside our cases page.

We will go through each div box and assign the class "cases-link" to it. Once we do this, we can see that the styling is now applied only to the intended boxes.

Next, we want to add some spacing between these boxes. To do this, we will add a margin to all the boxes. We will set the margin to 5 pixels on all sides. After refreshing the website, we can see that the boxes now have the desired spacing.

However, we notice that the spacing underneath the boxes is smaller than the spacing between them. To fix this, we need to adjust the margin values. We will set the top margin to 10 pixels, so that it matches the bottom margin. After making this change, all the boxes have consistent spacing around them.

The last issue we need to address is that the footer is touching the links. To fix this, we will make two changes. Firstly, we will set the overflow property of the cases links container to hidden. This will ensure that all the boxes are properly contained within the container. Secondly, we will add a margin to the bottom of the container to create spacing between the links and the footer. After refreshing the website, we can see that the spacing has been added.

Additionally, we notice that there is a difference in the spacing between the boxes on the cases page and the front page. On the front page, there is more spacing between the boxes. To match the spacing on the front page, we will recalculate the margin values. We will subtract 80 pixels (20 pixels on each side of each box) from the total width of the wrapper (which is 1000 pixels) and divide the result by 4 to get the new margin value. After making this change, we can see that the spacing between the boxes on the cases page matches the spacing on the front page.

We have fixed the issue of unintended styling being applied to all the boxes in the cases links section. We have added spacing between the boxes and adjusted the margin values to ensure consistent spacing. We have also addressed the issue of the footer touching the links by modifying the overflow property and adding a margin to the bottom of the container. Finally, we have matched the spacing between the boxes on the cases page to the spacing on the front page.

To create a responsive cases website example, we need to make sure that the elements on different pages match each other in terms of spacing and layout. In order to achieve this, we will adjust the width and height of certain elements to ensure consistency.

First, we will change the width and height of a specific element to 230 pixels. After saving the changes and refreshing the browser, we can see that the element now aligns better with the rest of the content on the front page. It is important to maintain consistency across all pages of a website to ensure a cohesive design.

Next, we need to include a link around the cases on the website so that they can be clicked. We also need to adjust the text inside these cases and add a contact banner. To do this, we can modify the code by including a link tag and adjusting the styling of the text.

In the code, we can identify the cases and links section. By adding an h2 tag and copying the previous text, we can style the text accordingly. We need to uppercase the text and change the font family, font size, font weight, line height, color, and text alignment to achieve the desired look.

Additionally, we need to insert the cases text inside the wrapper to ensure proper alignment with the rest of the website. By moving the cases tag inside the wrapper, we can refresh the page and see that it now aligns correctly.

To further enhance the design, we can add some spacing from the top to create a better visual balance. This can be achieved by adjusting the CSS code.

Now that the basic layout is complete, we can focus on styling the text inside the boxes. By copying the existing code and modifying it for the paragraph element, we can change the font size and text alignment.

To vertically align the text inside the boxes, we can use a different approach than the one used in the previous episode. Instead of wrapping the content inside a table, we can add padding to the top of the text element and set it to a percentage value. This will automatically adjust the text position when the box size changes.

To add links around all the boxes, we can edit the cases page and wrap each case inside a link tag. We can also modify the link URL to point to the corresponding case page. In this example, we create separate HTML pages for each case. Although this approach requires more manual work, it is feasible for a pure HTML and CSS website.

By following these steps, we can create a responsive cases website example with clickable boxes and properly styled text. The design will be consistent across all pages, creating a visually appealing and user-friendly experience.

To create a responsive cases website example, follow these steps:

1. Create a new page and save it as "case1.html".
2. Copy the content from the "cases" page and paste it into "case1.html".
3. Delete the content inside the section tag, keeping the h2 tag.
4. Delete everything inside the main tag until the h2 tag, but keep the h2 tag.
5. Change the h2 tag to "Bella" using CSS.
6. Add an HTML5 video by creating a video tag and specifying the file type as mp4.
7. Set the video source to a video file in the video folder.
8. Set a poster image for the video by specifying the image file in the image folder.
9. Enable video controls so users can pause and play the video.
10. Apply styling to the case using CSS.

To style all case pages with video content, follow these steps:

1. Open the CSS file and add a comment for the case styling.
2. Create a CSS class selector for the case video content, e.g., ".case-vid".
3. Apply the desired styling to the ".case-vid" class selector.

Now, when you refresh the website, you will see the "Bella" title and a video player with controls for the case one page.

To create a responsive cases website example, we will first style the h2 tag. We can start by copying the h2 tag from the existing code and adding the property "text-align:center" to center the text. Additionally, we want to censor the video, so we need to set the margin to 0. After saving and refreshing the browser, we notice that the video is not yet censored. To fix this, we need to add the property "display:block" to the video tag.

Next, we want to add some spacing below the video. We can achieve this by adding a padding to the top and bottom of the video and setting the sides to 0. This will create a bit of spacing between the video and the content below it.

To maintain consistency with the text on the cases page, we will keep the name of the video the same size as the text above it. This ensures a cohesive design throughout the website.

Underneath the video, we will add some text using an article tag. The article tag is used to contain independent content that can be used and read about separately from the rest of the page. We won't assign a class to the article tag in this example.

Inside the article tag, we have columns of text. Instead of creating separate text boxes for each column, we will use CSS to create columns. This method is faster and easier, but it doesn't provide as much control over the content of each column.

To create the columns, we will use the div tag. We will assign a class to the div tag and add the desired text inside it. We can copy and paste the text into each div to create multiple columns.

To style the h3 tag inside the article tag, we can use the existing CSS code and remove the "text-align:center" property.

For the div box inside the article tag, we can copy the styling from the h3 tag and make some adjustments. We will change the font size to 16 pixels and add a line height of 24 pixels to ensure the text is not too cramped.

It's important to note that in a real-world scenario, it would be better coding practice to style the paragraph text separately in a separate CSS rule, instead of repeating the code for each paragraph.

After saving the changes and refreshing the browser, we can see the styled h2 tag, censored video, and the text in columns.

In web development, creating responsive websites is crucial to ensure a seamless user experience across different devices and screen sizes. In this example, we will walk through the process of creating a responsive cases website.

To begin, we will use HTML and CSS to structure and style the website. One important aspect of creating a responsive layout is utilizing CSS properties such as "column-count" and "column-gap". By setting the "column-count" property to the desired number of columns, we can create a multi-column layout for our content. In this case, we want three columns. Additionally, we can add a "column-gap" property to specify the spacing between the columns. For example, setting it to 20 pixels will create a 20-pixel gap between each column.

After implementing these CSS properties, we can preview the changes by refreshing the browser. If the gap appears too small, we can adjust the value accordingly. In this case, we increased it to 30 pixels.

Next, we can address the issue of text overflowing and touching the border. To prevent this, we can add padding to the tags and titles. By adding a "padding-bottom" of 20 pixels, we create spacing between the content and the border.

In the footer section, we noticed that the font size appears larger than the rest of the content. To fix this, we can modify the font size by setting it to a smaller value, such as 16 pixels. This ensures consistency throughout the website.

To further improve the design, we can adjust the line spacing between the lines of text. This can be done by changing the "line-height" property. Experimenting with different values can help achieve the desired spacing.

Now, we have successfully created a case page with text, a video, and a title that describes the case. This example demonstrates the key elements of building a responsive website. Although there are additional pages that have not been created, you can create them yourself and apply the concepts discussed in these episodes to build a complete website.

Before concluding, it is important to note that there may be a delay in the image loading on the front page. This is because the image used is large in size. To address this issue, it is recommended to resize the image using software like Photoshop to reduce its file size. By doing so, the website will load faster and provide a better user experience.

Creating responsive websites involves utilizing HTML and CSS to structure and style the content. By incorporating CSS properties such as "column-count" and "column-gap", we can create a multi-column layout with appropriate spacing. Adjusting padding, font size, and line spacing further enhances the design. Additionally, optimizing image sizes improves website loading speed. By applying these techniques, we can create visually appealing and user-friendly responsive websites.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: FURTHER ADVANCING IN HTML AND CSS
TOPIC: OUTDATED CODE IN HTML AND CSS

Depreciated Functionalities in HTML and CSS

In the field of programming, it is common for functionalities to become outdated over time and be replaced by newer alternatives. This is particularly true for HTML and CSS, where advancements are constantly being made. In this didactic material, we will explore the concept of depreciated functionalities and discuss some of the newer techniques that have emerged.

One example of a depreciated functionality is the use of "float" in HTML and CSS. In previous episodes, we learned how to create layouts by floating elements next to each other. However, this approach has its limitations. For instance, the container that holds the floated elements does not automatically adjust its height to accommodate them. To address this, a hack called "hidden overflow" was used. While this technique has been used for many years, newer methods have been introduced to overcome these complications.

One such method is CSS grid, which allows for the creation of layouts using CSS. With CSS grid, it is possible to change the positioning of elements within a website when it is viewed on different devices, such as cell phones or tablets. In the next episodes, we will delve into CSS grid and explore its capabilities in creating flexible layouts.

It is important to note that programming languages, including HTML, CSS, PHP, JavaScript, and C#, are constantly updated. To stay informed about these changes, developers often rely on resources like Google. By searching for terms like "HTML trends 2017" or "HTML trends 2018," developers can access top 10 lists or other compilations of new functionalities introduced in HTML.

However, it is essential to ensure that these new functionalities are supported by all browsers. Some browsers may have a delay in incorporating these updates. For example, when using CSS grid, it is necessary to check if different browsers fully support this functionality. Websites like "caniuse.com" provide information on browser compatibility for various features. CSS grid is now supported by most browsers, although some mobile browsers and older versions of Internet Explorer may have limited support.

Another technique that we will explore in the next episode is flexbox. Flexbox provides a more efficient way to position elements horizontally or vertically within a browser, compared to using float. It is widely supported by different browsers, making it a reliable option for creating layouts.

Programming languages like HTML and CSS are constantly evolving, with new functionalities being introduced regularly. Developers must stay updated on these changes to leverage the latest techniques and improve their programming skills.

Flexbox is a powerful tool in web development that can be used for designing various layouts. In the previous episodes, we did not delve into flexbox because it requires a dedicated lesson to fully understand its concepts. However, before we dive into flexbox, I wanted to work on a project together to reinforce our understanding of CSS and HTML.

In the upcoming episode, we will explore flexbox and its functionalities. Additionally, we will create a gallery page within the website we previously developed. This project will allow us to apply flexbox techniques to design the layout of the gallery page.

I appreciate the support you have given me on YouTube and would like to extend a special thanks to those who support me on Patreon. If you are new and unfamiliar with Patreon, you can find a link in the video description. Patreon allows you to contribute a small amount each month to support me or access the lesson materials for my channel.

I hope you enjoyed this material and I look forward to seeing you in the next lesson.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: FURTHER ADVANCING IN HTML AND CSS
TOPIC: CSS FLEXBOX

Flexbox is a new method for creating layouts in websites. In the past, we have used float in CSS to position elements next to each other. However, float was not intended for this purpose, but it worked and was used for many years. Thankfully, we now have CSS grid and flexbox as new methods for creating layouts in websites.

There are some differences between CSS grid and flexbox. CSS grid has more features for responsive design, while flexbox is more widely supported across different browsers. To check browser support, we can use the website "caniuse.com". When we search for CSS grid, we can see that there are some browsers that do not fully support it yet. However, when we search for flexbox, we can see that it is supported in more browsers.

Apart from responsiveness, there are other differences between CSS grid and flexbox. CSS grid is more focused on the layout of the entire document, while flexbox is more about rearranging items within sections of a website.

To demonstrate flexbox, let's consider an example. We have a stylesheet with some default styling that removes margins and paddings from the website. There is a class called "container" with a width of 100% and a height of 500 pixels. Inside the body tag, there is a section with the class "container". Inside this section, there are multiple div boxes with the class "item" and each div box contains a paragraph with some placeholder text.

If we view the website in the browser, we can see the text at the top and the gray background of the container section. Now, we want to use flexbox to position these elements next to each other, similar to using float.

To make the container section a flexbox, we need to add some CSS styling. We set the display property to "flex", which tells the container that it will contain items to be rearranged. Additionally, we need to specify whether the content should wrap onto multiple lines. By default, float wraps the content, but in flexbox, we need to explicitly set the wrapping behavior. We can use the flex-wrap property to do this. The default value is "nowrap", meaning the content will not wrap. We can also set it to "wrap" to allow wrapping onto the next line, or "wrap-reverse" to wrap in the opposite direction.

In our example, we want the content to wrap onto the next line, so we set flex-wrap to "wrap". Finally, we can also use the flex-direction property to specify the direction of the flex items. By default, it is set to "row", which means the items are arranged horizontally. Other possible values are "column" for vertical arrangement and "row-reverse" or "column-reverse" for reverse arrangements.

By using flexbox, we can easily create layouts with multiple items arranged next to each other, allowing for more flexibility in designing websites.

CSS Flexbox is a powerful tool in web development that allows for flexible and responsive layouts. In this lesson, we will explore the different properties and values that can be used to manipulate the layout of elements within a container.

By default, the flex-direction property is set to "row", which means that elements will be arranged from left to right in a single row. This is similar to using the "float" property in CSS. However, if we want to change the direction, we can use the "row-reverse" value to go from right to left, the "column" value to go from top to bottom, or the "column-reverse" value to go from bottom to top.

To simplify our code, we can use the "flex-flow" property to combine the flex-direction and flex-wrap properties into a single line. For example, we can set it to "row wrap" to have elements arranged in a row and wrap onto the next line when there is no more room. This can make our code more concise and easier to read.

