Module 3 - Lecture 2

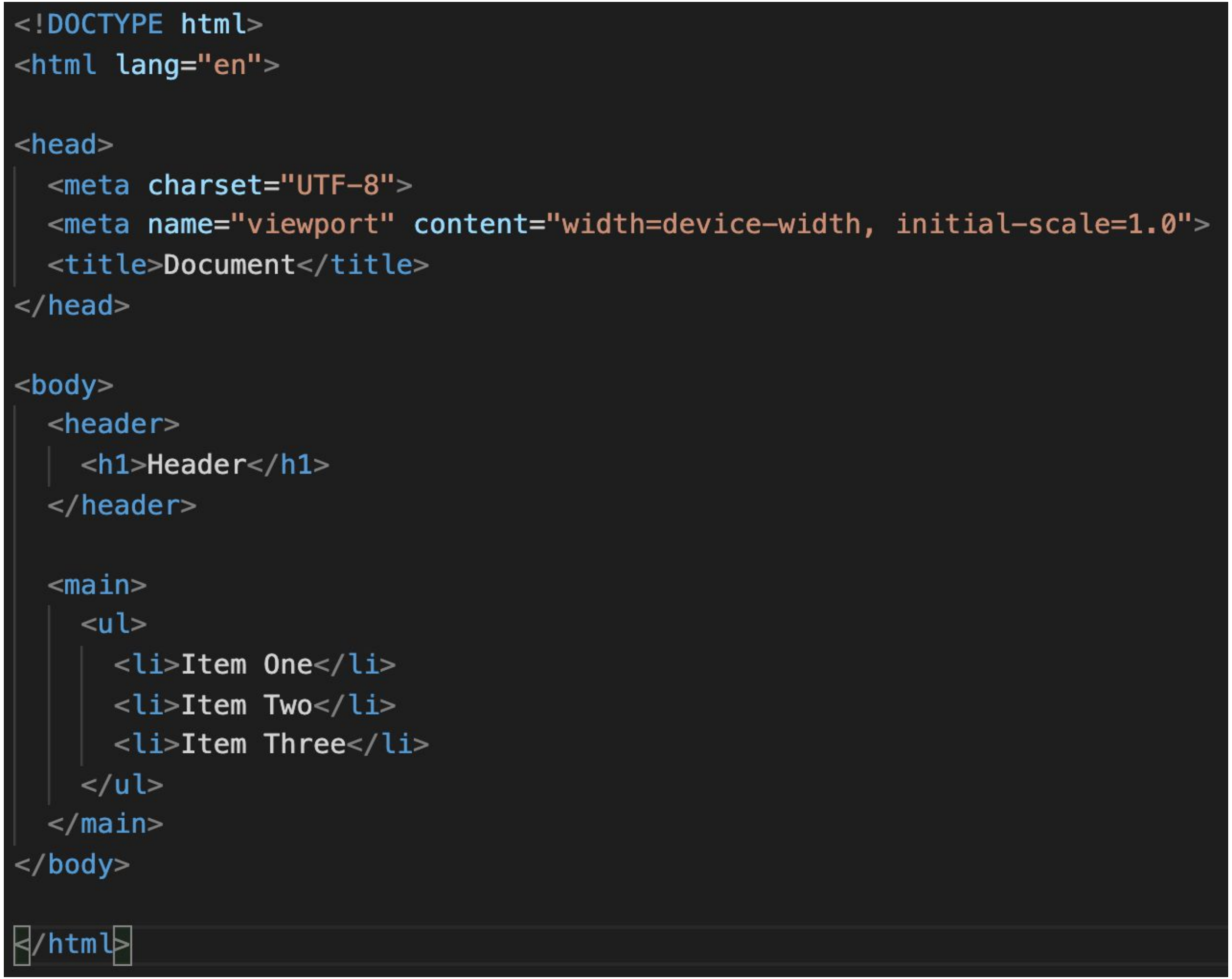
# CSS Selectors and Layout

# CSS: Selectors

<ul> element is parent of <li> elements and child of <main>

<li> elements are children of <ul>

<li> elements are siblings of each other



```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <header>
    <h1>Header</h1>
  </header>

  <main>
    <ul>
      <li>Item One</li>
      <li>Item Two</li>
      <li>Item Three</li>
    </ul>
  </main>
</body>

</html>
```

