Basic Web Design

CSS

LECTURE - 12

Course Contents

- CSS Image
- CSS Link
- CSS Form
- Display
- Position

CSS Table

```
table {
  font-family: Arial, Helvetica, sans-serif;   border-collapse: collapse;
width: 100%;

}

 td, th {

  border: 1px solid #ddd; padding: 8px;

}

table tr:nth-child(even){background-color: #f2f2f2;}

table tr:hover {background-color: #ddd;}

table th {

  padding-top: 12px;   padding-bottom: 12px;   text-align: left;  background-
color: #04AA6D;   color: white;

}
```

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Königlich Essen	Philip Cramer	Germany
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy
North/South	Simon Crowther	UK
Paris spécialités	Marie Bertrand	France

CSS Image

- **width and height:** Control the dimensions of the image.
- **border**: Adds a border around the image.
- **border-radius**: Rounds the corners of the image.
- **box-shadow**: Adds shadow effects to the image.
- **object-fit**: Defines how the image should fit within its container.
- Circular Image: ?

CSS Image

```
img {
      width: 300px;
      height: auto;
      border: 5px solid #333;
      border-radius: 15px;
      box-shadow: 5px 5px 15px black;
}
```

CSS Image

- **Circular Image**: creates a circular image by using *border-radius: 50%;* and ensures the image fits within the circle using *object-fit: cover;*
- **Image Hover Effects:** Use (*:hover*) pseudo class.
- **Image Alignment**: Using float, display, margin and text-align;

```
img {
      border-radius: 50%;
      object-fit: cover;
}
```

```
img:hover {
      border: 2px solid black;
}
```

CSS Link (<a>)

- **Links** can be styled using various *pseudo-classes* to change their appearance based on their state (normal, visited, hovered, or active).
- The most common pseudo-classes for styling links are:
  - ***:link*** – Targets an unvisited link.
  - ***:visited*** – Targets a visited link.
  - ***:hover*** – Targets a link when hovered over by the mouse.
  - ***:active*** – Targets a link when clicked

Example

```
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

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

/* selected link */
a:active {
  color: blue;
}
```

Properties for Link

- **Text-decoration**:Controls whether the link is underlined (text-decoration: underline;) or not (text-decoration: none;).
- **Color**: Sets the color of the link text.
- **Background-color**: Use this to create button-like links or enhance the design of links.
- **Border**: Use to create a button

Link Button

```
a:link, a:visited {
        background-color: #f44336;
        color: white;
        padding: 14px 25px;
       text-align: center;
       text-decoration: none;
       display: inline-block;
}

a:hover, a:active {
       background-color: red;
}
```

CSS Form

- Initially a form is designed as a section and each element inside a form is designed seperately.
- A specific input type can be designed using attribute selectors. Foe example:
  - input[type=text] - will only select text fields
  - input[type=password] - will only select password fields
  - input[type=number] - will only select number fields

CSS Form Example

```
input[type=text] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  box-sizing: border-box;
}
```

- **Box-sizing**: Most often we use *box-sizing* property to *border-box*. This makes sure that the padding and eventually borders are included in the total width and height of the elements.

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:** 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

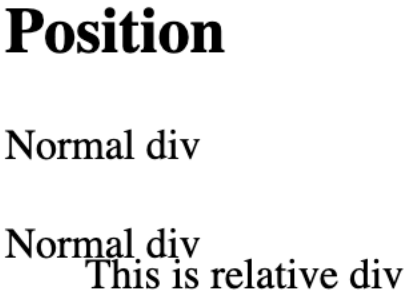
- The position property specifies the type of positioning method used for an element.

  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: 30px;
left: 30px;
}
```



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*

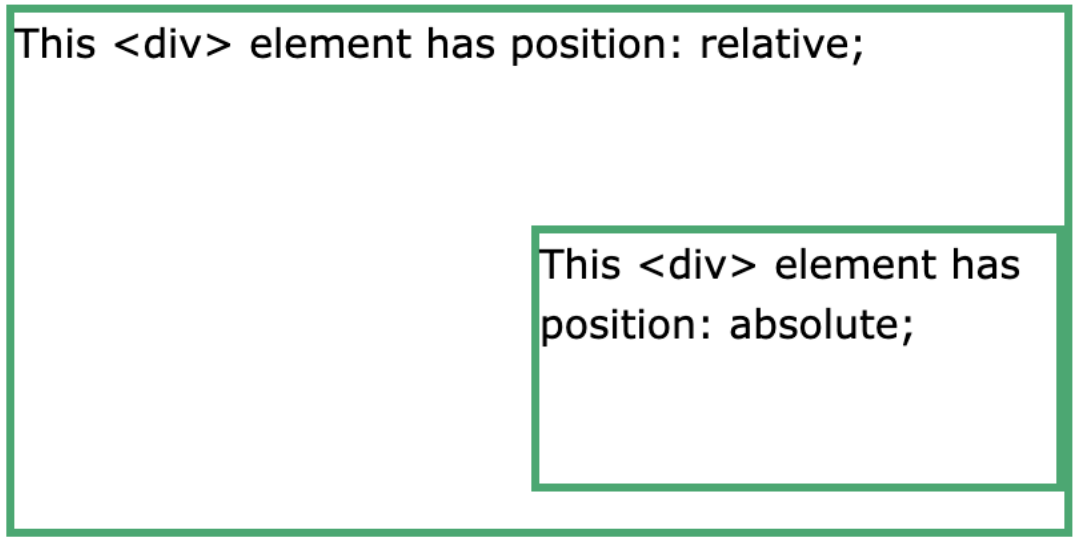
Position

- *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.

Class Work

- Design your complete the task from mid exam.
- Create an image gallery as described in the class.

End of Lecture

THANK YOU