In addition to controlling the direction of elements, we can also control the spacing between them using the "justify-content" property. This property allows us to specify how much space should be between elements horizontally. There are several values we can use, such as "center" to center the elements, "space-around" to evenly distribute space around the elements, and "space-between" to evenly distribute space between the elements.

If we want to remove the space on the edges of the container, we can use the "space-between" value. This will distribute space evenly between the elements, but not on the edges. On the other hand, if we want to distribute space evenly including the edges, we can use the "space-evenly" value.

Finally, we also have the "flex-start" and "flex-end" values for the justify-content property. "flex-end" will push the elements to the right side of the container, while "flex-start" is the default value that aligns the elements from the start.

To summarize, CSS Flexbox provides a flexible and efficient way to arrange elements within a container. By using properties like flex-direction, flex-wrap, and justify-content, we can create responsive layouts that adapt to different screen sizes and orientations.

In CSS Flexbox, there are two important properties to consider when positioning items inside a container: align-items and align-content.

The align-items property determines how the individual items are positioned horizontally within the line. The default value is flex-start, which aligns the items to the start of the line. If we change it to flex-end, the items will be aligned to the end of the line. Another option is center, which centers the items horizontally within the line.

The align-content property, on the other hand, determines how the rows of items are positioned vertically within the container. The default value is flex-start, which aligns the rows to the start of the container. If we change it to center, the rows will be centered vertically within the container. Another option is stretch, which makes the items stretch to fill the entire height of the line.

Additionally, there is a baseline value for align-items that aligns the items based on their text baseline. This is useful when some items have more vertical content than others. However, this value may not be easily visible in this example.

To style the container, we can simply add the align-items and align-content properties to the container's CSS. For example, align-items: flex-start and align-content: center will align the items to the start of the line horizontally and center the rows vertically within the container.

It's also worth noting that the align-content property affects the entire row of items, while the align-items property affects individual items.

By using the align-items and align-content properties in CSS Flexbox, we have control over how the items are positioned both horizontally and vertically within the container. This allows for flexible and responsive designs.

In CSS Flexbox, the "order" property determines the order in which items are displayed within a container. By assigning an order value to each item, we can control their positioning. For example, if all items have an order value of 1, they will be displayed in the default order specified in the HTML. However, if we assign a different order value to an item, it will be positioned accordingly, appearing before or after other items.

Another important property in Flexbox is "flex-grow". This property determines how much space an item should take up in relation to other items. By default, all items have a flex-grow value of 0, meaning they will not grow to fill available space. However, by assigning a flex-grow value of 1 or higher to an item, it will expand and take up more space. The amount of space each item takes up is proportional to its flex-grow value.

Similarly, the "flex-shrink" property controls how much an item should shrink if there is not enough space to accommodate all items at their specified widths. By default, all items have a flex-shrink value of 1, meaning they will shrink equally to fit the container. However, by assigning a lower flex-shrink value to an item, it will shrink less compared to other items, allowing it to maintain its width.

The "flex-basis" property determines the default width of an item before any flex-grow or flex-shrink calculations are applied. By default, all items have a flex-basis value of "auto", which means their width is determined by their content. However, by assigning a specific value (e.g., pixels or percentages) to flex-basis, we can control the initial width of an item.

For example, if we set flex-basis to 30% for all items and 60% for a specific item, all items will initially take up 30% of the container's width, except for the specific item, which will take up 60%. This allows us to give items specific widths and control their layout.

By combining these properties, we can create flexible and responsive layouts in CSS Flexbox. The order property determines the order of items, flex-grow controls how much space items take up, flex-shrink determines how items shrink, and flex-basis sets the default width of items.

The base styling in CSS Flexbox provides a default width for items. When the browser is resized and there is limited space available, the items can be adjusted using the properties "grow", "shrink", and "basis". By using these properties together, we can create a layout that dynamically changes the size of the content based on the browser width.

Alternatively, instead of writing "grow", "shrink", and "basis" in separate lines, we can use the shorthand property "flex". For example, "flex: 0 1 100px" is equivalent to using "grow: 0", "shrink: 1", and "basis: 100px" together.

Flexbox is a more advanced and flexible solution compared to using floats for aligning and positioning content horizontally. It offers better customization options and allows for easier management of layout changes.

In the next episode, we will explore how to create a gallery using Flexbox. This practical example will demonstrate how Flexbox can be used effectively in a real website.

If you enjoyed this lesson and would like to access additional materials, consider supporting me on Patreon. By visiting the provided link, you can access exclusive benefits and lesson materials.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: FURTHER ADVANCING IN HTML AND CSS
TOPIC: EXERCISE USING CSS FLEXBOX

In this exercise, we will learn how to create a gallery using CSS Flexbox. It's important to note that this exercise focuses on Flexbox and not on creating a functional gallery for a real-world website. In a later lesson, we will cover how to create a gallery with clickable images.

To start, we will use a website that we created in previous episodes. If you don't have access to that website, you can simply set up an HTML document and follow along.

In this exercise, we will be working with a "cases" page from the website. The page contains boxes that are not responsive, but this is what the gallery will look like. The gallery images will stretch to the width of the browser and rearrange themselves when the website is resized using Flexbox.

To begin, we will modify the HTML file by creating a wrapper div with an h2 tag inside. The h2 tag has been given a class for separate styling. We also have the option to view the HTML file and CSS file side by side in a text editor for easier navigation.

Next, we will set up a section within the main text to create the images for the gallery. The section will have a class called "gallery links". Inside the section, we will create an anchor tag, which is used to open images in a larger format. However, in this exercise, we won't focus on that functionality.

Inside the anchor tag, we will add an image. We will link to an image file called "gallery_image.jpg" located in the "images" folder. The alt tag can be used to describe the image.

This exercise demonstrates how to create a gallery using CSS Flexbox. Remember, this exercise focuses on Flexbox and not on creating a functional gallery for a real-world website.

In this exercise, we will be using CSS Flexbox to create a gallery of images. The goal is to showcase multiple images in a responsive layout.

To start, we need to create the HTML structure for our gallery. We will use anchor tags to wrap each image. Inside the anchor tag, we will add a class called "gallery-img" to style the images using CSS Flexbox. We will copy and paste the anchor tag multiple times to create a sufficient number of images for our gallery.

Next, we will apply the Flexbox properties to the section tag, which will serve as the container for our gallery. By applying Flexbox to the section tag, we can control the layout of the items inside the Flexbox. It's important to note that the styling applied to the items inside the Flexbox will only affect the direct children of the Flexbox. In this case, we will style the anchor tags using Flexbox.

Inside the gallery, we will also have a link to an h2 tag and a body with a responsive media query. The media query will allow us to change the format of the gallery when scaling down to a mobile format.

To style the gallery, we will declare it as a Flexbox by setting the display property to "flex". We will also set the flex-flow property to "row wrap", which means that the items will be displayed from left to right, and if there's no more room in one line, it will wrap to the next line. This will give us multiple lines inside our gallery.

To add spacing between the images, we will set a margin of 20 pixels to the gallery-img class. However, after reviewing the spacing, we decided to reduce it to 10 pixels for a better look.

To ensure that the images scale properly, we will set the width of the images to 100% inside the gallery-img class. This will make the images scale together with the Flexbox.

Lastly, we noticed that there was more spacing on the left side of the gallery. To fix this, we will set the justify-content property of the container to "center". This will align the content in the center of the website.

By following these steps, we have successfully created a gallery using CSS Flexbox. The gallery is responsive and scales with the website. The images are displayed in a grid-like layout with proper spacing.

When working with CSS Flexbox, it is possible to create a gallery layout that adjusts its appearance based on the screen size. By using media queries, we can customize the layout for different devices.

To start, let's assume we have a gallery image that we want to display as a full-width image on mobile devices. Inside the media query for mobile, we can copy the gallery image and change the Flex basis property to a percentage value, such as 90%. We don't need to adjust the margin in this case. After making these changes, we can refresh the browser and observe the updated layout when viewed on a mobile device.

However, if we want the image to return to its regular size when viewed on a desktop or tablet, we need to modify the layout again. We can place the original gallery image code outside the media query, so it applies to all screen sizes except for mobile. By doing this, the image will revert to its original size when viewed on larger screens.

It's worth noting that in this tutorial, the speaker mentions linking the gallery images to a separate page. By removing the "#" symbol from the links, the images will open in a new page when clicked. To exit the image, users will need to click the back button.

This tutorial demonstrates how to use CSS Flexbox to create a responsive gallery layout. By using media queries, we can customize the layout for different screen sizes, ensuring that the images display optimally on various devices.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: FURTHER ADVANCING IN HTML AND CSS
TOPIC: FILE PATHS IN HTML AND CSS

File paths in HTML and CSS are essential for linking files and navigating through directories. There are two types of links: internal links and URL paths.

Internal links are used to link to files within the same website. For example, if there is a folder named "folder" and inside it, there is a file called "gallery.html", we can link to it by specifying the path as "folder/gallery.html". This allows us to navigate to the desired file within the website.

On the other hand, URL paths are used to link to external websites. By including the URL of the website, we can create a link that directs users to that specific webpage. This is done by specifying the full URL, such as linking to a YouTube channel.

To navigate back a directory or folder, we use ".." followed by a forward slash. For example, if we are inside a folder called "folder" and need to go back a directory and then into the "includes" folder, we can use the path "../includes".

There is also an option to go back to the main directory of the website by using a forward slash at the beginning of the path. However, this is not recommended as it can cause issues with linking within the website, especially if the structure of the website changes.

File paths in HTML and CSS are used to link files and navigate through directories. Internal links are used to link to files within the same website, while URL paths are used to link to external websites. It is important to use the correct path and avoid using the forward slash at the beginning of the path unless necessary.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: FURTHER ADVANCING IN HTML AND CSS
TOPIC: FORMS IN HTML AND CSS

Forms in HTML and CSS are essential for creating interactive websites. While HTML allows us to create the visual components of a form, we need another programming language to handle the data submitted through the form. Forms can be used for various purposes such as contact forms or login systems.

To illustrate this, let's consider a website with a contact form at the bottom. Users can enter their name, email address, and message, and then click on the "Send Email" button. Another example is a login system at the top of websites like YouTube or Facebook, where users enter their username and password and click on the "Login" button.

HTML and CSS enable us to create the visual elements of the form, such as input fields and buttons. However, we cannot process the data entered into the form using only HTML and CSS. To handle the data, we need to use programming languages like PHP, JavaScript, or Python.

You might wonder why you should learn how to create forms in HTML and CSS if you eventually need to learn another programming language. There are a few reasons for this. Firstly, it is easy to find tutorials and guides on creating HTML contact forms or login systems. These resources provide step-by-step instructions, making it accessible even for beginners. You can simply copy and paste the code into your website and get it working. This way, you can still utilize forms in HTML without knowing other programming languages.

Secondly, when you progress to languages like PHP or JavaScript, you will often encounter tasks related to handling input, which involves working with HTML forms. Therefore, it is beneficial to learn how to create forms in HTML and CSS early on. By doing so, you avoid handicapping yourself when learning these languages.

Now, let's dive into creating a form. In a basic HTML file, you can create a form by using the opening and closing form tags. Inside the form tags, you can insert various input fields that you need for your form. Here are some examples of commonly used input fields:

- Text Input: This is a basic input field for users to enter text. It is defined using the `<input>` tag with the `type="text"` attribute.
- Email Input: This input field is specifically designed for email addresses. It is defined using the `<input>` tag with the `type="email"` attribute.
- Password Input: This input field is used for secure password entry. It is defined using the `<input>` tag with the `type="password"` attribute.
- Checkbox: This allows users to select multiple options. It is defined using the `<input>` tag with the `type="checkbox"` attribute.
- Radio Button: This allows users to select a single option from a list. It is defined using the `<input>` tag with the `type="radio"` attribute.

These are just a few examples, and there are many more input types available in HTML.

In addition to the type attribute, input fields also require a name attribute. This attribute is used to identify the input field when handling the form data. You can also use optional attributes like value or placeholder to provide default values or hints for users.

By understanding how to create forms in HTML and CSS, you can enhance the interactivity and functionality of your websites. It is a crucial skill to have, even if you plan to learn other programming languages.

Forms in HTML and CSS are used to collect user input and submit it to a server for further processing. In this section, we will discuss the different types of form inputs and how to use them effectively.

One commonly used feature in forms is the placeholder text. This is the faded text that appears inside an input field, providing a hint to the user about what information should be entered. The placeholder text is not editable and serves as a background for the input field. To add a placeholder text, we can use the "placeholder" attribute in the HTML code. For example, we can use "email" as a placeholder for an email input field.

Another way to provide default text in an input field is by using the "value" attribute. Unlike the placeholder text, the value attribute sets the actual text that appears inside the input field. This text can be edited by the user. By setting a default value, we can pre-fill the input field with specific information. To add a default value, we can use the "value" attribute in the HTML code.

In addition to text input fields, we can also use radio buttons in forms. Radio buttons allow users to select only one option from a group of choices. Each radio button is represented by a small circle that can be selected or deselected. To create a group of radio buttons, we need to use the same "name" attribute for each button. This ensures that only one option can be selected from the group. By setting the "value" attribute for each radio button, we can assign a specific value to each option.

To demonstrate the use of radio buttons, let's consider an example of a gender selection. We can create three radio buttons with the options "male," "female," and "other." By assigning the same "name" attribute to each button, we ensure that only one option can be selected at a time. Additionally, we can use the "checked" attribute to pre-select a default option.