# HTML id attribute

- A unique, case-sensitive name to identify an element.
- Enables CSS and JavaScript to select elements directly.
- Anchor elements can navigate directly to another element by id.

```
<a href=”#paraObviousPoint”>Link to p</a>


<p id=”paraObviousPoint”>
    Above point sounds a bit obvious.
    Remove/rewrite?
</p>
```

# HTML class attribute

- A space separated list of case-sensitive class names for an element.
- Enables CSS and JavaScript to select elements that share a class name.

```
<p class="note editorial">
   Above point sounds a bit obvious.
   Remove/rewrite?
</p>
```

# CSS Selector Types

| Name | Syntax | Example | Description |
|---|---|---|---|
| All / Wildcard | * | * | Every element |
| Element | element | h1 | All h1 elements |
| ID | #elementId | #application | Element with the id “application” |
| Class | .className | .btn-primary | Elements with the class “btn-primary” |
| Descendant | element  element | main li | li elements that are descendants of a main element |
| Child | element > element | ul > li | li elements that are children of a ul element |
| Attribute | [attribute=value] | [id=application] | Elements with an attribute id=”application” |

# Pseudo-Class Selectors

- Targets an element based on an element’s state. Prefaced with a colon.

Some examples:

a:visited -> anchor elements that have been visited

input:disabled -> input elements that are currently disabled

table tr:nth-child(even) -> even numbered rows in a table

Ref:
https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

# Combining CSS Selectors Using Boolean Logic (AND)

- Selectors can be combined together to indicate multiple selectors have to be true (AND)

**#content div > ul.list**

- SELECT a ul element with class name “list”
- AND the ul element is a child of a div element
- AND the div element is a descendant of an element with an id “content”

# Combining CSS Selectors Using Boolean Logic (OR)

- Selectors can be combined together with a comma to indicate this selector or that selector have to be true (OR)

**div, span, #content**

- SELECT div elements, span elements, or an element with id “content”

# Specificity

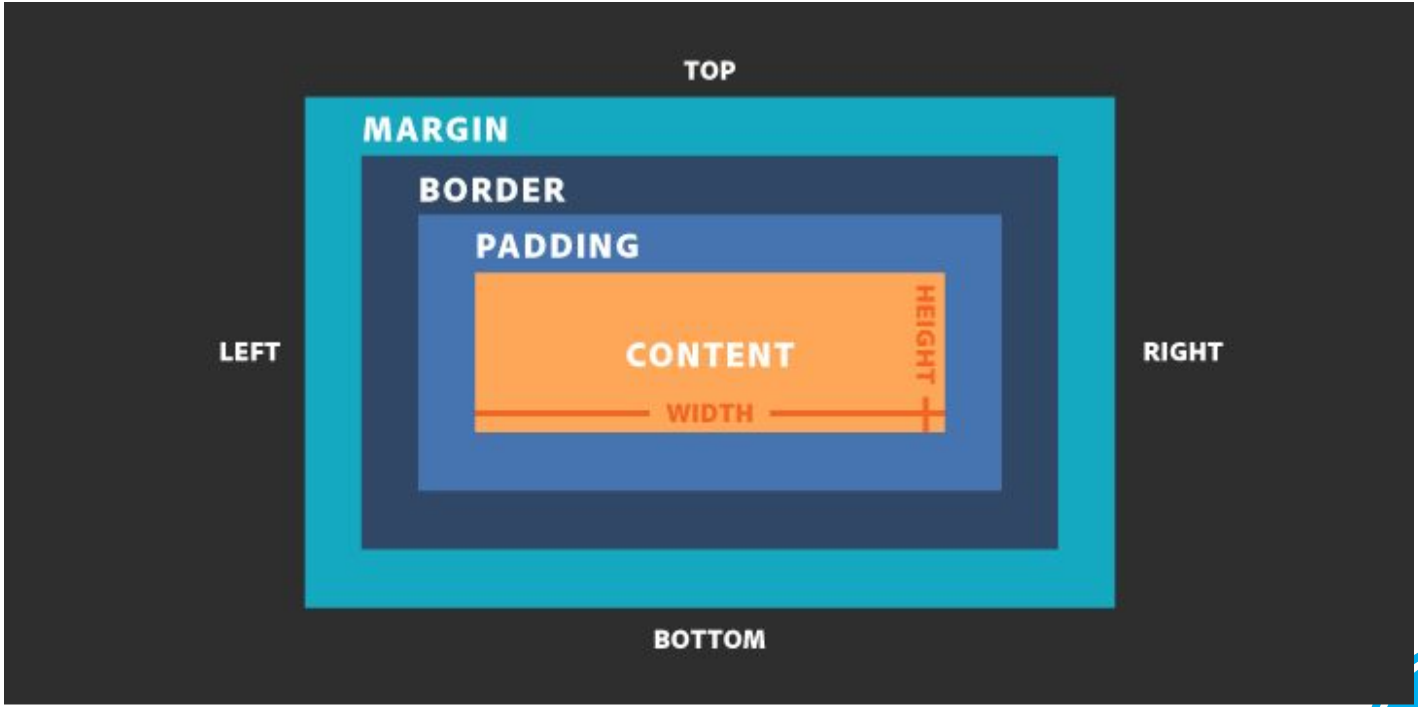
- The specificity algorithm is basically a three-column value of three categories or weights - ID, CLASS, and TYPE - corresponding to the three types of selectors. The value represents the count of selector components in each weight category and is written as *ID - CLASS - TYPE*. The three columns are created by counting the number of selector components for each selector weight category in the selectors that match the element.

