Lecture # 11

Advance Web Development

Ms. Hafiza Alia
(alia@TheProTec.com)

ROBERT W. SEBESTA

CSS Display

CSS Layout - The display Property

- The display property is the most important CSS property for controlling layout.
- Every HTML element has a default display value depending on what type of element it is.
- The default display value for most elements is block or inline.

Type of Elements

- Block-level Elements
  - A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

This is div (block level)

- Inline Elements
  - An inline element does not start on a new line and only takes up as much width as

Span inline

Display none

```
h1.display {
    display: none;
}
```

CSS Layout width and max-width

Example

```
<style>
div.ex1 {
    width:500px;//absolute
    margin: auto;
    border: 3px solid #73AD21;
}
div.ex2 {
    max-width:500px;
    margin: auto;
    border: 3px solid #73AD21;
}
</style>
```

```
<div class="ex1">This div element has
width: 50%;</div>

<br>

<div class="ex2">This div element has max-
width: 500px;</div>
```

CSS Layout Position property

CSS Position

- Values
  1. Static
  2. Fixed
  3. Absolute
  4. Sticky
  5. Relative

Position fixed

- An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.
- The top, right, bottom, and left properties are used to position the element.

position: relative;

- An element with position: relative; is positioned relative to its normal position.
- Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.

position: static;

- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.

Position absolute

- An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
- However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
- Note: A "positioned" element is one whose position is anything except static.

position: sticky;

- An element with position: sticky; is positioned based on the user's scroll position.
- A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like Position: fixed).

Overlapping Elements

- When elements are positioned, they can overlap other elements.
- The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

CSS Layout - Horizontal & Vertical Align an Element

- Center Align Elements
  - To horizontally center a block element (like <div>), use margin: auto;

```
.center {
    margin: auto;
    width: 50%;
    border: 3px solid green;
    padding: 10px;
}
```

Note: Center aligning has no effect if the width property is not set (or set to 100%).

Center Align Text

- To just center the text inside an element, use text-align: center;

Center an Image

- To center an image, set left and right margin to auto and make it into a block element:

```
img {
    display: block;
    margin-left: auto;
    margin-right: auto;
}
```

Left and Right Align - Using position

```
<div class="right">
  <p>Left and Right Align - Using position Left and Right Align - Using position Left and Right Align - Using position Left and Right Align - Using position</p>
</div>

.right {
    position: absolute;
    right: 0px;
    width: 300px;
    border: 3px solid #73AD21;
    padding: 10px;
}
```

Left and Right Align - Using float

```
.right {
    width: 300px;
   float: right;
   border: 3px solid red;
   padding: 10px;
}
```

The clear fix Hack

Example

Discussed in class

Center Vertically - Using padding

- There are many ways to center an element vertically in CSS.
- A simple solution is to use top and bottom padding:

Clear Property

- The clear property can have one of the following values:
  - none - Allows floating elements on both sides. This is default
  - left - No floating elements allowed on the left side
  - right- No floating elements allowed on the right side
  - both - No floating elements allowed on either the left or the right side
  - inherit - The element inherits the clear value of its parent

box-sizing

- The CSS box-sizing property allows us to include the padding and border in an element's total width and height.

- Without the CSS box-sizing Property
  - By default, the width and height of an element is calculated like this:

  - width + padding + border = actual width of an element
  - height + padding + border = actual height of an element

Example

-
  ```
  .div1 {
      width: 300px;
      height: 100px;
      border: 1px solid blue;
  }
  ```

-
  ```
  .div2 {
      width: 300px;
      height: 100px;
      padding: 50px;
      border: 1px solid red;
  }
  ```

- Add box sizing border box to both

Universal Selector

```
* {
    box-sizing: border-box;
}
```

descendant selector (space)/ child selector

```
div p {
   background-color: yellow;
}
```

CSS Pseudo-classes

- What are Pseudo-classes?
  - A pseudo-class is used to define a special state of an element.
- For example, it can be used to:
  - Style an element when a user mouse over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus
- Syntax
  - The syntax of pseudo-classes:

  -
    ```
    selector:pseudo-class {
        property:value;
    }
    ```

Anchor Pseudo-classes

```
/* unvisited link */
a:link {
    color: #FF0000;
}

/* visited link */
a:visited {
    color: #00FF00;
}

/* mouse over link */
a:hover {
    color: #FF00FF;
}

/* selected link */
a:active {
    color: #0000FF;
}
```

Pseudo-classes and CSS Classes

```
a.highlight:hover {
    color: #ff0000;
}
```

Hover on <div>

- An example of using the :hover pseudo-class on a <div> element:

  -
    ```
    div:hover {
        background-color: blue;
    }
    ```

Simple Tooltip Hover

```
p {
    display: none;
    background-color: yellow;
    padding: 20px;
}
div:hover p {
    display: block;
}
```

```
<div>Hover over me to show
the p element

  <p>Tada! Here I am!</p>
</div>
```

CSS - The :first-child Pseudo-class

- The :first-child pseudo-class matches a specified element that is the first child of another element.
- Match the first <p> element

```
p:first-child {
   color: blue;
}
```

Match the first <i> element in all <p> elements

-
  ```
  p  {
        color: blue;
  }
  ```

-
  ```
  li:first-child {
     background: yellow;
  }
  ```

Input:checked Selector

- The :checked selector matches every checked <input> element (only for radio buttons and checkboxes) and <option> element.

```
option:checked{
        height:50px;
        width:50px;
}
```

:disabled

- Option:disabled
- Input:disabled

All pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:empty	p:empty	Selects every <p> element that has no children
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:focus	input:focus	Selects the <input> element that has focus

Pseudo Classes

:hover	a:hover	Selects links on mouse over
:invalid	input:invalid	Selects all <input> elements with an invalid value
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent

End Of Lecture 11