Finally, we have the submit button. This button is used to submit the form data to a server for processing. To create a submit button, we use the "type" attribute with the value "submit." The "name" attribute is optional for the submit button, but it is recommended to include it for better error handling in the server-side code.

Forms in HTML and CSS allow users to input data and submit it for further processing. We can use placeholder text or default values to guide users in filling out the form. Radio buttons are useful for selecting one option from a group, and the submit button is used to send the form data to a server.

In HTML and CSS, forms are an essential component for collecting user input on a webpage. In this section, we will explore how to create forms and handle the data submitted by users.

To create a form, we use the `<form>` tag. Inside this tag, we can specify various attributes to define the behavior of the form. One important attribute is the `action` attribute, which determines where the form data will be sent once the user clicks the submit button. For example, we can set the `action` attribute to a PHP file called "contact_form.php".

Another attribute we can set is the `method` attribute, which determines how the data will be sent to the server. There are two options: `get` and `post`. The `get` method appends the form data to the URL, while the `post` method sends the data in the body of the HTTP request. It is generally recommended to use the `post` method when dealing with sensitive data like passwords, as the data will not be visible in the URL.

It's important to note that the `get` method has a limit of 3000 characters in the URL, while the `post` method has no such limit. In our example, since we don't have any sensitive data, we can choose the `get` method.

To illustrate the use of a password input field, we can add an `<input>` tag with the `type` attribute set to "password". We can also set the `name` attribute to "PWD" and provide a placeholder text like "Password". This ensures that the user's input is hidden behind dots or asterisks to protect their privacy.

When a user submits the form, the data is sent to the server. If we use the `get` method, the form data will be visible in the URL, which is not ideal for sensitive information. However, if we use the `post` method, the data will not be visible in the URL, providing better security.

In addition to text input fields and passwords, we can also include a `<textarea>` tag to allow users to enter longer messages or comments. The `<textarea>` tag has an opening and closing tag, and we can set a `name` attribute to identify the data. For example, we can set the `name` attribute to "message". We can also provide a placeholder or initial text inside the `<textarea>` tag, such as "Write a message".

By understanding how to create forms and handle form data in HTML and CSS, we can build interactive webpages that allow users to submit information and interact with our websites.

In HTML and CSS, forms are an essential part of creating interactive websites. Forms allow users to input and submit data, making them a crucial component in web development. In this didactic material, we will explore the concepts and techniques involved in creating forms using HTML and CSS.

One of the elements commonly used in forms is the text area. A text area provides a larger space for users to input text compared to a regular input field. To create a text area, we use the `<textarea>` tag. Within this tag, we can set attributes such as the name and size of the text area. By default, the text area can be resized by the user. However, if you find this feature to be unnecessary, you can use CSS to limit the resize button. This can be achieved by setting the `resize` property of the text area to "none". This ensures that the text area remains a fixed size, providing a better user experience.

Another aspect we can modify in forms is the color of the text inside the input fields. By default, the placeholder text appears in a gray color. However, you may want to customize this color to match the design of your website. Using CSS, you can target the input fields and change the color of the placeholder text to your desired value.

Additionally, we can alter the appearance of the input fields themselves. By default, the outline of an input field is displayed in blue when it is selected. If you prefer a different color, you can use CSS to change the colored outline of the input fields. This allows you to customize the visual style of your forms, making them more visually appealing and cohesive with the overall design of your website.

Moving on to another form element, we have the select dropdown menu. This element allows users to choose from a list of options. To create a select dropdown menu, we use the `<select>` tag. Within this tag, we define a name for the select dropdown menu and provide the available options using the `<option>` tag. Each option can have a value associated with it, which can be used for further processing on the server-side. By default, the first option is selected. However, you can set a default option by specifying a value for the "selected" attribute.

Lastly, let's discuss the submit button. In the previous examples, we used the `<input>` tag with the type attribute set to "submit" to create a submit button. However, HTML also provides a `<button>` tag specifically designed for buttons. To create a submit button using the button tag, we set the type attribute to "submit" and specify a name for the button. Inside the button tag, we can add text or other elements to customize the appearance of the button.

To style the form elements, we can use CSS. By linking an external stylesheet to our HTML document, we can define specific styles for the form and its elements. In the stylesheet, we can set properties such as width and height to control the dimensions of the form. We can also modify the width of the input fields to make them span the entire width of the form. This ensures that the form elements are visually consistent and aligned properly.

Forms are an integral part of web development, allowing users to interact with websites and submit data. By utilizing HTML and CSS, we can create and customize various form elements such as text areas, select dropdown menus, and submit buttons. With the ability to style these elements using CSS, we can ensure that our forms are visually appealing and consistent with the overall design of our websites.

To further advance in HTML and CSS, we can explore forms in HTML and CSS. Forms are used to collect user input and are an essential part of web development. In this section, we will learn how to resize form elements, remove default styles, and customize the appearance of form inputs.

First, let's discuss resizing form elements. By default, form elements can be resized in both directions. However, if we want to restrict the resizing to only vertical or horizontal, we can use CSS. For example, setting the "resize" property to "vertical" allows resizing only in the vertical direction. On the other hand, setting it to "both" enables resizing in both directions. Additionally, we can set a maximum height or width using the "max-height" or "max-width" property. This prevents the form element from being infinitely resized. For instance, setting "max-height" to a value like 300 pixels limits the vertical resizing.

Next, let's address the default styles applied to form inputs, particularly the blue outline. In some cases, we may want to change or remove this outline. To do so, we can target the input and textarea elements in our CSS stylesheet. By using the "outline" property and setting it to "none", we can remove the default outline. This provides a cleaner and more customized appearance for our form inputs.

Furthermore, we can customize the border color of form inputs. Instead of relying on the default color, we can define our own using the "border" property. By setting the "border" property to "1 pixel solid" followed by a color value, such as "#DDD" for a light gray color, we can achieve the desired look.

To enhance the user experience, we can also change the appearance of form inputs when they are focused. By using the ":focus" pseudo-class, we can apply different styles to the input and textarea elements when they are clicked or focused. For example, we can change the border color or add a box shadow effect. In our example, we set a red box shadow with a slight transparency when the inputs are focused.

Finally, we can modify the color of the placeholder text inside the form inputs. This is an optional customization. To style the placeholder text, we can use the "::placeholder" pseudo-element. By applying CSS properties within the curly brackets, we can change the color and opacity of the placeholder text. For instance, setting the color to red and opacity to 1 will make the placeholder text appear in red.

By understanding and implementing these techniques, we can create visually appealing and user-friendly forms in HTML and CSS.

Different web browsers may render HTML and CSS code differently, which can be frustrating when trying to achieve consistent styling across all browsers. In this material, we will discuss how to style elements for specific browsers using HTML and CSS.

To begin, let's consider the issue of opacity. In order to make an element look the same in Firefox as it does in other browsers, we need to add a specific styling. To do this, we can use the following code:

```
::-moz-placeholder {
  opacity: 0.5;
}
```

This code targets the placeholder text in Firefox and applies the desired opacity. However, if we want to apply the same styling to Internet Explorer or Microsoft Edge, we need to add some additional code. We can use the following code for Internet Explorer:

```
::-ms-input-placeholder {
  opacity: 0.5;
}
```

And for Microsoft Edge, we can use:

```
::-ms-input-placeholder {
  opacity: 0.5;
}
```

It's important to note that in the future, there may be support for styling Microsoft Edge and Internet Explorer without the need for these additional codes. However, as of now, it is necessary to include them.

Once we have added these styles, we can save the code and refresh the browser to see the changes. The placeholder text will now have the desired opacity in all supported browsers.

Additionally, it's worth mentioning that the positioning of elements can also vary between browsers. For example, in the provided code, the "mail" input and the adjacent button are not displayed next to each other. This is because they are placed on separate lines by default. To place them side by side, we can use flexbox or other positioning techniques.

If you are unfamiliar with flexbox, there is a tutorial available that explains how to use it to position elements. You can refer to that tutorial for a more detailed explanation.

To conclude, this material has covered the topic of styling elements for different browsers using HTML and CSS. By using specific codes, we can achieve consistent styling across various browsers. If you are interested in creating forms or other interactive elements using HTML and CSS, there are tutorials available in the description of this material that you can refer to.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: FURTHER ADVANCING IN HTML AND CSS
TOPIC: TABLES IN HTML AND CSS

Tables are an important element in HTML and CSS for displaying data in a structured manner. In this material, we will learn how to create tables using HTML and CSS.

To begin, open an index page and ensure that there is nothing inside the body tags. Also, open a stylesheet that is already linked to the index page, as we will be applying some styling to the tables.

Within the body tag, create a table using the table tag. Inside the table, we work with rows and columns. By default, the table will not have any borders. However, we can add borders using CSS to visualize the structure.

To create a row, use the TR tag, which stands for table row. The first row of a table is usually reserved for column names, which can be created using table headers (TH). For example, you can have column names like "First Name," "Last Name," and "Gender."

After creating the header row, you can create additional rows by copying the structure of the first row. Inside these rows, change the table header tags to table data (TD) tags. Fill in the data for each column, such as first name, last name, and gender.

Once the table structure is in place, you can refresh the browser to see the table. If the table appears small, you can set a width for the table in the stylesheet, such as 400 pixels.

By default, the table header text is centered, while the table data is aligned to the left. You can change the alignment by modifying the text-align property in the stylesheet.

To add borders to the table, create a separate styling element in the stylesheet for the table, table header, and table data. Set the border property to "1 pixel solid" to create a black border.

After refreshing the browser, you will see that borders are now visible within the cells of the table. If there is unwanted spacing between the borders, you can adjust it by setting the border-spacing property in the stylesheet.

Tables in HTML and CSS provide a structured way to display data. By understanding the tags and properties associated with tables, you can create visually appealing and organized tables for your web pages.

Tables in HTML and CSS are a powerful tool for organizing and presenting data on a webpage. In this lesson, we will explore some advanced techniques for working with tables.

One important aspect of table design is controlling the spacing between cells. By using the CSS property "border-spacing", we can specify the amount of space between cells. For example, setting "border-spacing: 40px;" will create a larger gap between cells. However, if we want to remove the spacing altogether, we can use the "border-collapse" property. By setting "border-collapse: collapse;", the borders between cells will merge into a single border.

To further customize the appearance of our table, we can create separate styles for table headers and table data cells. By removing the table selector from the style rule, we can apply the styles specifically to the headers and data cells. For example, we can set a padding of 10 pixels to create some space between the text and the border of the cells.

In addition to styling the cells, we can also add a caption to our table. The caption tag is used to provide a title or label for the table. By creating a caption element and styling it using CSS, we can position it and align it as desired. For example, setting "text-align: left;" will align the caption to the left side of the table.

Finally, we can use CSS selectors to apply different styles to alternate rows of a table. By using the "nth-child" selector, we can target every even or odd row and apply different background colors. For example, setting "background-color: white;" for even rows and "background-color: light gray;" for odd rows will create a visually appealing alternating pattern.

It is worth noting that the "nth-child" selector can be further customized to target specific patterns, such as every third or fourth row. This provides a lot of flexibility in styling tables.

Tables in HTML and CSS offer a wide range of customization options. By controlling spacing, styling individual cells, adding captions, and applying different styles to alternate rows, we can create visually appealing and well-organized tables on our webpages.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: OVERVIEW OF HTML AND CSS EXTENDING SKILLS

Welcome to this overview of HTML and CSS extending skills. In this lesson, we will discuss the importance of continuing your learning journey in web development beyond the basics of HTML and CSS.

It is crucial to understand that there is much more to learn when it comes to HTML and CSS. Uploading a website to the internet is an essential skill, but it does not mark the end of your learning journey. As a web developer, it is important to ask yourself what your goals are and what you want to achieve in this field. Do you want to create a portfolio website for yourself, work professionally as a freelance web developer, or be a part of a company's web development team? Your aspirations will determine the extent of your learning.

To give you an overview of the skills you need to develop if you want to become a professional web developer, let's explore three main areas of specialization: design, front-end development, and back-end development.

Design is the process of creating the visual aspect of a website. Designers ensure that the website is visually appealing, user-friendly, and has a positive user experience. They create mock-ups and test them with users before the actual coding process begins.

Front-end developers, on the other hand, take the design created by the designers and bring it to life in the browser. They use HTML, CSS, and JavaScript to make the website interactive and visually appealing. They are responsible for ensuring that the website functions as intended and provides a seamless user experience.

Back-end developers focus on the behind-the-scenes functionality of a website. They take the front-end code and add dynamic features, such as login systems, database interactions, and server management. Back-end developers handle the technical aspects that make a website work smoothly.

In larger companies, web developers often specialize in one area, allowing them to become experts in their chosen field. However, in smaller companies, you may find yourself working on multiple aspects of web development.

It is important to note that as you progress in your web development journey, there will be more practical examples and tutorials to help you further enhance your skills. These tutorials will focus on the practical aspects of creating websites and will provide you with valuable hands-on experience.

Remember, learning HTML and CSS is just the beginning. There is a vast world of web development waiting for you to explore. So, keep learning, practicing, and honing your skills to become a proficient web developer.

Web development encompasses various aspects, including front-end development, back-end development, and design. Depending on the size of the company or if you plan to work as a freelance developer, it may be beneficial to have knowledge in all three areas. However, it is important to understand your limitations and communicate them to clients.

After learning HTML and CSS, it is recommended to expand your skills by learning JavaScript. Even if you don't specialize in front-end development, having a basic understanding of JavaScript is valuable in web development. The three core languages in front-end development are HTML, CSS, and JavaScript.