- Inline styles have the highest level of specificity.

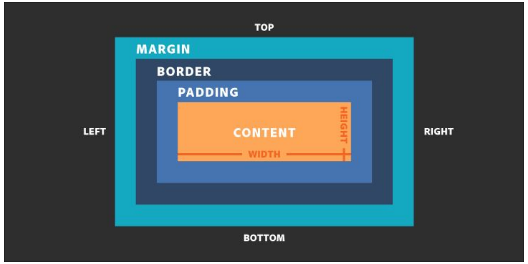
- Applying **!important** to a selector be applied regardless of cascading.
    - Best to avoid using this!

# CSS: Layout

# CSS Box Model



# CSS Box Model



- Every element in a browser is rendered as a box.

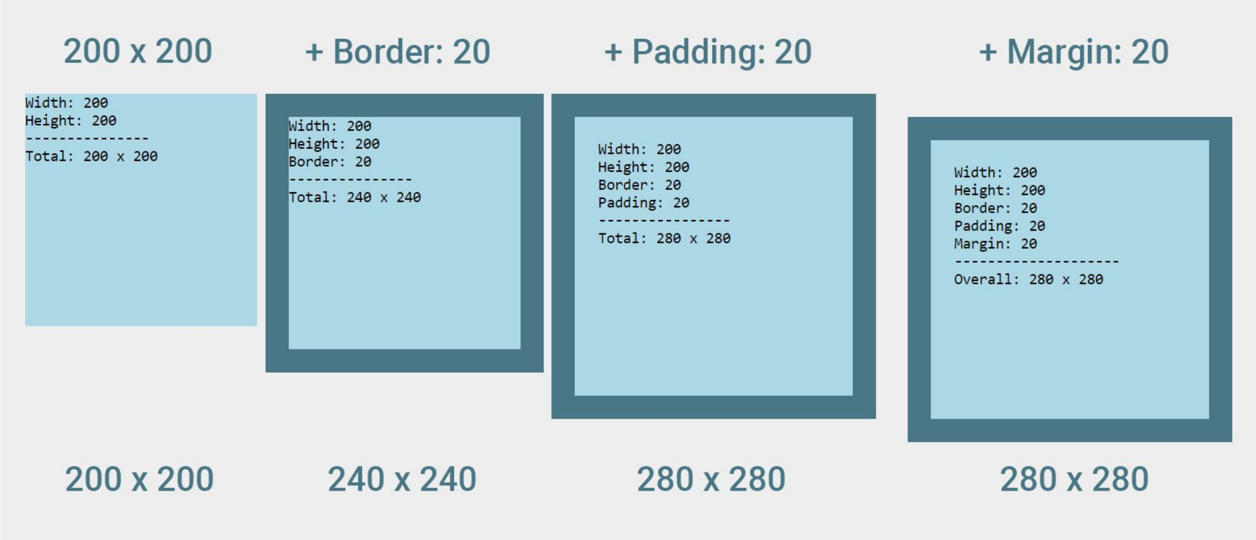
- Every element is made up of content, padding, border, and margin.

- Margin, Border, and Padding each have 4 component widths, a top, right, bottom, left width (think clockwise from the top).

- When adjusting the width and height of an element you are adjusting the width and height of the CONTENT only. Margin, padding, and border sizes are independent.

