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

Section 6: Positioning Elements with CSS

80. Module Introduction

81. Why Positioning will Improve our Website

82. Understanding Positioning – The Theory

83. Working with the “fixed” Value

Position property : involves taking the element from the normal document flow in this positioning of an element there are property values such as static, fixed, absolute, relative and sticky. And move the elements we use top, right, bottom, left properties with values.

The default property of Positioning of an element is static. If we apply top or right or bottom or left for static position property there will no change. Top, right, bottom and left will work only for other than static position property. For static position property top, right, bottom and left will not work.

position: fixed;

If we apply the position fixed to an element, then the corresponding element will be out of the normal document work flow and turns into inline block element and hence the next following element will occupy the position of this place and then we can apply top, right, bottom and left properties to move the position: fixed element. The position: fixed element’s position context is not html or body or parent element. Its (position: fixed element’s) position context is view port

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title>Position</title>

</head>

<body>

<div class="parent">

<div class="child-1">Navigation Bar</div>

<div class="child-2">Background Image</div>

<div class="child-3">Features</div>

</div>

</body>

</html>

Main.css file:

html {

background: #b3b3b3;

padding: 15px;

border: 5px solid white;

margin: 15px;

height: 2000px;

}

body {

background: #fa923f;

padding: 20px;

border: 5px solid black;

margin: 0;

}

.parent {

background: white;

padding: 20px;

border: 5px solid black;

margin: 0;

}

.parent div {

background: rgb(105, 105, 109);

color: white;

padding: 10px;

border: 5px solid black;

margin: 10px;

}

.parent .child-1 {

position: fixed;

width: 100%;

margin: 0;

top: 0;

left: 0;

/\* child-1 element will touch to top view port \*/

box-sizing: border-box;

}

Note: 1) We can apply position: fixed property to both inline and block level elements. If we apply position: fixed property to either inline or block level element, then the elements converts into inline-block element.

If we have any margins to html element or body element or any parent element, then the position fixed element will not touch to the viewport for that we need to specify top: 0 and left : 0 explicitly.

If the html element, body element and parent element doesn’t have any margins the position fixed element will automatically touch the top and left view ports implicitly.

2) We can’t apply width property to the inline elements, i.e. if we apply width property to the inline elements there will not be any chane.

**84. Creating a Fixed Navigation Bar**

85. Using “position” to add a Background Image

86. Understanding the Z-index.

z-index is used to arrange the elements along the z axis. This Z-index can be applied only when the position property is applied to the element. After applying the position property to the element then if we apply zindex then only zindex will function otherwise it will not function.

87. Adding a Badge to our package.

Position: absolute;

In the position fixed value the positioning context is view port but in the position absolute value the positioning context depends. If the position absolute element’s ancestors doesn’t have any position property then the position absolute’s positioning context is html element. If the position absolute element’s ancestors has position property then the position absolute element’s position context is its nearest ancestor element having position property.

**88. Styling & Positioning our Badge with “absolute” and “relative”**

**89. Diving Deeper into Relative Positioning**

position: relative;

if we simply apply the position relative property to html element there won’t be any change i.e. the position relative will not move out from document work flow. The position context of relative position is the element itself. If we again apply the properties such as top: 50px and left: 50px then the element moves 50px from top of intial position of element itself and and 50 px towards left of initial position of element itself.

**90. Working with “overflow” and Relative Positioning**

overflow: hidden;

the above property hides the element once the element is outside of the parent.

If the child elements comes out of the parent element then if we want to hide the child elements which came out from the parent element then we use the property called overflow on the parent element and this value is set to hidden then the elements which came out from the parent element will be hidden. Here to be noted is that if the parent element is body in this case if we apply overflow: hidden property to the body and simultaneously the html element doesn’t have overflow property then the child elements which came out of the body (parent) element will not be hidden. In order to hide the child elements from the body the overflow property is set be hidden on the body and overflow is set auto or hidden on the html elements. Then only the child elements which came out from the body element will be hidden.

**91. Introducing “sticky” Positioning**

**92. Understanding the Stacking Context**

**93.**

**94.**

**Section 7: Understanding Background Images & Images**

**95.**

**96. Module Introduction**

**97. Understanding “background-size”**

background-size: cover;

The cover property value of the background size fills the empty space with the image. This property value automatically adjusts the image and fills all the available space with the image.

background-size : contain;

in this background size contain value the image is set to fit in the available space but this contain value will not ensure that it would fill all the available space but ensures to place the complete image in the space and there may be chances of occurrence of white spaces. Whereas in cover property value ensures that the space will be completely filled with the image and it doesn’t ensure to place a complete image in the available space. This cover property value may crop the image inorder to fill the available space with the image.

background-repeat : no-repeat;

the above value doesnot repeat the image.

: repeatx => this value repeats the image only in the x axis.

:repeat => this value repeats the image only in the y axis.

98. Working with “background-position”

background-position: center;

above center property value crops the excess part of the image which is not fitted in the available space cropped equally in all directions.

There is much manual setting involved in this video watch this video once again.

99.

100.