If you are interested in back-end development, it is advised to learn PHP after gaining proficiency in HTML, CSS, and JavaScript. While there are many programming languages available, PHP is commonly used in back-end development.

To determine the specific skills required for a web development job, it is recommended to visit job sites and review the job descriptions for positions you are interested in. This will give you an understanding of the skills and requirements needed to secure a job interview.

If you are still unsure about the different areas of web development, there are resources available, such as an episode on a web development channel, that explain the various aspects of web development, including front-end, back-end, CMS systems, optimization, and frameworks.

It is important to stay motivated and inspired while learning web development. Starting with HTML and CSS is commendable, as it forms the foundation for further learning. Pat yourself on the back for the progress you have made so far. Web development can be challenging, especially when starting from scratch, but with determination and continuous learning, you can achieve your goals.

In the upcoming lessons, we will discuss how to upload a website on the Internet. We will cover the topic of meta tags and other essential elements that need to be included in a website before it is released. These elements are not overly complex but are necessary for a functioning website. We will go through the process step-by-step, allowing you to understand what is required before uploading a website. I hope you found this material informative and I look forward to our next session.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: REQUIRED HTML META TAGS

Meta tags are an important aspect of HTML that can provide additional information about a website. In this didactic material, we will discuss the significance of meta tags and their role in improving search engine optimization (SEO). It is essential to note that not all meta tags contribute to SEO, and including unnecessary meta tags can be counterproductive.

One type of meta tag that can enhance SEO is the charset meta tag. This tag specifies the character encoding used on the website. It is recommended to include the charset meta tag with the value "utf-8" in all websites.

Another crucial meta tag is the title tag. Although not strictly a meta tag, it is important to have it within the head tag of a webpage. The title tag defines the title of the webpage and should be concise and descriptive.

The description meta tag is another significant meta tag for SEO. It provides a brief description of the webpage's content. It is important to note that the content of the description meta tag should not exceed 160 characters. To optimize SEO, it is advisable to use keywords that are commonly searched for and avoid highly competitive words.

The viewport meta tag is essential for creating responsive websites. It ensures that media queries in the CSS file are correctly interpreted by the browser. Including the viewport meta tag with the value "width=device-width, initial-scale=1" is crucial for enabling responsiveness.

These are the required meta tags that should be included in all websites. However, there are also optional meta tags that can be used. It is important to carefully consider the relevance and necessity of each meta tag before including it in a website.

Meta tags are an important aspect of website development. They provide information about the website to search engines and other machines. While some meta tags are essential, others are optional and may not have a significant impact.

One commonly used meta tag is the "keywords" tag. This tag is used to specify the keywords that describe the content of the website. However, it is important to note that search engines like Google no longer consider this tag as relevant due to past spamming practices. Nevertheless, there may still be some machines or devices that consider keywords as part of the meta tag description. In our case, it is not necessary to include this tag unless you want to target specific machines.

Another optional meta tag is the "author" tag, which indicates the person or entity responsible for creating the website. This tag does not have a significant impact on search engine optimization or other aspects, but it can provide additional information about the website.

It is crucial to include necessary meta tags before uploading a website to the internet. However, there are many other meta tags that can be used depending on specific requirements. To explore more options, you can refer to the provided link in the description. Additionally, a link to a search engine optimization chart is also available, which can help improve the visibility of your website on search engines.

In the next episode, we will discuss some good conventions and rules that should be followed when coding websites. Following that, we will cover the process of uploading a website to the internet. Stay tuned for more informative content.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: IMPROVING HTML AND CSS CODE

This part of the material is currently undergoing an update and will be republished shortly.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: UPLOADING A WEBSITE

To upload a website to the internet, there are a few things that you need to have in place. First and foremost, you need a website. It can be your own website or any other website that you want to upload. However, you must have an index.html page for the website to work. If you don't have a complete website, you can quickly set up an index page and add some text to it.

Next, it's important to note that while making a website is free, uploading a website to the internet does come with a small fee. To upload your website, you will need a domain and a server. A domain is the name of your website, such as example.com. The server is the place where you upload your website and it is what makes your website accessible on the internet.

To get a domain and a server, you will need to purchase them. The domain is a one-time purchase, while the server requires a monthly payment. The cost of the server is relatively low and varies based on the duration of your hosting plan. It's important to mention that there are hosting companies that offer free or cheap domain and server services, but it's recommended to choose a reputable and reliable hosting company.

One such hosting company is Hostgator, which is known for its popularity and affordability. Despite its cartoony appearance, Hostgator is widely used around the world. You can visit their website at hostgator.com or a similar hosting provider. Once you are on their website, you can navigate to the web hosting section to explore different plans. For a small business website, the cheapest plan is usually sufficient. This plan typically includes a single domain, 1-click installs, and a certain amount of bandwidth.

When purchasing a domain, you can choose a name that suits your website. For example, you can select a domain like example.com. The hosting company's website will inform you if the domain is available or not. If it is available, you can proceed with the purchase.

Once you have a domain and a server, you can start the process of uploading your website. However, it's important to note that the specific steps for uploading a website may vary depending on the hosting company you choose. If you are using Hostgator, you can follow their instructions and guidelines for uploading your website.

To upload a website to the internet, you need a website with an index.html page, a domain, and a server. The domain is the name of your website, and the server is where you upload your website. While there is a small fee associated with uploading a website, there are affordable hosting companies like Hostgator that offer domain and server services. Once you have a domain and a server, you can follow the specific instructions provided by your hosting company to upload your website.

When uploading a website, there are several options and settings to consider. One option is purchasing multiple domain extensions to prevent others from using a similar name. However, this can be expensive for small individuals or businesses. Choosing a domain with a secure server is important to protect against hackers. Domain privacy protection is a setting that hides personal information from public access. It is up to the individual to decide if they want to enable this feature and pay extra for it.

There are different options to choose from when purchasing a domain. The hat sling packets type is the cheapest option. The billing cycle determines how many months you pay for upfront, with longer cycles resulting in cheaper prices. If the domain is only needed for a specific tutorial and won't be used afterwards, it is possible to purchase just one month.

During the purchasing process, personal information needs to be provided. It is important to keep this information private and not share it. After entering the billing information, additional services can be chosen. These services are optional but can enhance the security of the website. For example, SiteLock monitoring protects against hackers, and it is highly recommended to use this service for personal or business websites. Another service is backup, which automatically creates a backup of the website in case of any changes or issues. This is useful for websites with dynamic content or products.

There is also a coupon code that can be applied to reduce the overall cost. The final price will depend on the selected services and any discounts applied.

When uploading a website, it is important to consider purchasing multiple domain extensions, choosing a secure server, and deciding on additional services to enhance website security. The cost will vary depending on the selected options.

After purchasing the domain and server, there is a software called FileZilla that can be used to upload the website to the server. This software makes it easier to switch out files from the website in the future. Another option is Cyberduck, which is also popular. Links to download both FileZilla and Cyberduck can be found in the video description. It is important to download the file transfer protocol client, as it allows for the transfer of files to the online service and the website's domain. Once the software is installed, the website can be uploaded to the internet.

Before uploading the website, there are a few things to check. First, proofread the website to ensure there are no spelling mistakes or broken links. Once the website is live, visitors will be able to point out any errors. Next, make sure the website has the necessary meta tags inside the head tag. This was covered in a previous episode. Additionally, test the website in different browsers such as Internet Explorer, Chrome, Firefox, and Opera to ensure it works properly. Each browser may display the website slightly differently. Finally, validate the website using an online validator such as validator.w3.org. This will check if the website is marked up correctly. The website can be uploaded directly or by using the file upload option on the validator website.

During the validation process, it is possible to encounter errors. For example, there may be an error related to using a long link to fonts from Google's font website. In such cases, it is important to evaluate whether the error is significant or if it can be ignored.

To ensure the smooth functioning of a website, it is crucial to validate all pages within the website for errors. This can be done by using a link provided in the description of this material. Additionally, it is important to keep a copy of the website offline before uploading it to the internet. This ensures that there is a backup in case any issues arise with the online domain and the website needs to be re-uploaded.

To upload a website, certain information is required. In this case, we will be using the FileZilla program to access the server. The necessary information can be found in the hosting provider's dashboard. In this example, the hosting provider used is Hostgator. After logging into the Hostgator dashboard, navigate to the domains section to select the domain that was purchased. Once the domain is selected, check for any pending payments or notifications. It may take some time for the payment to be processed.

While waiting for the payment to be processed, it is important to check the email inbox for any correspondence from the hosting provider. In this case, two emails were received. The first email is a billing confirmation for the domain and server purchase. The second email contains all the information needed to log into the Hostgator dashboard. It is crucial not to discard this email. If no email is received, it is recommended to contact the hosting provider and inform them about the issue. Sometimes, it may take some time before the email is received, so it is advised to wait for about half an hour to an hour before contacting the hosting provider.

After receiving the necessary email, follow the instructions to verify the email address. Once the email address is verified, return to the Hostgator dashboard to confirm that the domain is working. At this point, it is possible to visit the domain and see the default files provided by the hosting provider.

To upload the website files, it is necessary to access the server and delete the existing files. In the Hostgator dashboard, navigate to the hosting tab and select the files and folders option. Here, there are two options: open the file manager or use an FTP program like FileZilla. In this example, FileZilla will be used.

To use FileZilla, return to the Hostgator dashboard and go to the FTP accounts section. Create a new FTP account by providing the necessary information, such as the account name and password. Once the account is created, the required information to access the website using FileZilla will be displayed.

Open FileZilla and enter the provided information to connect to the server. Once connected, navigate to the public HTML folder, which contains the default files. Delete these files and replace them with the website files that need to be uploaded.

By following these steps, the website can be successfully uploaded and accessed online.

To upload a website, you will need to follow a few steps. First, open your FTP client and go to the file site manager. Click on "New Site" and give it a name. In the host section, enter the server name provided by your hosting provider. The port is usually 21 for FTP. Choose the logon type as "normal" and enter your username and password. Click on "Connect" to establish a connection with the server.

Once connected, you will see a list of files and folders on the server. Look for the folder called "public_html" or something similar. This is where you will upload your website files. Open this folder.

Now, on your computer, locate the folder that contains all the files of your website. Select all the files and folders, and drag them into the FTP client window where you opened the "public_html" folder. The files will start uploading one by one.

You can monitor the progress of the upload in the bottom section of the FTP client. It will show you the number of files remaining to be uploaded and the status of each transfer. Once all the files have been uploaded successfully, you can go to your website by entering its URL in a web browser. You will now see your website online.

If you need to make changes to your website in the future, you can simply drag and drop the updated files into the "public_html" folder, replacing the existing files.

Remember that you can save the connection details in the site manager, so you don't have to enter them every time you want to upload or make changes to your website.

In case you encounter any issues with your hosting provider, you can use the live chat function on their website to get assistance from their support team.

That's it! You have now successfully uploaded your website to the internet. Keep in mind that this process may vary slightly depending on your hosting provider and FTP client.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: VALIDATING A WEBSITE

Validating a website is an important step in the web development process. It ensures that your website adheres to the standards set by the W3C (World Wide Web Consortium) and helps identify any errors or issues in your HTML and CSS code.

When you upload a website to the internet, it is crucial to validate it before making it live. Validating a website involves checking the correctness and compliance of the HTML markup and CSS styling used in the website.

To validate a website, you can use the validation tool provided by the W3C. This tool allows you to test your website for HTML and CSS errors. You can access the tool by visiting "validator.w3.org".

There are three ways to test your website using the W3C validator. The first method is to enter the URL of your website if it is already online. However, if your website is not yet live, you can choose the second method, which involves uploading the HTML file of your website to the validator. The third method allows you to directly input your HTML code into the validator.

Once you have chosen the appropriate method, the validator will analyze your HTML and CSS code and provide a detailed report of any errors or potential issues found. The report will include a list of validation errors, which indicate the specific areas in your code that need to be fixed.

It is important to note that even if your website appears to be error-free when viewed in a browser, there may still be underlying issues that are not visible. Browsers often try to fix errors in the code to display the website correctly. However, this can lead to a false sense of correctness. Validating your website with the W3C tool helps ensure that your code is truly error-free and compliant with web standards.

By validating your website, you can identify and fix any errors or issues in your HTML and CSS code, ensuring that your website functions correctly across different browsers and devices. It also helps improve the accessibility, usability, and search engine optimization (SEO) of your website.

Validating a website is a crucial step in the web development process. It helps ensure that your website adheres to web standards, identifies and fixes errors in your HTML and CSS code, and improves the overall quality and performance of your website.

Validating a website is an important step in the web development process to ensure that the code is error-free and adheres to the standards set by the World Wide Web Consortium (W3C). In this didactic material, we will discuss how to validate a website's HTML and CSS code using the W3C Markup Validation Service.

When validating HTML code, the first step is to check if a document type declaration (DOCTYPE) is present. The DOCTYPE specifies the version of HTML being used and helps browsers interpret the code correctly. If a DOCTYPE is missing, it can lead to errors. By adding a DOCTYPE declaration at the beginning of the HTML code, such as <!DOCTYPE html>, we inform the validator that the document is using HTML5.

Next, we can upload the HTML code to the W3C Markup Validation Service. Upon validation, the service will highlight any errors or warnings in the code. By addressing these errors, we can ensure that the website follows the correct syntax and structure.

One common error that may occur is the absence of text between the title tags. The title tags define the title of the webpage that appears in the browser's title bar or tab. To resolve this error, we can simply add relevant text between the opening and closing title tags.