- **This can be altered using the box-sizing property.** border-box sizing includes padding and border when calculating the content size.

context-box



200 x 200

+ Border: 20

+ Padding: 20

+ Margin: 20

```
Width: 200
Height: 200
---------------
Total: 200 x 200
```

```
Width: 200
Height: 200
Border: 20
---------------
Total: 240 x 240
```

```
Width: 200
Height: 200
Border: 20
Padding: 20
----------------
Total: 280 x 280
```

```
Width: 200
Height: 200
Border: 20
Padding: 20
Margin: 20
--------------------
Overall: 280 x 280
```

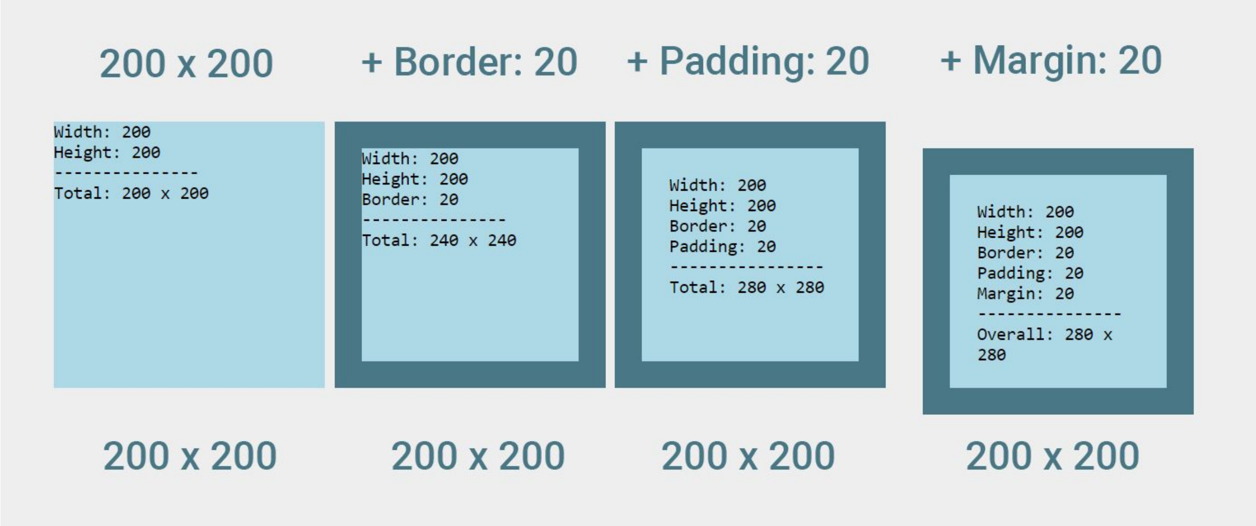
200 x 200

240 x 240

280 x 280

280 x 280

border-box



200 x 200

+ Border: 20

+ Padding: 20

+ Margin: 20

```
Width: 200
Height: 200
---------------
Total: 200 x 200
```

```
Width: 200
Height: 200
Border: 20
---------------
Total: 240 x 240
```

```
Width: 200
Height: 200
Border: 20
Padding: 20
---------------
Total: 280 x 280
```

```
Width: 200
Height: 200
Border: 20
Padding: 20
Margin: 20
---------------
Overall: 280 x
280
```

