**Tutorial 5 - CSS**

Repo link: <https://github.com/tinkerness/S7_Web_Programming/tree/main/tutorials/tut5>

Hosted link: <https://tinkerness.github.io/S7_Web_Programming/tutorials/tut5/index.html>

1. **Create an unordered list in HTML forms with the options CS, EC, EEE and IT with disc as the bullet type.**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<ul style="list-style-type:disc;">

<h2>BRANCHES</h2>

<li>CS</li>

<li>EC</li>

<li>EEE</li>

<li>IT</li>

</ul>

</body>

</html>

1. **What are elements in HTML? Compare inline and block elements.**

In HTML, elements are the fundamental building blocks used to structure and define content within a web page. An element in HTML has a structure that consists of the start tag, content, and then the end tag. There are two display values - block and inline elements in HTML.

**Block elements** in HTML begin on a new line and occupy the complete horizontal space of its parent element and have the same height as the content. They create a visual block-level structure in the HTML document. Examples :- <div>, <p>, <h1> to <h6>, <ul>, <ol>, <li>, <table>, <form>.

**Inline elements** in HTML are elements that do not create a line break before or after the element. They are typically used within text or other inline elements to apply formatting or behavior to specific parts of the content. Examples :- <a>, <strong>, <em>, <span>, <img>, <br>, <input>.

Example:

<!DOCTYPE html>

<html>

<body>

<!--Block element div tag -->

<div> About Me </div>

<!--Inline element anchor tag -->

Here you can check out <a href="https://github.com/tinkerness"

alt="Anitta’s GitHub">Anitta’s GitHub</a>

</body>

</html>

**3. Write HTML code for the following:**

1. **Display an image when a hyper link is clicked**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<div style="padding: 20px;">

<a href="flower.jpg" target="\_blank"><h1>View Image</h1></a>

</div>

</body>

</html>

1. **A large webpage document which contains five headings. Create a hyperlink at the top of the web page which when clicked will scroll the page down to the third heading.**

#code in repo 🡪 headings.html

1. **What are pseudo classes in CSS?**

A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s). For example, the pseudo-class :hover can be used to select a button when a user's pointer hovers over the button and this selected button can then be styled.

Example:

/\* Any button over which the user's pointer is hovering \*/

button:hover {

color: blue;

}

A pseudo-class consists of a colon (:) followed by the pseudo-class name (e.g., :hover). A functional pseudo-class also contains a pair of parentheses to define the arguments (e.g., :dir()). The element that a pseudo-class is attached to is defined as an anchor element (e.g., button in case button:hover).

Pseudo-classes let you apply a style to an element not only in relation to the content of the document tree, but also in relation to external factors like the history of the navigator (:visited, for example), the status of its content (like :checked on certain form elements), or the position of the mouse (like :hover, which lets you know if the mouse is over an element or not).

1. **Compare Absolute Positioning and Relative positioning.**

**Relative** - the element is positioned relative to its normal position. It can be shifted from its original position using properties like top, right, bottom, and left.

* + Relative positioning is used when you want to nudge an element from its original position without affecting the layout of other elements.
  + It's often used for small adjustments to the positioning of elements.

Example:

.relative-element {

position: relative;

top: 10px;

left: -20px;

}

**Absolute** - the element is positioned absolutely to its first positioned parent.

* + Absolute positioning is often used to position elements precisely on a page, such as overlays, tooltips, or pop-up menus.
  + It is also used for creating complex layouts when you want to place elements at specific coordinates.

Example:

.absolute-element {

position: absolute;

top: 50px;

left: 100px;

}

1. **Write down the general format of an HTTP request and an HTTP response. What is the purpose of the following HTTP headers? Also, identify whether they are included with an HTTP header/response or both. i. host ii. last-modified**

**HTTP Request:**

An HTTP request consists of several parts, including the following:

1. **Request Line:** Specifies the HTTP method, the requested URL (Uniform Resource Locator), and the HTTP version. HTTP Method Request-URI HTTP-Version.

HTTP Method Request-URI HTTP-Version

Example:

GET /index.html HTTP/1.1

1. **Headers:** These are optional and provide additional information about the request, such as user-agent, accepted content types, and more. Some common request headers include:
   1. **Host**: Specifies the domain name of the server (e.g., [www.example.com](http://www.example.com/)).
   2. **User-Agent**: Identifies the client (e.g., the browser) making the request.
   3. **Accept**: Specifies the media types accepted by the client (e.g., HTML, JSON).
   4. **Authorization**: Provides authentication credentials if required.
2. **Empty Line:** Separates the headers from the request body. It consists of a carriage return and line feed (**\r\n**).
3. **Request Body:** Contains optional data sent by the client, such as form data in a POST request.

**HTTP Response:**

An HTTP response consists of several parts as well:

1. **Status Line:** Contains the HTTP version, a three-digit status code, and a reason phrase.

HTTP-Version Status-Code Reason-Phrase

Example:

HTTP/1.1 200 OK

1. **Headers:** Provide additional information about the response, such as content type, date, and server information. Some common response headers include:

* **Content-Type**: Specifies the media type of the response body (e.g., text/html, application/json).
* **Content-Length**: Indicates the length of the response body in bytes.
* **Server**: Identifies the server software.

1. **Empty Line:** Separates the headers from the response body.
2. **Response Body:** Contains the actual data or content being sent back to the client, such as an HTML page or JSON data.