Another error that may be flagged is the presence of a bad URL that links to fonts used within the website. This can happen when using Google Fonts, as the link provided by Google contains a pipe symbol that is not recognized by the validator. To overcome this issue, we can manually remove the pipe symbol and use URL encoding. URL encoding allows us to represent special characters, such as the pipe symbol, with a specific code. In the case of the pipe symbol, we can use "%7c" as the encoded representation.

After addressing these errors, we may encounter an error related to unclosed elements before the body tag. This error indicates that there are elements in the code that have not been properly closed. By examining the code, we can identify the unclosed element and add the necessary closing tag to resolve the error.

It is important to note that unclosed elements are a common mistake made by developers. While they may remember to open tags, they often forget to close them. Therefore, it is crucial to double-check the code and ensure that all elements are properly closed.

Once we have fixed all the errors in the HTML code, we can re-upload it to the validator and verify that there are no remaining issues. The W3C Markup Validation Service will provide a report indicating that the HTML code is valid.

It is also worth mentioning that CSS code needs to be validated separately using the W3C CSS Validation Service. The CSS validator can be accessed through the Jigsaw CSS Validator website. By uploading the CSS code to this service, we can ensure that it follows the correct syntax and adheres to CSS standards.

Validating a website's HTML and CSS code is essential to ensure its correctness and compliance with industry standards. By using the W3C Markup Validation Service for HTML and the W3C CSS Validation Service for CSS, we can identify and fix any errors or warnings, resulting in a well-structured and error-free website.

The process of validating a website is crucial in web development to ensure that the website's markup and styling are error-free. The HTML validator is commonly used to check for errors in the HTML code, while the CSS validator is used to validate the CSS styling.

To validate a website, there are several methods that can be used. One option is to provide a link to the website, allowing the validator to analyze the markup and styling remotely. Another option is to upload the HTML and CSS files directly to the validator. Lastly, the validator also allows for direct input of the code.

In the case of CSS validation, if an error is found, the validator provides an error message indicating the line number where the error occurred. By locating the specified line in the CSS file, the error can be identified and addressed. For example, a missing curly bracket in a CSS styling block can cause a parse error.

After making the necessary corrections to the code, the files can be re-uploaded or re-entered into the validator for re-validation. It is important to ensure that no errors are found in the markup and styling before uploading the website to the internet. Not only can errors affect the functionality of the website, but they can also impact the website's search ranking on search engines like Google.

Validating a website is especially important for students studying web development. Teachers often require students to validate their websites before submitting them for exams or projects. Failure to validate a website can result in penalties and may affect the overall assessment of the project.

Validating a website is an essential step in web development. It helps identify and fix errors in the markup and styling, ensuring a smooth and error-free user experience. Whether you are a student or a professional web developer, it is crucial to validate your website before publishing it.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: CREATING AN XML SITEMAP

This part of the material is currently undergoing an update and will be republished shortly.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: CREATING A 404 PAGE IN HTML

A 404 page is a page that is displayed when a user tries to access a page on a website that does not exist. In this tutorial, we will learn how to create a 404 page for our own website using HTML and CSS.

To begin, we need to create the 404 page using HTML and CSS. You can name the page whatever you like, as long as it makes sense for a 404 page. In this example, the page is named "not-found-page.html". Once you have created the page, you will need to style it and add the necessary content. For this tutorial, we will keep it simple.

The 404 page will include a title and a paragraph informing the user that the page they are looking for does not exist. It should also provide a link to the front page of the website. Here is an example of the HTML code for the 404 page:

```html
<!DOCTYPE html>
<html>
<head>
  <title>404 Page Not Found</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
    }

    h1 {
      color: #333;
    }

    p {
      color: #777;
    }

    a {
      color: #007bff;
      text-decoration: none;
    }
  </style>
</head>
<body>
  <h1>404 Page Not Found</h1>
  <p>The page you are looking for does not exist.</p>
  <p>Go back to the <a href="/">front page</a>.</p>
</body>
</html>
```

Once you have created and styled the 404 page, you will need to upload it to your server. You can use a tool like FileZilla to do this. Simply connect to your server and upload the "not-found-page.html" file to the appropriate directory.

After uploading the file, you will need to configure your server to recognize the 404 page. This step will vary depending on your server setup. You may need to consult your hosting provider or server documentation for specific instructions.

Once the server is configured, if a user tries to access a page that does not exist on your website, they will be redirected to the 404 page you created. They will see the title, the message indicating that the page does not exist, and a link to the front page of the website.

It is important to have a 404 page on your website, as users may accidentally try to access pages that do not exist. By providing a custom 404 page, you can ensure that the user has a better experience and is directed back to your website.

To create a 404 page in HTML, you need to follow a few steps. First, open your text editor and create a new file. Save it as ".htaccess" (without the quotes) - note that the name is crucial and should not be changed. The ".htaccess" file is a configuration file for your server, specifically for an Apache web server.

Inside the ".htaccess" file, you will need to add a single line of code. Use the following syntax: "ErrorDocument 404 /not-found-page.html". This line tells the server to display the "not-found-page.html" file whenever a user tries to access a page on your website that does not exist. Make sure to save the file after adding the code.

Next, you need to upload the ".htaccess" file to the main directory of your website on the server. You can use an FTP client like FileZilla to accomplish this. Once the file is uploaded, go back to your browser and refresh the website. Now, if you enter a URL that does not exist on your website, you will see the custom 404 page you created.

Creating a 404 page is a straightforward process. The ".htaccess" file allows you to configure your server to handle errors and provide a user-friendly experience. While the scope of the ".htaccess" file extends beyond creating a 404 page, this tutorial focuses solely on that functionality. In future tutorials, we will explore other uses for the ".htaccess" file.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: REMOVING THE PAGE FILE EXTENSION FROM THE URL

In this lesson, we will learn how to remove the file extension from a URL using HTML and CSS. This technique is known as URL rewriting and it allows us to create cleaner and more user-friendly URLs for our website.

Before we begin, there are a few important things to note. Firstly, we will be working with the .htaccess file, which is a server configuration file. If you are unfamiliar with this file, don't worry, we will explain how to create it. Secondly, we will go through each line of code that we write, so you will understand what each part does. This is important because making a mistake in the .htaccess file can cause your website to go down. Lastly, the code we use may differ slightly from what you find on the internet. This is because there are many ways to achieve the same result, and we will be using a simple and straightforward approach.

Now, let's get started with the lesson. The first step is to take a look at the website we will be working on. In front of me, I have the website with a page called "cases.html" in the URL. Our goal is to remove the ".html" extension so that the URL simply reads "cases". To do this, we will open our index page in an editor and locate the link to the "cases.html" page. We will delete the ".html" part from the link, as we only need the page name in the URL. We will handle the extension later in the .htaccess file.

Before we make any changes, let's test if the link to the "cases" page works. We will upload the modified index page to our server and then create a new document called ".htaccess". This file will contain the code that enables URL rewriting.

The HT Access file is a configuration file for the server that allows us to create code that runs before the website loads. In this episode, we will focus on creating a URL rewrite using HT Access.

To begin, we need to save a file named ".htaccess" in the root folder of our website. The file extension is crucial, so make sure to name it exactly as ".htaccess". Inside the HT Access file, we will write code that enables the rewrite function and sets some conditions and rules.

First, we need to enable the rewrite function by adding the following line of code:

```
RewriteEngine On
```

Make sure to write it exactly as shown, with "RewriteEngine" capitalized.

Next, we will define the conditions that need to be met for the rule to run. The first condition ensures that if a folder with the same name as the requested document exists, it won't cause an error. This is important to prevent errors when accessing a folder with the same name as a document. Add the following code:

```
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^ - [L]
```

The hyphen "-" in the RewriteRule means that no action will be taken if the condition is met.

The second condition checks if the requested file doesn't exist. If the file doesn't exist, the rule won't run. Add the following code:

```
RewriteCond %{REQUEST_FILENAME} !-f
```

Finally, we will define the rule that performs the URL rewrite. This rule will allow us to access pages without the file extension in the URL. Add the following code:

```
RewriteRule ^([^\.]+)$ $1.html [NC,L]
```

This rule captures the requested URL without the file extension and appends ".html" to it. The [NC,L] flags at the end mean that the rule is case-insensitive (NC) and it will be the last rule to be processed (L).

Save the HT Access file and upload it to the root folder of your website. Now, when you access a page without the file extension in the URL, the server will automatically load the corresponding HTML file.

Remember to name the file ".htaccess" and place it in the root folder of your website for this URL rewrite to work properly.

To remove the file extension from the URL, we need to create a rule using regular expressions. Regular expressions are a powerful tool used to match and manipulate text patterns.

The first step is to write the regular expression, which is enclosed in parentheses. Inside the parentheses, we use a combination of symbols and characters to define the conditions for the search term. In this case, we want to allow any characters before the file name and any characters after it. To achieve this, we use the ".*" symbol, which means "any character, any number of times".

After defining the regular expression, we indicate that we want to grab the URL by using the "$1" symbol. This means that we will capture everything that matches the regular expression.

Next, we specify what we want to add after the captured URL. In this case, we want to add the ".html" extension.

To ensure that the rule applies regardless of whether the file name contains uppercase or lowercase letters, we use the "NC" flag, which stands for "non-case sensitive". This means that the rule will ignore the case of the letters in the URL.

Additionally, we use the "L" flag to indicate that the conditions specified in this rule only apply to this specific rule and not to any subsequent rules.

It is important to note that regular expressions can be complex and may require further study to fully understand their capabilities. If you are interested in learning more about regular expressions, you can explore additional resources on the topic.

To remove the page file extension from the URL, you need to follow a few steps. First, you need to upload the file to your server. Go to your desktop and locate the file you want to upload. Then, drag and drop the file into the main directory of your website.

Next, navigate to your website and go back to the front page. When you click on the "Cases" link, it will show the URL as "internet/cases" but it will also display the content of the "cases.html" file inside the page. This is not the desired outcome, as we want the content to be displayed within the website.

To achieve this, you can use a specific technique. By using server-side scripting or a content management system (CMS), you can configure the server to handle the URL without the file extension. This way, when you click on the "Cases" link, it will display the content within the website without revealing the file extension in the URL.

It's important to note that some people may argue that a forward slash should be added in front of the link inside the URL. However, according to Google Webmasters, it is rare for a forward slash to actually matter within a website. Therefore, if you are concerned about not having a forward slash in front of the link, you can rest assured that it will not significantly impact the functionality of your website in the majority of cases.

Removing the page file extension from the URL can be achieved by uploading the file to your server, configuring the server to handle the URL without the file extension, and ensuring that the desired content is displayed within the website.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: USING CSS POSITION TO MOVE ELEMENTS

Positioning elements in web development using CSS is an important skill to have when it comes to creating visually appealing websites. In this lesson, we will explore the different positioning options available in CSS and how they can be used to move elements within a webpage.

Before we dive into the details of CSS positioning, it's important to understand the concept of the box model. In CSS, every element on a webpage is treated as a rectangular box. The box model consists of four main components: content, padding, border, and margin. The content is the actual content of the element, while the padding is the space between the content and the border. The border is the line that surrounds the element, and the margin is the space between the border and other elements on the page.

When it comes to positioning elements, we have several options to choose from: relative, absolute, fixed, and sticky. Let's take a closer look at each of these options.

1. Relative Positioning:
Relative positioning allows us to move an element relative to its normal position on the page. By using the CSS property "position: relative", we can specify how far the element should be moved from its original position using properties like "top", "bottom", "left", and "right". For example, if we want to move an element 100 pixels down from its original position, we can use "top: 100px".

2. Absolute Positioning:
Absolute positioning allows us to position an element relative to its nearest positioned ancestor. If no ancestor is positioned, then the element will be positioned relative to the initial containing block, which is usually the viewport. By using the CSS property "position: absolute", along with the positioning properties mentioned earlier, we can precisely position elements on the page. This can be useful when we want to overlay elements or create complex layouts.

3. Fixed Positioning:
Fixed positioning is similar to absolute positioning, but the element is positioned relative to the viewport and remains fixed even when the page is scrolled. This can be useful for creating elements like navigation bars or sidebars that stay in a fixed position regardless of the page's scroll position. To use fixed positioning, we can use the CSS property "position: fixed" along with the positioning properties mentioned earlier.

4. Sticky Positioning:
Sticky positioning is a relatively new feature in CSS that allows elements to become sticky when the user scrolls past a certain point. This can be useful for creating elements like sticky headers or sidebars that stay fixed until a specific scroll position is reached. To use sticky positioning, we can use the CSS property "position: sticky" along with the positioning properties mentioned earlier.

It's important to note that when using positioning, we should have a specific purpose in mind. If we want to move an element within its normal flow, it's usually better to use padding or margin. Positioning should be used when we have a specific need to move an element in a precise manner.

Additionally, when using positioning, we may need to consider the "z-index" property. The "z-index" property determines the stacking order of elements on the page. Elements with a higher "z-index" value will appear on top of elements with a lower value. This can be useful when we want to layer elements on top of each other.

Understanding and utilizing CSS positioning is essential for creating dynamic and visually appealing websites. By using the different positioning options available, we can move elements within a webpage with precision and create unique layouts.

CSS Positioning: Moving Elements with CSS Position

In web development, CSS position is a powerful tool that allows us to precisely control the placement of elements on a webpage. There are three main values for the position property: absolute, relative, and fixed. In this tutorial, we will focus on using CSS position to move elements.

To begin, let's consider the absolute positioning. When an element is set to position:absolute, it is taken out of the normal flow of the webpage and can be placed anywhere on the screen. By specifying the top and right properties, we can determine the exact position of the element. For example, setting top:0 and right:0 will move the element to the top right corner of the browser window.