200 x 200

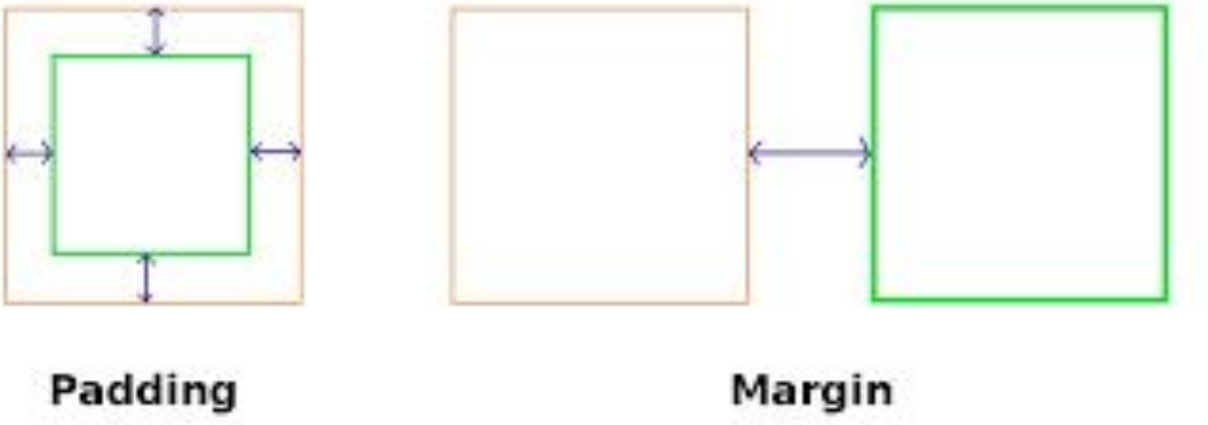
200 x 200

200 x 200

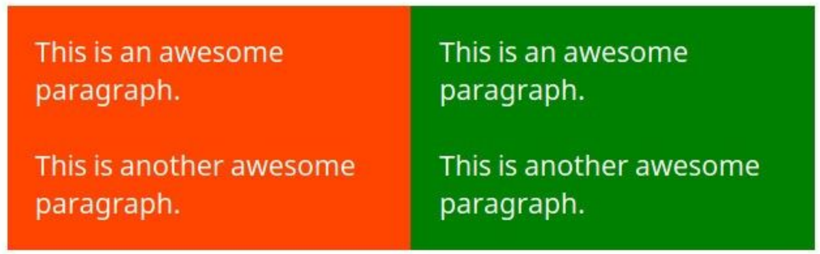
200 x 200

# Padding vs. Margin

- Use margin to separate the box from things outside it
- Use padding to move the contents away from the edges of the box.



# Padding vs. Margin

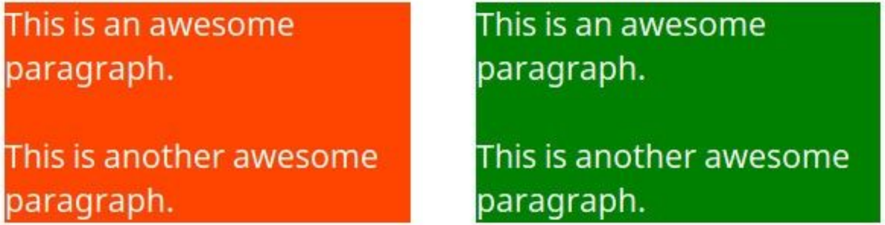


This is an awesome paragraph.

This is another awesome paragraph.

This is an awesome paragraph.

This is another awesome paragraph.

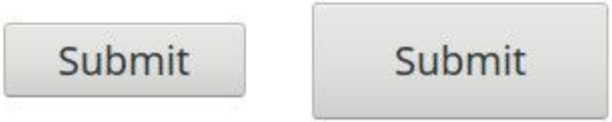


This is an awesome paragraph.

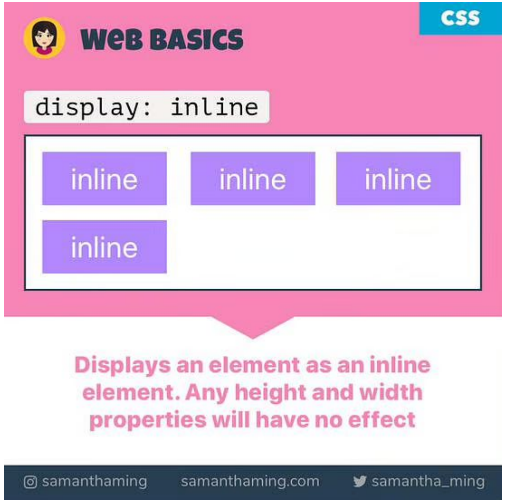
This is another awesome paragraph.

This is an awesome paragraph.

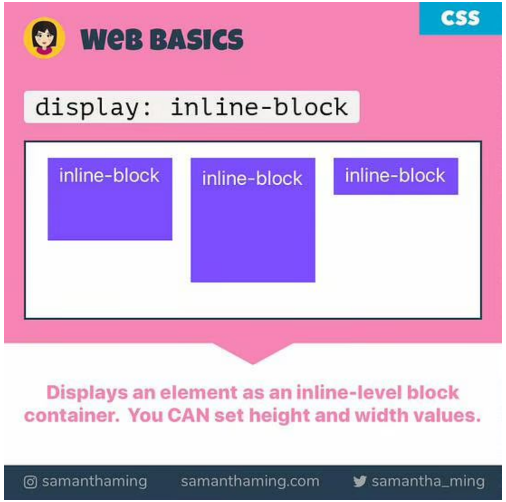
This is another awesome paragraph.



