3: Diving deeper into CSS

**36. display: none vs visibility: hidden**

display: none vs visibility: hidden

We had a look at display: none;  - this value removes the element to which you apply it from the document flow. This means that the element is not visible and it also doesn't "block its position". Other elements can (and will) take its place instead.

There is an alternative to that though.

If you only want to hide an element but you want to keep its place (i.e. other elements don't fill the empty spot), you can use visibility: hidden;

Here's a visual example:

1. .box-1 {
2. display: none;
3. }
5. .box-2 {
6. display: inline-block;
7. }

Will render:

x

where x  has the class box-2 . The first element just isn't displayed. **It's still part of the DOM though**, you can still access it via JavaScript for example.

Here's an example for visibility: hidden :

1. .box-1 {
2. visibility: hidden;
3. }
5. .box-2 {
6. display: inline-block;
7. }

Will render:

\_x

where \_  simply is an empty spot and x  has the class box-2 .

The element is only invisible, **it's not removed from the document flow and of course also not from the DOM.**

**37. HTML Refresher: Block-level vs Inline Elements**

HTML Refresher: Block-level vs Inline Elements

It's not really a CSS topic, though it's related to it: The difference between **block-level** and **inline elements**.

You can read a more detailed article (which also includes a YouTube video about HTML at the top of the page) here: <https://academind.com/learn/html/beginner-s-guide/diving-deeper-into-html#block-level-vs-inline-elements>

*Here's the executive summary:*

**Block-level elements** are rendered as a block and hence take up all the available horizontal space. You can set margin-top and margin-bottom and two block-level elements will render in two different lines.

Some examples are: <div> , <section> , <article> , <nav>  but also <h1> , <h2>  etc and <p> .

**Inline elements**on the other hand only take up the space they require to fit their content in. Hence two inline-elements will fit into the same line (as long as the combined content doesn't take up the entire space in which case a line break would be added).

They also use the box-model you learned about but margin-top  and margin-bottom  have no effect on the element. padding-top  and padding-bottom  also have a different effect. They don't push the adjacent content away but they will do so with the element border. You can read more about that behavior in the following article: <https://hacks.mozilla.org/2015/03/understanding-inline-box-model/>

Additionally, setting a width  or height  on an inline element also has no effect. The width and height is auto to take as much space as required by the content.

Logically, this makes sense since you don't want your inline elements to destroy your multi-line text-layout. If you want to do so or need both block-level and inline behavior, you can set display: inline-block  to merge behaviors.

Some example elements are: <a> , <span> , <img>

38. Applying the Display Property & Styling our Navigation Bar

Section 4: More on Selectors & CSS Features

51. Module Introduction

52. Using Multiple CSS Classes & Combined Selectors

<div class = “class1 class2”>

<a href = “#” class = “active”>

In the above html code there is a div tag which has 2 classes namely class1 and class2 seperated by white space and we can style that div tag with class1 and class2 selectors separately. We can specify class1 and class2 in any order in the div tag. While specifying css properties using class1 and class2 sperators seperately the css properties which are over ridden in the later class sperator will be applied to the div tag.

<a href = “#” class = “active”>

In the above statement active class is specified to the anchor tag. We can apply styling using below formats such as

.active {

…..

}

and

a.active {

…………….

}

You can apply more than one class to an element.

You can chain selectors (e.g. a.active,.priority.highlighted)

Class selectors are the most used type of CSS selectors

53. Classes or Id’s

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>CSS Course</title>

<link rel="stylesheet" href="main.css">

</head>

<body>

<nav>

<a href="#intro" class="active">Intro</a>

<a href="#outro">Outro</a>

</nav>

<section id="intro" class="main-section highlighted">

<p>This is the intro section.</p>

</section>

<section id="outro" class="main-section">

<p>This is the outro section.</p>

</section>

</body>

</html>

In the above html page id’s such intro and outro are added to section tags respectively and inside the nav tag there are anchor tags in href’s are assigned with id names adding # before id names by which if we click on the any anchor tag it will move the page to the corresponding section.

54. (Not) using !important

body {

font-family: sans-serif;

}

nav {

margin-bottom: 16px;

background: #fa923f;

padding: 8px 0;

}

a {

text-decoration: none;

color: white;

margin: 10px;

}

a.active {

color: #521751;

}

.main-section {

height: 800px;

border: 1px solid #ccc !important;

padding: 16px;

}

.highlighted {

border: solid 5px #fa923f;

height: 200px;

}

55. Selecting the Opposite with :not()

a:not(.active) {

……….

}

The above not pseudo class selects all the anchor tags which doesn’t have classname as active and will not select the anchor tag which has classname as active.

56. CSS & Browser Support

Section 5: Practicing the Basics

59. Module Introduction