If we want to move the element relative to its current position, we can use position:relative. By adding position:relative to the parent container, we can create a reference point for the child elements. When we specify top and right properties for the child element, it will be moved relative to the parent container. For instance, setting top:100px will move the element 100 pixels down from its original position.

Another useful value for the position property is fixed. When an element is set to position:fixed, it will be positioned relative to the browser window, regardless of scrolling. This is commonly used for creating fixed navigation menus that stay in place as the user scrolls through the webpage.

When using position:fixed, it's important to note that all child elements within the fixed element will also be affected. This means that they will move along with the parent element as it remains fixed on the screen.

In the case of sticky positioning, it used to require JavaScript to achieve the desired effect. However, with the introduction of position:sticky, we can now achieve sticky elements without the need for JavaScript. Sticky elements behave like position:relative elements until they reach a certain scroll position, at which point they become fixed. This is particularly useful for creating sticky headers or sidebars that stay in view as the user scrolls.

To summarize, CSS position is a powerful tool for moving elements on a webpage. By using position:absolute, position:relative, position:fixed, or position:sticky, we can precisely control the placement of elements and create dynamic and interactive webpages.

When designing a website, it may be necessary to have elements, such as a navigation menu, stick to the top of the page when scrolling. This can be achieved using CSS position sticky. By setting the position property to sticky and the top property to zero pixels, the element will stick to the top of the browser window as it is scrolled down. Alternatively, the top property can be set to a specific value, such as a hundred pixels, to create a gap between the element and the top of the website.

It is important to note that not all browsers fully support the position sticky feature yet. For example, Safari does not currently support it. To ensure compatibility, CSS prefixes can be used. The WebKit prefix can be added before the value "sticky" to target certain browsers, such as Safari. Other prefixes, such as -moz- for Firefox and -o- for Opera, can be used to target specific browsers as well. In most cases, the WebKit prefix is sufficient.

Using CSS position to move elements should be done with caution and only when there is a specific purpose in mind. It is not recommended to use position for every element that needs to be moved around on a website. Instead, margins and padding should be used for most layout adjustments. Overusing position can lead to design issues and inconsistencies.

CSS position sticky can be used to make elements stick to the top of the page when scrolling. However, it is important to be aware of browser compatibility and to use position sparingly for specific purposes.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: CREATING VARIABLES IN CSS

CSS variables, also known as custom properties, are a relatively new feature in CSS styling that can greatly simplify the process of creating websites. Similar to variables in programming languages like JavaScript and PHP, CSS variables allow us to assign a value to a container, making it easier to manage and update styles throughout a website.

Imagine you have a large website and you want to change the color of all the text. Instead of manually updating the color in multiple places within your stylesheet, you can use a CSS variable. By defining a variable at the top of your stylesheet, you can then reference that variable throughout your styles and easily update the color in just one place.

To create a CSS variable, you use the double dash notation followed by a name for the variable. For example, you could define a variable called "text-color" using the following syntax:

```
--text-color: #CCC;
```

In this example, the variable "text-color" is assigned the value of "#CCC", which represents a shade of gray. Once the variable is defined, you can use it in any element within your stylesheet by referencing the variable name.

To access the variable, you need to apply it to a child element. In CSS, the highest level element is called the root element, which is typically the HTML tag. To apply the variable to the root element, you can use a pseudo-element called ":root". The styles applied to the ":root" pseudo-element will be inherited by all elements within the website.

Here is an example of how you can apply the CSS variable to a paragraph within a specific section:

```
.section-one p {
color: var(--text-color);
}
```

In this example, the paragraph within the "section-one" class will inherit the color specified by the "text-color" variable.

By using CSS variables, you can easily update styles across your entire website by modifying the value of the variable at the top of your stylesheet. This saves time and effort, especially when dealing with large websites with numerous styles.

To summarize, CSS variables are a powerful tool in web development that allow you to define and reuse values throughout your stylesheets. By using variables, you can easily update styles across your website and maintain consistency in your design.

In CSS, variables can be used to store and reuse values throughout a website's styling. This provides a more efficient way to manage and update styles, especially when there are common values that need to be repeated across multiple elements.

To create a variable in CSS, we use the `--` prefix followed by a name of our choice. For example, we can create a variable called `--main-color` and set its value to a specific color, like `#CCC`. This variable can then be referenced in any styling rule by using the `var()` function. So instead of directly setting the color value, we can use `var(--main-color)`.

By using variables, we can easily update the value of `--main-color` in one place, and it will automatically be reflected in all the styles that reference it. This is particularly useful when we want to change the color scheme of a website or have a consistent color throughout.

We can also create multiple variables to store different values. For example, we can create a variable called `--tip-box-border` and set its value to `solid 1px`. This variable can then be used to define the border style for any element in the website.

Variables can also be used within child elements. For example, if we have a variable called `--text-size` with a value of `14 pixels`, we can apply it to a paragraph element inside a section by using `font-size: var(--text-size)`. This allows for consistent styling across related elements.

It's important to note that variables can only be used within the same HTML page or within child elements. They cannot be accessed across different pages or elements that are not directly related.

Using variables in CSS provides a convenient way to manage and update styles, making it easier to maintain a consistent design across a website. It is recommended to utilize variables when creating websites in HTML, PHP, or any other programming language that supports CSS.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: CSS PSEUDO ELEMENTS AND CLASSES

A pseudo element is a way to style a specific part of an element in CSS. There are two types of pseudo elements: pseudo classes and pseudo elements. Pseudo classes change the state of an element, such as when it is hovered over or clicked on. Pseudo elements, on the other hand, allow us to select and style a specific part of an element.

One example of a pseudo class is the root pseudo element. The root element is the highest level element in a website, and styling it will apply to all sub elements. For example, if we style the root element to have a red color, all text inside the website will have that color.

Another commonly used pseudo class is the hover pseudo element. This allows us to create hover effects on elements. For example, if we apply the hover pseudo element to all h1 tags, when we hover over a header, it will change color. This is a useful effect to have on websites.

These are just a few examples of the many pseudo elements that can be used in CSS. Pseudo elements are a powerful tool for styling websites and can be used to create dynamic and interactive effects.

In web development, HTML and CSS are fundamental languages used to create and style websites. In this lesson, we will focus on extending our skills in CSS by exploring pseudo elements and classes.

One way to enhance the design of a webpage is by adjusting the size and positioning of text. We can achieve this by using CSS properties such as margin, padding, and heading. For example, to move text around, we can set the margin or padding values. By using the "left" property, we can position text 50 pixels from the left side. It's important to note that the appearance of text may vary depending on the browser and font size settings.

We can also modify the font size to further customize the appearance of text. By changing the font size property, we can make the text larger or smaller. Additionally, we can apply different styles when hovering over an element. This can be done by using the "hover" pseudo class in CSS. For example, when hovering over a header, we can change its appearance, such as the color. We can also add transitions to make the hover effects smoother.

Moving on to pseudo elements, let's explore the "active" pseudo class. When clicking on a link, the "active" class can be applied to change the text color, for instance, to green. However, once the mouse button is released, the color change will not persist. Another pseudo class is "visited", which can be used to style links that have been previously visited. When a link is visited, it can be styled differently, such as changing the color.

Now, let's discuss the "last-child" and "nth-child" pseudo classes. The "last-child" class allows us to select the last element of a specific type within a container. For example, we can select the last paragraph within a div box and apply a style, like changing the color to blue. On the other hand, the "nth-child" class allows us to select specific elements based on their position. We can select elements using numbers or keywords like "even" or "odd". For instance, we can select the second child element and apply a style, such as changing the color to pink. We can also select all even or odd-numbered elements and apply a consistent style to them.

These are just a few examples of the CSS pseudo elements and classes that can be used to enhance the design and functionality of a website. By utilizing these techniques, web developers can create visually appealing and interactive webpages.

In CSS, pseudo elements and pseudo classes are used to style specific parts or states of elements. Pseudo classes change the state of an element, while pseudo elements can change part of the element itself.

One example of a pseudo class is the "nth-child" selector. This selector allows you to style every nth element of a parent element. For example, using the "nth-child(2n+1)" selector will style every odd-numbered child element. This can be useful for styling alternating rows in a table or applying different styles to specific elements.

Another example of a pseudo class is the "nth-of-type" selector. This selector allows you to style every nth element of a specific type within a parent element. For example, using the "nth-of-type(3n)" selector will style every third paragraph element within a parent container.

Pseudo elements, on the other hand, allow you to style specific parts of an element. One example is the "::first-line" pseudo element. This allows you to style the first line of text within a block-level element. For example, you can change the background color and text color of the first line of a paragraph using the "::first-line" pseudo element.

Similarly, the "::first-letter" pseudo element allows you to style the first letter of a block-level element. By using the "::first-letter" pseudo element, you can apply different styles to the first letter of a paragraph, such as changing its font size or color.

Another useful pseudo element is the "::selection" pseudo element. This allows you to style the selected text within an element. For example, you can change the background color and text color of the selected text within a paragraph.

Lastly, the "::before" and "::after" pseudo elements are commonly used in web development. These elements allow you to insert content before or after an element. This content can be text, images, or other HTML elements. By using the "::before" and "::after" pseudo elements, you can add decorative elements or additional content to your web page.

Pseudo elements and pseudo classes are powerful tools in CSS that allow you to style specific parts or states of elements. By understanding and utilizing these selectors, you can create more dynamic and visually appealing web pages.

In HTML and CSS, we have the ability to extend the styling of elements using pseudo elements and classes. Pseudo elements allow us to insert content before or after an element, while classes allow us to insert the value of an attribute as text within an element.

To insert content before an element, we can use the "::before" pseudo element. We can specify the content we want to insert by using the "content" property in CSS. For example, if we want to insert plain text before an element, we can use the following code:

```css
.element::before {
  content: "This is a paragraph - ";
}
```

By applying this code to an element, we will see the specified text inserted before the element.

Alternatively, we can insert the value of an attribute as text within an element. To do this, we can use the "attr()" function in CSS. We specify the name of the attribute we want to use and enclose it in parentheses. For example, if we want to insert the value of the "href" attribute as text, we can use the following code:

```css
.element::before {
  content: attr(href);
}
```

By applying this code to an element, we will see the value of the "href" attribute inserted as text before the element.

This technique can be used for any attribute, such as class or ID. We can also create our own attributes and use their values as text within elements.

Additionally, we can insert images before or after an element using the "url()" function in CSS. We specify the URL of the image we want to insert. For example, if we want to insert an image before an element, we can use the following code:

```css
.element::before {
  content: url(path/to/image.png);
}
```

By applying this code to an element, we will see the specified image inserted before the element.

It's important to note that these techniques can also be used with the "::after" pseudo element, which inserts content after an element.

HTML and CSS provide us with the ability to extend the styling of elements using pseudo elements and classes. Pseudo elements allow us to insert content before or after an element, while classes allow us to insert the value of an attribute as text within an element. We can also insert images using the "url()" function. These techniques provide flexibility in customizing the appearance of our webpages.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: CREATING TRANSITIONS USING CSS

In web development, transitions are a powerful tool for creating smooth animations and effects. In this lesson, we will focus on using CSS to create transitions. Transitions allow us to smoothly change the style of an element over a specified duration.

To understand transitions, let's consider an example. Imagine we have a tip box with a background image. Currently, when we hover over the box, the image changes instantly. However, we want to create a smooth transition between the two images.

To achieve this, we can use the `transition` property in CSS. This property allows us to specify which CSS properties should transition and how long the transition should take. We can also define the timing function and delay for the transition.

To apply a transition to our element, we need to add the `transition` property to its CSS styling. For example, if we want to transition the background image, we can set the `transition-property` to `background-image`. To control the duration of the transition, we can use the `transition-duration` property. In our case, we can set it to 2 seconds for a slow transition.

Additionally, we can define the timing function for the transition using the `transition-timing-function` property. This allows us to control how the transition progresses over time. Common options include `ease`, `ease-in`, `ease-out`, and `ease-in-out`.

Lastly, we can introduce a delay before the transition starts using the `transition-delay` property. For example, we can set it to 4 seconds to delay the transition.

It is important to note that while transitions work in most modern browsers, there may be some older browsers that do not support them fully. Therefore, it is always recommended to test your transitions in different browsers to ensure compatibility.

By using transitions, we can create visually appealing effects and animations on our website. Whether it's changing images, text, or other elements, transitions provide a smooth and engaging user experience.

Transitions in CSS allow us to create smooth animations and effects on our website. We can control the properties to transition, the duration, timing function, and delay. By incorporating transitions into our CSS, we can enhance the user experience and make our website more engaging.

To create transitions using CSS, we need to add prefixes to our code in order to support different browsers and older versions. A prefix is a code snippet that is added before the styling to make it work in specific browsers. For example, to add support for all versions of Chrome, we need to add the prefix "-webkit-". Similarly, for Firefox, we use the prefix "-moz-", and for Opera, we use the prefix "-o-".

However, instead of adding all these prefixes manually, we can simplify the code by using a shorthand notation. We can write the transition property followed by a colon and a semicolon. Inside the brackets, we specify the property we want to change, the duration of the transition, the timing function, and any delay we want to add.

For example, if we want to change the width of an element over a duration of 2 seconds, with an easing-in timing function and a 1-second delay, we can write:

```
transition: all 2s ease-in 1s;
```

To add support for different browsers, we can copy this code and paste it three more times, adding the respective prefixes before each line. This way, we ensure that the transition works correctly in Chrome, Firefox, and Opera.