**Purpose of HTTP Headers:**

Now, let's discuss the purpose of the specific HTTP headers you mentioned:

i. **Host Header:**

* **Included in:** HTTP request.
* **Purpose:** The **Host** header is used to specify the domain name of the server to which the client is making the request. It is crucial in virtual hosting scenarios, where multiple websites are hosted on the same server with a single IP address. The **Host** header helps the server determine which website or resource the client is requesting.

ii. **Last-Modified Header:**

* **Included in:** HTTP response.
* **Purpose:** The **Last-Modified** header indicates the date and time when the requested resource was last modified on the server. It is commonly used in conjunction with the **If-Modified-Since** request header to implement efficient caching mechanisms. The client can send an **If-Modified-Since** header with the date it received the resource, and the server can respond with a 304 Not Modified status if the resource hasn't changed since that date, reducing unnecessary data transfer.

These headers play crucial roles in the proper functioning of the HTTP protocol and the efficient exchange of data between clients and servers.

**7. Write CSS code for the following:**

1. **set the background color for the hover and active link states to "yellow".**

<a href="#home" class="custom-link">HOME</a>

.custom-link:hover, .custom-link:active {

background-color: yellow;

}

1. **Set the list style for ordered lists to "lower case alphabet".**

ol{

list-style-type: lower-alpha;

}

1. **Set "Boat.gif" as the background image of the page.**

body{

background-image: url('boat.jpg');

background-repeat: no-repeat;

background-size: cover;

}

1. **Set dotted border for the document.**

body{

border: 10px dotted yellow;

}

1. **How are conflicts resolved when there are two different values for the same property on the same element in a document.**

When there are conflicting CSS rules that specify different values for the same property on the same HTML element in a document, the browser uses a set of rules to determine which value should be applied. This process is known as CSS specificity and the cascade. Here's how conflicts are resolved:

1. **Specificity:** CSS rules are assigned a specificity value based on the combination of selectors they use. Specificity determines which rule takes precedence when multiple conflicting rules target the same element. The more specific a selector is, the higher its specificity value.
   * Inline styles have the highest specificity.
   * IDs in selectors have higher specificity than classes, pseudo-classes, and type selectors.
   * The **!important** declaration gives the highest specificity.
2. **Cascade:** The cascade refers to the order in which CSS rules are applied. Rules defined later in the stylesheet or with a higher specificity take precedence over earlier or less specific rules.
3. **Specificity Wins:** When two conflicting rules have the same specificity, the rule defined later in the stylesheet takes precedence. In other words, the "last rule wins."
4. **!important:** If a rule includes the **!important** declaration, it will override any conflicting rule, regardless of specificity or cascade order.

Here's an example to illustrate how these principles work:

/\* CSS Rules \*/

#my-element {

color: blue;

}

p {

color: red !important;

}

/\* HTML \*/

<div id="my-element">

<p>This text is inside a paragraph.</p>

</div>

In this example:

* The #my-element selector has higher specificity because it targets an element with an ID.
* The color: blue; rule for #my-element would normally take precedence.
* However, the p selector sets the color to red with the !important declaration, which overrides the specificity.
* Therefore, the text inside the <p> element will be red because the !important rule takes precedence.

1. **Differentiate between id selector and generic selector in CSS with suitable examples.**

* ID selectors are specific and target a single, unique element with a specific ID attribute value.
* Generic selectors target all elements of a specified type or tag and are less specific.
* ID selectors have higher specificity and take precedence over generic selectors.
* ID selectors should be used for unique elements with specific styles, while generic selectors are used for consistent styles across multiple elements of the same type.
* Example:

**ID Selector:**

/\* HTML \*/

<div id="my-element">

This is a div with a unique ID.

</div>

/\* CSS Rules \*/

#my-element {

background-color: lightblue;

color: white;

}

**Generic Selector:**

/\* HTML \*/

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

p {

font-size: 16px;

line-height: 1.5;

}

1. **What are the uses of text-indent and text-align properties? What are the possible values the properties can take? Demonstrate with example.**

The **text-indent** and **text-align** properties in CSS are used for controlling the alignment and indentation of text within an HTML element.

**text-indent Property:**

* Use: The text-indent property specifies the indentation of the first line of text within an element's content box. It is commonly used for creating paragraphs with an indented first line, such as in body text or quotes.
* Possible Values: **length, percentage, initial, inherit,** and **unset**.

**text-align Property:**

* Use: The text-align property defines the horizontal alignment of text within an element. It is used to control how text content is aligned within its containing element, such as centering text within a div or aligning text within table cells.
* Possible Values: **left, right, center, justify, start, end, initial, inherit,** and **unset**.

**Examples**:

**text-align Property:**

**/\* HTML \*/**

<div class="center-text">

<p>This text is center-aligned.</p>

</div>

<table border="1">

<tr>

<td>This text is right-aligned.</td>

<td>This text is right-aligned.</td>

</tr>

<tr>

<td>right</td>

<td>right</td>

</tr>

</table>

**/\* CSS Rules \*/**

.center-text {

text-align: center;

}

table {

text-align: right;

}

**text-indent Property:**

**/\* HTML \*/**

<p>This is a paragraph with an indented first line. This is a paragraph with an indented first line. This is a paragraph with an indented first line.</p>

<div class="quote">

<p>Quotation: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla nec odio nec tortor ultrices dapibus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla nec odio nec tortor ultrices dapibus.</p>

</div>

**/\* CSS Rules \*/**

p {

text-indent: 20px;

}

.quote {

text-indent: 15%;

}