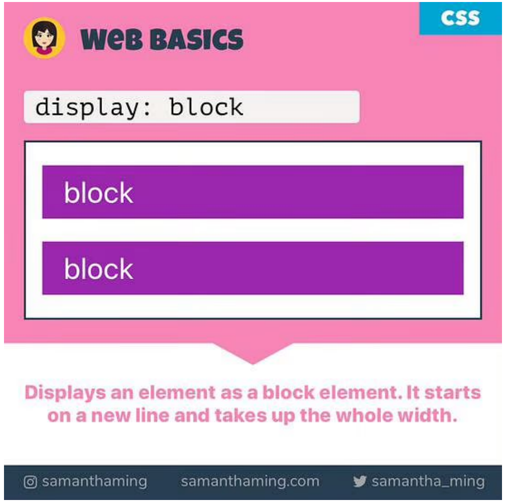
# Inline vs Block



Displays an element as an inline element. Any height and width properties will have no effect



Displays an element as an inline-level block container. You CAN set height and width values.



Displays an element as a block element. It starts on a new line and takes up the whole width.

**Default of inline***

- span
- a
- img

**Default of block***

- div
- h1
- p
- section
- li

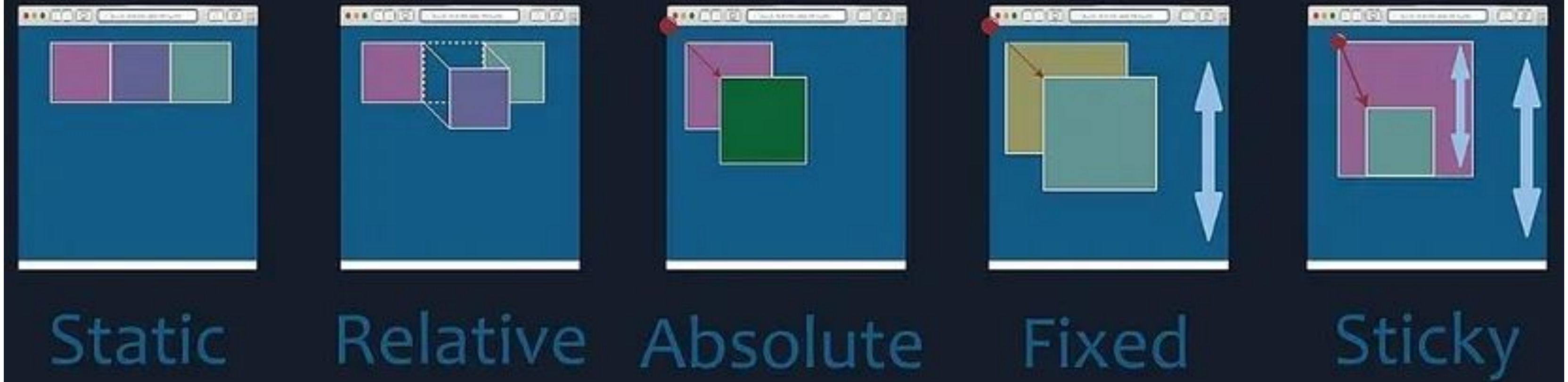
*Not exhaustive lists

# Position

- The **normal flow** of a page is for elements to appear left to right and top to bottom based on the order in which they appear in the HTML document and the rules of block and inline display. This is also referred to as **Static**.

- There are options for other types of positioning. **Relative**, **Absolute**, **Fixed,** and **Sticky**.
- With relative, absolute, fixed, and sticky you can adjust the position of an element by using css properties **top, left, bottom,** and **right** and supplying how many pixels to offset from each.

# CSS Position Property



# Relative Sizing

**FONTS**

- **em** and root em (or **rem**) are sizing measurements relative to the font size.

**ELEMENTS**

- Sizing of elements can also be done in **percentages**, from **0-100%**
  - This is not the same as viewport sizing. A percentage is based on an element’s parent.
- CSS3 introduced a unit of sizing that enables sizing relative to the height and width of the viewport (browser window).
  - The unit is **vh for viewport height** and **vw for viewport width.** Each ranges from 0 - 100, meaning 0 to 100% of the viewport.

# QUESTIONS?