By using this shorthand notation, we can reduce the amount of code we need to write and maintain, making it easier to create transitions in CSS. Transitions can be applied to various properties, such as width, height, padding, and more, allowing us to add stylish and smooth effects to different elements on a website.

In the provided example, a transition effect is applied to a box, which is a link. When hovering over the box, it slowly opens out and reveals a small downward arrow. This effect is achieved by changing the width and height of the box using the transition property. By adding transitions to elements on a website, we can enhance the visual appeal and create a more engaging user experience.

To create transitions using CSS, we can use the shorthand notation for the transition property and specify the desired changes, duration, timing function, and delay. By adding prefixes, we ensure cross-browser compatibility. Transitions can be applied to various properties, allowing us to enhance the visual appeal of elements on a website.

Transitions are a powerful feature in CSS that allow us to create smooth animations and effects without the need for JavaScript. By using transitions, we can change various properties of elements, such as height, width, padding, background color, and more, with a gradual and visually pleasing transition.

To create a transition, we need to specify the property we want to transition, the duration of the transition, and optionally the timing function. The property can be any CSS property that has a numerical value, such as width, height, padding, margin, and so on.

For example, let's say we have a div element and we want to change its height when a certain event occurs. We can achieve this by applying a transition to the height property. We can specify the duration of the transition in milliseconds, for example, 500 milliseconds. This means that the height change will take half a second to complete.

```
div {
  transition: height 500ms;
}
```

Now, when the height of the div changes, either through user interaction or JavaScript manipulation, the transition will smoothly animate the height change over the specified duration.

We can also apply transitions to multiple properties at once by separating them with commas. For example, if we want to transition both the height and the background color of an element, we can do it like this:

```
div {
  transition: height 500ms, background-color 500ms;
}
```

By default, transitions have an ease timing function, which means that the animation starts slowly, accelerates in the middle, and slows down towards the end. However, we can specify different timing functions to achieve different effects. Some common timing functions are ease-in, ease-out, ease-in-out, linear, and cubic-bezier.

Transitions can be used to create a wide range of effects and animations. For example, we can create a button that changes color when hovered over, or a menu that slides in and out when clicked. The possibilities are endless, and the only limit is our creativity.

It's important to note that transitions are supported in all modern browsers, including Internet Explorer 10 and above. However, older versions of Internet Explorer, such as IE9 and below, do not support transitions. In such cases, we can use JavaScript to achieve similar effects.

Transitions are a powerful tool in CSS that allow us to create smooth animations and effects without the need for JavaScript. By specifying the properties we want to transition and the duration of the transition, we can create visually appealing and interactive web experiences.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: CREATING WEBSITE LAYOUTS USING CSS GRID

CSS grids are a new way to create layouts inside websites. In this material, we will discuss what CSS grids are, how they compare to traditional layout methods, and how to set up and use CSS grids in a real website.

Before we can start using CSS grids, we need to have a basic HTML file with a linked stylesheet and a meta tag for responsive design. CSS grids were invented to address the limitations of traditional layout methods, such as using floats and absolute positioning. With CSS grids, developers can use CSS alone to determine the placement of elements within the browser.

To demonstrate the use of CSS grids, let's consider a basic front page layout with a header, a section, a sidebar, and a footer. Traditionally, we could use floats to position the sidebar and content next to each other. However, this approach becomes problematic when trying to achieve different layouts for different devices. For example, if we want the footer to appear next to the header on a mobile device, it is not possible using floats.

CSS grids solve this problem by allowing developers to create a grid within the page and insert elements from the HTML file into the grid using CSS. This means that the order of elements in the HTML file does not matter, and we can place them anywhere within the grid. This provides flexibility and ease of creating responsive layouts.

While frameworks like Bootstrap offer responsive grid systems, CSS grids have their own advantages. Bootstrap's flex grid is not the same as CSS grids, and there are benefits to using CSS grids instead. For more information on the differences between Bootstrap and CSS grids, refer to the provided link.

To set up a CSS grid, we need to include a stylesheet with a reset code to remove any unwanted spacing. The code example shown on the screen provides a basic reset code that sets the margin and padding to 0.

CSS grids offer a more flexible and responsive way to create layouts in websites. By using CSS alone, developers can easily position elements within a grid, regardless of their order in the HTML file. This allows for greater control over website layouts on different devices.

In this didactic material, we will explore the fundamentals of creating website layouts using CSS grid. CSS grid is a new concept for many individuals who are accustomed to using float for layout designs. To facilitate a clear understanding of CSS grid, we will utilize animations to visualize the concepts.

To begin, let's set up the elements inside the body tag. We will create a container, which will serve as the grid for our website's content. This container will hold the header, content, sidebar, and footer. To create the container, we will use a div element with a class of "grid".

Inside the "grid" class, we will insert multiple divs to represent the different sections of our website. For demonstration purposes, we will insert five divs with the numbers one through five. We will then apply a background color to each div to visualize their placement in the browser.

To differentiate the divs within the "grid" class, we will assign different background colors to the even and odd divs. We will use the CSS pseudo-class "nth-child" to select the even and odd divs and apply the desired background colors.

Now, if we view the website in the browser, we can observe the placement of the divs. Each div represents a different section of the website, such as the title, header, sidebar, content, and footer.

To utilize the power of CSS grid, we need to define the grid itself. Inside the CSS file, we will add styling for the "grid" class. We will set the display property of the "grid" class to "grid". This will define the divs inside the "grid" class as a grid.

However, simply defining the grid is not sufficient. We also need to specify the number of columns and rows within the grid. To define the number of columns, we will use the "grid-template-columns" property. By default, the grid has no columns. We will define one column using the "1fr" measurement, which represents one fraction of the grid's width. To add additional columns, we can specify additional fractions.

In addition to defining the columns, we also need to determine where the content will be placed within the grid. To do this, we refer to the lines outside the columns. In our current illustration, we have line one and line two, which define the two columns within the grid.

After implementing these changes, we can refresh the browser to see the updated layout.

To create website layouts using CSS grid, we can utilize various techniques to control the positioning and sizing of elements within the grid. In this didactic material, we will explore how to manipulate columns and rows within the grid, as well as how to span elements across multiple lines.

Firstly, when working with columns, we can specify the width of a column using different measurements such as fractions, pixels, or percentages. For example, by setting a column to be 200 pixels wide, we can control its size within the grid. Similarly, using percentages allows for more flexible and responsive designs. Additionally, we can use the value "auto" to make a column adjust its width to fit the content it contains.

Moving on to the grid items, which are the elements placed within the grid, we can determine how much space they occupy. By assigning specific classes to these items, we can target them in our CSS styles. For instance, we can define a class called "title" and specify that it should span from the first column to the second column, effectively taking up the entire row.

To achieve this, we have two approaches. The first approach involves using the properties "grid-column-start" and "grid-column-end". By setting the start line to 1 and the end line to 3, we ensure that the title spans across the entire row. The second approach utilizes the "grid-column" property, where we can specify the start and end lines in a shorthand format.

Additionally, we can use the "span" keyword to extend an item's span across multiple lines within the grid. For example, by setting the end line to span 2 lines, the item will occupy two lines vertically.

Similarly, we can control the positioning of items within rows using properties like "grid-row-start" and "grid-row-end". By specifying the start and end lines, we can determine where an item appears vertically within the grid.

To summarize, CSS grid provides powerful tools for creating website layouts. By manipulating columns and rows, we can control the size and position of elements within the grid. We can use measurements like pixels and percentages to define column widths, and we can span items across multiple lines using the "grid-column" and "grid-row" properties. With these techniques, we can create flexible and responsive website layouts.

Creating Website Layouts Using CSS Grid

CSS Grid is a powerful tool that allows web developers to create flexible and responsive website layouts. With CSS Grid, you can easily define the structure of your webpage using a grid system, without having to worry about columns or rows.

To create a website layout using CSS Grid, you can start by assigning a grid area to each section of your webpage. For example, you can give the title section a grid area called "title". Similarly, you can assign grid areas for the header, sidebar, content, and footer sections.

Once you have assigned grid areas to each section, you can define the columns and rows of your grid. This can be done using the "grid-template-rows" property. For instance, you can set the height of the title section to one frame, and do the same for the header, content, sidebar, and footer sections.

To specify the layout of the grid, you can use the "grid-template-areas" property. This property allows you to define the placement of each section within the grid. For example, you can use double quotes to specify the placement of each section in the grid, with each row representing a different section.

By defining the grid template areas, you can arrange the sections of your webpage in a desired layout. For example, you can have the title section span across the entire grid, while the header, sidebar, content, and footer sections are placed in specific areas of the grid.

CSS Grid also allows you to easily modify the layout of your webpage. For instance, you can move the footer section to the top of the layout by simply switching its position with the title section in the grid template areas.

In addition, CSS Grid provides flexibility in handling white space. You can adjust the placement of sections within the grid to create different spacing effects. For example, you can have the header section appear on the left side of the columns by deleting the right header and adding punctuation instead.

CSS Grid is a powerful tool for creating website layouts. It simplifies the process of defining the structure of your webpage and allows for flexibility in arranging and modifying sections within the grid. With CSS Grid, you can create visually appealing and responsive website layouts.

CSS grid is a powerful tool in web development that allows for the creation of complex website layouts. One useful feature of CSS grid is the ability to create wrappers, which can be used to contain and organize elements within a website. Wrappers are commonly used in website design to create a consistent layout and to ensure that content is displayed in a visually pleasing manner.

To create a wrapper using CSS grid, you can insert additional columns within the desired section of your code. By specifying the width of these columns, you can create white spaces on either side of your website. This creates a wrapper effect, where the content is contained within a specific area on the page. By adding the appropriate CSS properties and values, you can ensure that the wrapper is applied consistently throughout your website.

CSS grid also allows for the customization of grid items within the layout. By using the justify-self property, you can specify the horizontal positioning of a grid item within the grid. For example, you can align an item to the left, right, or center of the grid. Similarly, the align-self property allows for the vertical positioning of grid items within the grid.

Nested items are another feature of CSS grid that allows for greater flexibility in website design. Nested items are elements that are contained within a grid item. Although they are not part of the main grid, nested items can be positioned and styled independently from the main grid. This allows for more intricate and complex layouts within a website.

Mobile responsiveness is an important aspect of modern web design, and CSS grid makes it easy to create responsive layouts. With CSS grid, you can easily adapt your website's layout to different screen sizes and devices. By using media queries and adjusting the grid properties and values, you can ensure that your website looks great on both desktop and mobile devices.

CSS grid is a powerful tool for creating website layouts. By utilizing wrappers, customizing grid items, working with nested items, and ensuring mobile responsiveness, you can create visually appealing and user-friendly websites.

CSS Grid is a powerful tool that allows us to create complex website layouts with ease. By using CSS Grid, we can define rows and columns and place elements within them, giving us full control over the layout of our website.

One of the key features of CSS Grid is the ability to create responsive designs using media queries. Media queries allow us to apply different styles based on the size of the viewport. This means that we can create different layouts for desktop, tablet, and mobile devices, ensuring that our website looks great on any screen size.

To create a responsive layout using CSS Grid, we can define our grid in the main stylesheet. Then, inside a media query, we can make changes to the grid to adapt it to different screen sizes. For example, we can adjust the number of columns, change the placement of elements, or even add or remove elements altogether.

To demonstrate this, let's consider an example where we have a grid layout with three columns for desktop devices. Inside a media query for smaller screens, we can modify the grid to have only one column, making it more suitable for mobile devices. We can also add a gap between the grid items to improve the spacing.

By using media queries, we can easily switch between different layouts based on the screen size. This allows us to provide the best user experience for our website visitors, regardless of the device they are using.

It's important to note that CSS Grid is not supported by all browsers, particularly older versions of Internet Explorer and Microsoft Edge. However, it's worth mentioning that the trend in web design is to prioritize mobile-first design, which means focusing on creating a layout that works well on mobile devices and then adapting it for larger screens. Therefore, even if a browser doesn't support CSS Grid, it will still display the mobile version of the website, which is generally more important.

By using CSS Grid and embracing its capabilities, we can push for wider adoption and encourage browsers to support this modern layout system. As more and more websites start using CSS Grid, browser developers will be motivated to implement support for it, ensuring a better web experience for everyone.

CSS Grid is a powerful tool for creating website layouts. By using media queries, we can easily adapt our layouts to different screen sizes, providing a responsive design that looks great on any device. While not all browsers currently support CSS Grid, the benefits of using it outweigh the limitations. By embracing CSS Grid and advocating for its use, we can help drive its adoption and improve the overall web experience.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: ADDING A FAVICON TO A WEBSITE IN HTML

A favicon is a small icon that appears in the browser tab when viewing a website. It is a visual representation of the website and can be customized to display the website logo or any other desired image. In this tutorial, we will learn how to add a favicon to a website using HTML.

To begin, you will need an image that you would like to use as the favicon. It is recommended to use a small image to save space and resources. The image should be in PNG format and ideally not larger than 32x32 pixels.

First, create a folder in the root directory of your website called "image". Inside this folder, place the favicon image file. For example, you can name it "favicon.png".

Next, open your HTML document and locate the <head> tag. Inside the <head> tag, we will add a link element to specify the favicon.

Add the following code inside the <head> tag:

```
<link rel="shortcut icon" type="image/png" href="image/favicon.png">
```

Let's break down the code:

- The `rel` attribute specifies the relationship between the current document and the linked resource. In this case, we set it to "shortcut icon" to indicate that it is a favicon.
- The `type` attribute specifies the MIME type of the linked resource. In this case, it is set to "image/png" to indicate that the favicon is in PNG format.
- The `href` attribute specifies the location of the favicon image file. In this example, it is set to "image/favicon.png".

