
CSS
IS
AWESOME


hotel-room.css
.air-conditioner {
margin-left: -25px;
}

Module 3-2

CSS Selectors

Objectives

- Normal flow of positioning of elements on a page
- Block, inline, and inline-block
- General CSS Selectors
- Specificity and cascading to apply styles
- Box model
- Discuss different positioning styles
- Common units of measure

Default element positioning

- By default, element laid out as they appear in source code
  - Some will take the entire width (Block elements)
  - Some will appear next to each other (inline)

```
<p>This is an inline span <span style="border: 1px solid black">Hello World</span> element inside a paragraph.
</p>
```

This is an inline span Hello World element inside a paragraph.

```
<div style="border: 1px solid black">Hello World</div>

<p>The DIV element is a block element, and will always start on a new
(stretches out to the left and right as far as it can).</p>
```

Hello World

The DIV element is a block element, and will always start on a new line and take up the full width available (stretches out to the left and right as far as it can).

HTML Elements: Inline vs Block vs Inline-Block

- HTML elements are also classified as being inline vs block.
  - **Inline**: Does not start on a new line
  - **Block**: Starts on a new line
  - **Inline-Block**: similar to inline but you **can** set the height and width

- Common inline elements: a, img, span
- Common block elements: p, div, table
- Common inline-block elements: select, button

Let’s Code

HTML Elements: The Box Model

- HTML content that have been annotated with tags **are known as HTML elements**.
- All elements come with a margin, border, and padding, this is referred to as the box model.

Image - Image of Box Model

Box Model demo

CSS Selectors

- Element selectors
- Class selectors
- Id Selectors
- Advanced Selectors
  - Universal
  - Attribute
  - Pseudo-class
- Combinators
  - Combo
  - Multiple
  - Descendant
  - Child

CSS Selectors

- CSS uses selector to determine which HTML elements will be “targeted” or selected to have a specific format.
- Generally speaking, there will be a CSS block that looks like so:

```
[SELECTOR] {

    [property] : [property value]

}
```

- We will start discussing three most important types of selectors in the next section.

CSS Selectors: By Element

Example

```
div {

        color : red;
}
```

A valid HTML element type is used, in the example to the left, the DIV type.

What this code does: Finds all HTML elements that are 's and applies the formatting, which is to make all the enclosed text red.

CSS Selectors: By Class

Example

```
.warning {
        color : red;
}
```

The dot is required for selection by class.

Anytime you see a dot, it is selection by class!

A valid class name is used, in the example to the left, any element of class “warning.”

```
<div class=”warning”>...</div>
<p class=”warning”>...</p>
```

What this code does: Finds all HTML elements that have the class specified. In other words find all HTML elements that are enclosed by any tag with an attribute class=”warning”.

CSS Selectors: By ID

Example

```
#demo {
        color : red;
}
```

The # sign is required anytime you do selection by ID.

Anytime you see a #, it is selection by id!

A valid HTML element type is used, in the example to the left, the id of type “demo.”

```
<div id=”demo”>...</div>
```

What this code does: Finds the HTML element that has an attribute id=”demo” and apply the format.

NOTE: Id’s should be unique!

CSS Selectors: ID takes precedence over Class

Example

```
#demo {
        color : red;
}

.someId {
        color: blue;
}
```

If there is a HTML element has both an id of someId and a class of demo it will appear red.

```
<p id=”someId” class=”someId”>...</p>
```

The Id attribute takes precedence over the class attribute.

Advanced Selectors: Universal and Attribute

Example

```
* {
  text-align: center;
  color: blue;
}
```

Selects every element on a page. Useful for default values.

```
/* <a> elements with a title attribute */
a[title] {
  color: purple;
}
```

Matches elements based on the presence or value of a given attribute.

