Web Engineering Lab

CSS

LECTURE - 04

Recap

- Table
- Image
- Link
- Form

Course Contents

- Display
- Position
- Overflow
- Z-index
- Float

CSS Display

- The display property is the most important CSS property for controlling layout.
- The display property is used to specify how an element is shown on a web page.
- 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.
- The display property is used to change the default display behavior of HTML elements.
- The display property has various values that influence how elements behave in terms of *size, position, and layout*.

Inline Elements

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

  -
  - <a>
  - <img>
  - <input>
  - <label>
  - <button>
  - <select>
  - <textarea>

Block Elements

- A block-level element ALWAYS starts on a new line and takes up the full width available
  -
  - <h1> - <h6>
  - <p>
  - <form>
  - <header>
  - <footer>
  - <section>
  - <article>
  - <nav>
  - , <ol>,
  - <table>, <tr>, <thead>, <tbody>

Display Property
(Inline, Block, Inline-Block, flex, none)

- Inline
- Block
- **Inline-block:** This combines features of both inline and block. It allows an element to be *laid out inline* (next to other elements), but the element *behaves like a block in terms of width and height* (you can set its width and height explicitly).

```
button {
display: inline-block;
width: 150px;
height: 40px;
}
```

Display Property

- Inline
- Block
- **Inline-block:** This combines features of both inline and block. It allows an element to be *laid out inline* (next to other elements), but the element *behaves like a block in terms of width and height* (you can set its width and height explicitly).

```
button {
display: inline-block;
width: 150px;
height: 40px;
}
```

Display Property

- **none**: An element with *display: none;* will be completely removed from the layout and won't occupy any space on the page. This is useful for hiding elements.
- **flex:** *display:flex;* turns an element into a flex container, enabling flexbox layout. Its child elements (flex items) can then be laid out along a main axis and cross axis with properties like ***justify-content, align-items, and flex-direction***.

```
/* Flexbox Layout */
.flex-container {
display: flex;
justify-content: space-between;
background-color: lightgray;
padding: 10px;
}
```

Position

- The position property specifies the type of positioning method used for an element.

  There are *five* different position values:
  - static
  - relative
  - *fixed*
  - absolute
  - *sticky*
- Elements are then positioned using the *top, bottom, left, and right* properties.
- However, these properties will not work unless the position property is set first.

Position

- **Static**: Default position
- **Relative**:
  - The element is positioned relative to its normal position in the document flow.
  - This means you can move it from its original position using the top, right, bottom, and left properties.

```
.relative-element {
position: relative;
top: 10px;
left: 20px;
}
```

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.

```
div.fixed {
        position: fixed;
        bottom: 0;
        right: 0;
        width: 300px;
        border: 3px solid #73AD21;
        }
```

- Absolute:
- *sticky*

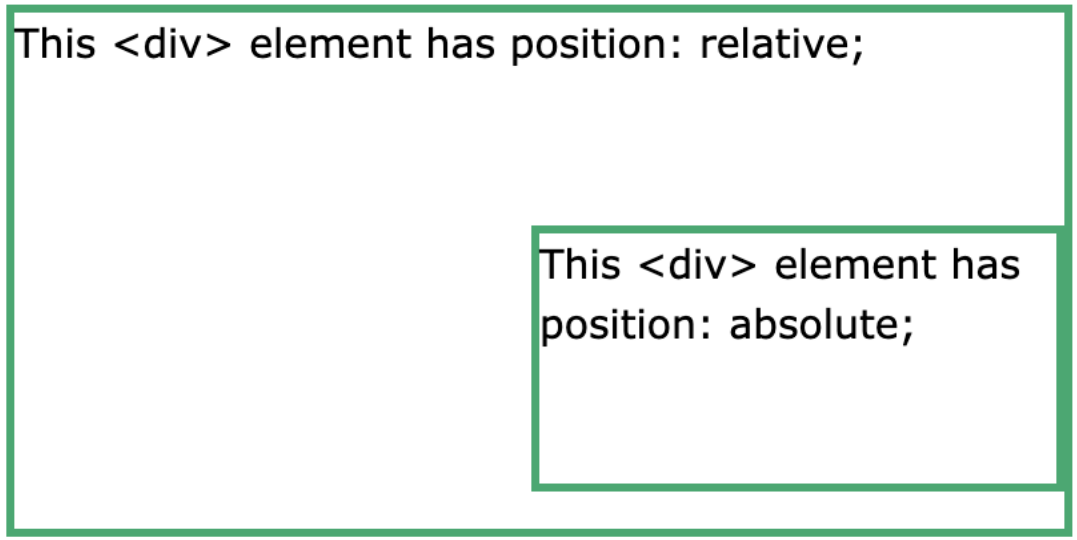
Position

- *Absolute:* An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
-
```
div.fixed {
        position: fixed;
        bottom: 0;
        right: 0;
        width: 300px;
        border: 3px solid #73AD21;
        }
```
- Absolute:
- *sticky*

Position

- *Absolute:* An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
-

```
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```



Position

- *Sticky:* An element with position: sticky; is positioned based on the user's scroll position.

```
div.sticky {
position: sticky;
top: 0;
background-color: green;
border: 2px solid #4CAF50;
}
```

Overflow

- The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.
- The overflow property has the following values:
  - *visible - Default*. The overflow is not clipped. The content renders outside the element's box
  - *hidden* - The overflow is clipped, and the rest of the content will be invisible
  - *scroll* - The overflow is clipped, and a scrollbar is added to see the rest of the content
  - *auto* - Similar to scroll, but it adds scrollbars only when necessary

Overflow

- *Overflow:visible;*
- *Overflow:hidden;*
- *Overflow:scroll;*
- *Auto:scroll;*

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what
happens if content overflows an element's box.

Overflow

- *Overflow:visible;*
- *Overflow:hidden;*
- *Overflow:scroll;*
- *Auto:scroll;*

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what
happens if content overflows an element's box.

Z-index

- The z-index property specifies the stack order of an element.
- 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).
- An element can have a positive or negative stack order:
  - z-index of -1, it will be placed behind other element

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

Float

- The *float property* is used for positioning and formatting content e.g. *let an image float left to the text in a container.*
- The float property can have one of the following values:
  - **left** - The element floats to the left of its container
  - **right** - The element floats to the right of its container
  - **none** - The element does not float (will be displayed just where it occurs in the text). This is default
  - **inherit** - The element inherits the float value of its parent
- In its simplest use, the float property can be used to *wrap text around images.*

End of Lecture

THANK YOU