Save your HTML document and open it in a web browser. When you refresh the page, you should see the favicon displayed in the browser tab.

Adding a favicon to your website is a simple and effective way to enhance its branding and visual appeal. It helps users easily identify and recognize your website when multiple tabs are open.

Remember to keep the favicon image small and optimized to ensure fast loading times and efficient resource usage.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: CREATING A HTML DROPDOWN MENU

In this didactic material, we will learn how to create a dropdown menu using HTML and CSS. The dropdown menu will appear when the cursor hovers over a menu item.

To begin, we need to have a basic website structure. We will create a new HTML file and save it as "index.html". We will also create a new CSS file and save it as "style.css". In the HTML file, we will include the necessary HTML tags, such as the doctype declaration, head, and body tags. We will also link the CSS file to the HTML file by using the link tag inside the head tag.

Inside the body tag, we will create a navigation section. We will use the HTML5 nav tag as the container for the navigation. Inside the nav tag, we will create an unordered list (ul) to hold the menu items. Each menu item will be represented by a list item (li) and a link (a) tag. To indicate that a menu item has a dropdown, we will use a hash tag (#) as the href attribute value for the link.

We can add as many menu items as we want by duplicating the list item and link tags. In this example, we will add four menu items: Home, Portfolio, About Me, and Contact. We can also include a logo or website name before the navigation section by adding a paragraph (p) tag.

To create the dropdown effect, we need to add another unordered list inside the list item that we want to have a dropdown. This will be done right before closing the list item tag. This second unordered list will contain the dropdown menu items. We can add as many dropdown menu items as we want by duplicating the list item tags inside this second unordered list.

It is worth noting that there are other ways to create dropdown menus using HTML, such as using div tags or button tags. However, in this example, we will be using unordered lists.

By following these steps, we can create a dropdown menu using only HTML and CSS, without the need for JavaScript or jQuery.

In this didactic material, we will learn how to create a dropdown menu using HTML and CSS. A dropdown menu is a common feature in web development that allows users to navigate through different sections of a website.

To create a dropdown menu, we will start by creating an unordered list in HTML. Inside this list, we will create anchor tags for each menu item. For example, we can have menu items like "Digital Art," "Video Production," and "Web Development." To make the menu items clickable, we need to include the appropriate links.

To style the navigation, we can use CSS. We can set the width of the navigation to 100% and give it a height of 60 pixels. We can also set a background color to make it visually appealing.

Next, we will style the website name inside the navigation. We can use a paragraph tag and apply styles like font family, font size, color, and line height. Additionally, we can set the float property to "left" to align the website name to the left side of the navigation.

To style the menu items, we will use an unordered list. We can set the float property to "left" to align the menu items horizontally. We can also remove the bullet points using the "list-style" property.

After applying these styles, we can preview the website in a browser. We might notice that the dropdown menu is visible even when we haven't hovered over it. To fix this, we need to add some CSS to hide the dropdown until it is hovered over.

To complete the styling, we can add additional CSS properties like margin and padding to adjust the spacing and layout of the website.

By following these steps, we can create a functional and visually appealing dropdown menu for our website.

To create a dropdown menu in HTML and CSS, follow these steps:

1. Open your HTML file and create a navigation section using the `<nav>` element. Inside the navigation section, create an unordered list using the `<ul>` element.

2. Inside the unordered list, create list items using the `<li>` element. Each list item will represent a menu item in the dropdown.

3. Inside each list item, create an anchor tag using the `<a>` element. This will be the clickable link for each menu item.

4. In your CSS file, target the navigation section using the `nav` selector. Apply a padding of 0 pixels to the top and bottom, and 20 pixels to the left and right. This will create some distance between the navigation and other elements on the page.

5. Target the anchor tags inside the navigation section using the `nav a` selector. Apply styling to these anchor tags to customize the appearance of the menu items. For example, you can set the padding to 24 pixels for the top and bottom, and 14 pixels for the left and right. Set the display property to block to ensure the menu items are displayed vertically.

6. Adjust the font size of the anchor tags to differentiate them from the website name. For example, you can set the font size to 14 pixels.

7. Remove the text decoration (underline) from the anchor tags using the `text-decoration` property and set it to none.

8. To create the dropdown effect, target the list items inside the navigation section using the `nav li` selector. Set the position property to relative to allow for absolute positioning of the dropdown items later.

9. Inside each list item, create another unordered list using the `<ul>` element. This will be the dropdown menu.

10. Apply the `display:block` property to the dropdown menu to ensure it is displayed vertically.

11. Customize the appearance of the dropdown menu by targeting the unordered list inside the list items using the `nav li ul` selector. For example, you can set the background color to white to make it stand out from the rest of the page.

12. To decrease the spacing between the menu items in the dropdown, target the anchor tags inside the dropdown menu using the `nav li ul li a` selector. Adjust the padding as desired. For example, you can set it to 12 pixels for the top and bottom.

13. Test your dropdown menu by refreshing the browser. You should now have a functional dropdown menu with customized styling.

In this tutorial, we will learn how to create a basic dropdown menu using HTML and CSS. We will start by adding a hover effect to the dropdown items so that they change color when the cursor is hovered over them. To do this, we will use the CSS pseudo-class "hover". Inside the hover effect, we will set the background color to a light gray color, specifically #f3f3f3.

Next, we will address a couple of issues. First, we need to decide whether we want the background color of the dropdown to extend all the way from left to right or if we want some padding on the sides. To add padding, we will modify the CSS for the unordered list, which represents the dropdown menu. We will set the padding to 10 pixels on all sides.

The next issue we will tackle is preventing the text in the dropdown from wrapping onto multiple lines. To achieve this, we will style the list items inside the dropdown and set a fixed width for them. In this example, we will set the width to 180 pixels, but you can adjust it to fit your needs.

Before we activate the hover effect, let's change the background color of the dropdown items back to the light gray color. We will also add some rounded corners to the dropdown to give it a more visually appealing look. We will use the CSS property "border-radius" and set it to 4 pixels.

Finally, we will activate the hover effect by initially setting the display of the unordered list (dropdown) to "none". Then, when we hover over the menu item in the navigation, we will change the display to "block" to show the dropdown. We will achieve this by using the CSS pseudo-class "hover" and targeting the unordered list inside the list item.

By following these steps, we can create a basic dropdown menu using HTML and CSS without the need for JavaScript or jQuery.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: KEEPING A FOOTER AT THE BOTTOM OF A PAGE

In order to keep a footer at the bottom of a webpage, we can use CSS. By setting the height of the container div to 100% of the browser's height, the footer will be pushed down below the content of the webpage. When the content becomes larger and reaches the bottom of the page, the footer will start moving down as well.

To implement this, we need to create a basic HTML structure with a section inside the body tag and the footer starting right underneath the section. Then, in a separate CSS file, we can style the elements accordingly.

First, we create a div with an ID of "container" that will cover the entire height of the browser. Inside this div, we create another div with an ID of "main" to contain the content of the webpage. Finally, we create the footer with an ID of "footer".

The purpose of the "container" div is to be 100% of the browser's height, which will cause the footer to be pushed down below the visible content of the webpage. The "main" div will increase the height of the "container" div as the content grows, pushing the footer further down.

To make the footer visible when the page loads, we need to apply some styling. In the CSS file, we set the margin and padding of all elements to zero to remove any default spacing. Then, we style the HTML and body tags by setting their height to 100%. This allows us to set the height of the "container" div to 100% as well.

By using CSS and the proper HTML structure, we can keep a footer at the bottom of a webpage. The "container" div, set to 100% height, pushes the footer below the visible content, and the "main" div increases the height of the "container" div as the content grows.

To keep a footer at the bottom of a web page, we need to apply a few CSS styles. First, we set the height of the footer to a specific value, such as 100 pixels. This ensures that the footer has a fixed height. Next, we style the footer by adding a background color and setting the position to relative. The position property allows us to position the footer relative to its normal position.

To make sure the footer stays at the bottom of the page, we need to adjust the positioning of the main content container. We set the overflow property of the main container to auto. This ensures that a scrollbar appears when the content inside the container exceeds its height. Additionally, we set the padding-bottom property of the main container to the same value as the height of the footer. This creates space at the bottom of the container for the footer.

To move the footer back into view, we set a negative margin-top value equal to the height of the footer. This moves the footer up by the same amount, effectively positioning it at the bottom of the page. Finally, we add a clear property with the value of both to clear any floats that may affect the positioning of the footer.

By applying these styles, we can keep the footer at the bottom of the web page, even when there is a large amount of content. The padding in the main container ensures that the content does not disappear underneath the footer. This technique is useful when designing websites to ensure a consistent layout and user experience.

To keep a footer at the bottom of a web page, we need to set the height of the footer, style it, adjust the positioning of the main content container, and use negative margin and clear properties. This ensures that the footer stays at the bottom of the page regardless of the amount of content.

EITC/WD/HCF HTML AND CSS FUNDAMENTALS DIDACTIC MATERIALS
LESSON: HTML AND CSS EXTENDING SKILLS
TOPIC: CREATING A GOOGLE MAP IN A WEBSITE

In this didactic material, we will learn how to insert a Google API map into a website. This means that we will take the map from Google Maps and incorporate it into our own website. The final result will be a functional map that can be resized, dragged, zoomed in and out, and customized according to our needs.

To begin, we need to create a new HTML document with the basic HTML5 structure. It is important to note that in order to use Google's API map, we must include the HTML5 doctype declaration at the beginning of our document.

Next, we need to access Google's API guideline, which provides valuable references for using the API map. We will not go through the guideline directly from the website, but I will provide a link in the description for you to explore it further.

Now, let's move back to our coding program and start styling the map. To insert the map into our website, we need to create a div element with an ID of "map" inside the body tag. We can then style this div according to our preferences. For example, we can set the height to 500 pixels and the width to 100% to make it span the entire width of the browser.

To ensure that there are no default margins or paddings interfering with our map, we need to include a CSS rule that sets the margin and padding of all elements to zero. This can be achieved by using the universal selector (*) and setting the margin and padding properties to zero.

After styling the div, we can now move on to adding the necessary JavaScript code to make the API map work. Although we haven't covered JavaScript yet, I will explain the code in a simple and straightforward manner. We will create a function called "initMap" that will be responsible for initializing the map and specifying its behavior.

Inside the "initMap" function, we can define various settings for the map, such as the location and the marker. This is important because we need to specify where we want the marker to be placed on the map. Without this information, the map would not serve any purpose on our website.

To include the JavaScript code, we need to add a script tag to our HTML document. This is where we can write our JavaScript code. Inside the script tags, we will define the "initMap" function and provide instructions on how the map should function.

Please note that this is a simplified explanation of the code, and we will cover JavaScript in more detail in future lessons. For now, it is important to understand the basic structure and purpose of the code we are writing.

By following these steps, we can successfully insert a Google API map into our website. This will allow us to enhance our website's functionality and provide users with an interactive and dynamic map experience.

In web development, it is common to include a Google Map on a website to provide location-based information or services. To achieve this, we need to use HTML and CSS to extend our skills and create a Google Map on our website.

One important concept we need to understand is variables. In this case, a variable is a piece of data that we need to set equal to something. For our Google Map, we will create a variable called "location" and set it equal to latitude and longitude coordinates. These coordinates can be found online and are typically represented as two numbers.

To create the "location" variable, we use curly brackets {} and inside them, we define the latitude and longitude. For example, we can set the latitude to -25.363 and the longitude to 131.044. These numbers can be found using websites that provide latitude and longitude coordinates. After defining the coordinates, we need to end the line of code with a semicolon.

Next, we need to insert the Google Map into our website. We do this by creating another variable called "map" and setting it equal to "new Google.map". This refers to a method that allows us to create a new Google Map. Inside the parentheses, we use the "document.getElementByID()" method to select the element with the ID "map" in our HTML code. This element is where we want to insert the Google Map. We also set some additional options for the map, such as the zoom level and the center location.

To use Google Maps on our website, we need to obtain a key from Google. This key gives us permission to use the Google Maps service. We can get a free key with some limitations on the number of visits per day or per hour. Once we have the key, we need to include it in our code to activate the Google Maps feature on our website.

To create a Google Map on a website, we need to define a variable for the location coordinates, insert the Google Map into the desired element using the "getElementByID()" method, set options for the map, and obtain a key from Google to activate the Google Maps feature.

To create a Google Map in a website, we need to follow a few steps. First, we need to obtain a key from the Google Maps website. This key will allow us to use the Google Maps API in our website. Once we have the key, we can proceed with the code.

To begin, we need to add a script tag in our HTML code. Inside this script tag, we need to include the key using the source attribute. The source attribute should be set to the Google Maps API URL, followed by the key. Additionally, we need to include the attributes "async" and "defer" to ensure the script loads properly. Finally, we need to add the attribute "callback" with the value "initMap" to initialize the map.

After adding the script tag, we can save the code and refresh the browser. Now, we should see a map displayed on our website. We can zoom in and out using the mouse control or the provided buttons.

Next, we can add a marker to indicate a specific location on the map. Inside the script tag, we need to create a new variable called "marker" and set it equal to "new google.maps.Marker()". Within the parentheses, we can specify the position of the marker using the "location" property. This location should be the same as the one we set for the center of the map. Additionally, we need to include the "map" property and set it equal to the map variable we created earlier.

Once the marker code is added, we can save the code and reload the map. Now, we should see a marker on the map indicating the desired location.

To create a Google Map in a website, we need to obtain a key from the Google Maps website, include the key in a script tag, initialize the map using the "callback" attribute, and add a marker to indicate a specific location on the map.