Advanced Selectors: Pseudo-class

Example

```
/* Any button over which the user's pointer is hovering */
button:hover {
  color: blue;
}
```

Keyword added to selector that specifies a special state of the element.

```
/* 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;
}
```

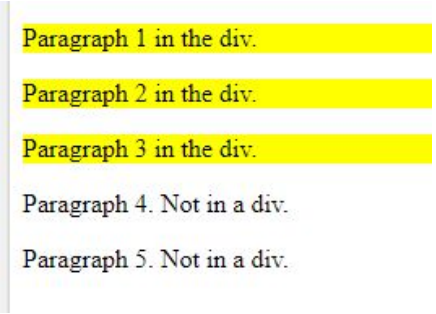
Combinators

- Explains relationship between selectors
  - Descendant selector (space)
  - Child selector (>)

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

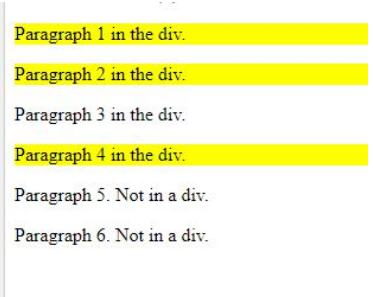
```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>
```




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

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section> <!-- not Child but Descendant -->
  <p>Paragraph 4 in the div.</p>
</div>
```



Combinators

- Explains relationship between selectors
  - Adjacent sibling selector (+)
  - General sibling selector (~)

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

```
<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
</div>

<p>Paragraph 3. After a div.</p>
<p>Paragraph 4. After a div.</p>

<div>
  <p>Paragraph 5 in the div.</p>
  <p>Paragraph 6 in the div.</p>
</div>

<p>Paragraph 7. After a div.</p>
<p>Paragraph 8. After a div.</p>
```

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. After a div.

Paragraph 4. After a div.

Paragraph 5 in the div.

Paragraph 6 in the div.

Paragraph 7. After a div.

Paragraph 8. After a div.

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

```
<div>
  <p>Paragraph 2.</p>
</div>

<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>
```

Paragraph 2.

Paragraph 3.

```
Some code.
```

Paragraph 4.

HTML Elements: Positioning

- All elements have a default flow, a position they will fall into in the absence of additional instructions. This is known as “static” flow. There are additional defined positions:
  - **static**: “default” value. Put an element in the page normally

```
<style>
div.static {
  position: static;
  border: 3px solid #73AD21;
}
</style>
```

This div element has position: static;

  - **relative**: “relative” to what it would be positioned per the normal flow. (Hard to explain, we’ll do an example)

```
<style>
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
</style>
```

An element with position: relative; is positioned relative to its normal position:

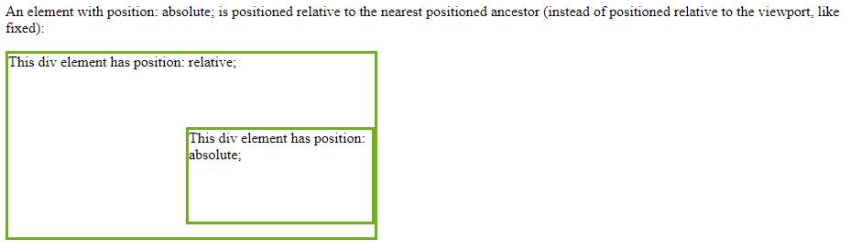
This div element has position: relative;

HTML Elements: Positioning

- All elements have a default flow, a position they will fall into in the absence of additional instructions. This is known as “static” flow. There are additional defined positions:
    - **absolute**: positioned relative to its ancestor.

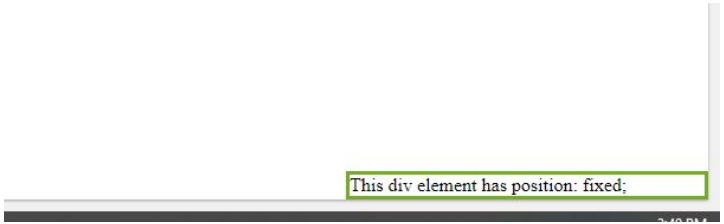
```
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}

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



    - **fixed**: positioned relative to your screen, will follow you as you scroll.

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



HTML Elements: float

- Property specifies if the element should be taken from normal flow and placed along the left or right side of a container.

    - **none**: element does not float
    - **left**: element floats to the left of the container
    - **right**: element floats to the right of the container
    - **inherit**: element inherits the float direction of its parent

    Floated elements automatically display as block. Floats should be avoided in modern web apps and replaced with Flexbox instead (we will discuss later this week)

Units of Measurement

- Absolute
  - Fixed to physical length
  - Pixels - px
- Relative
  - “Relative” to something else – will scale as your page scales
  - Em – 1em is current font size (default is 16px)
  - Rem – root em – relative to root element (typically <html>)
  - % - Percentage – relative to parent element

Let’s Code!