HTML – Interview Question of the Day

Explain what "Semantic HTML" means.

Use the example of <i> for italic and for empahsis to explain semantic html. Both the tags will effectively *italicize* the chosen word.

IO Lab

Spring 2016

Module 2 - Lec 2 - CSS

Jan 29, 2016

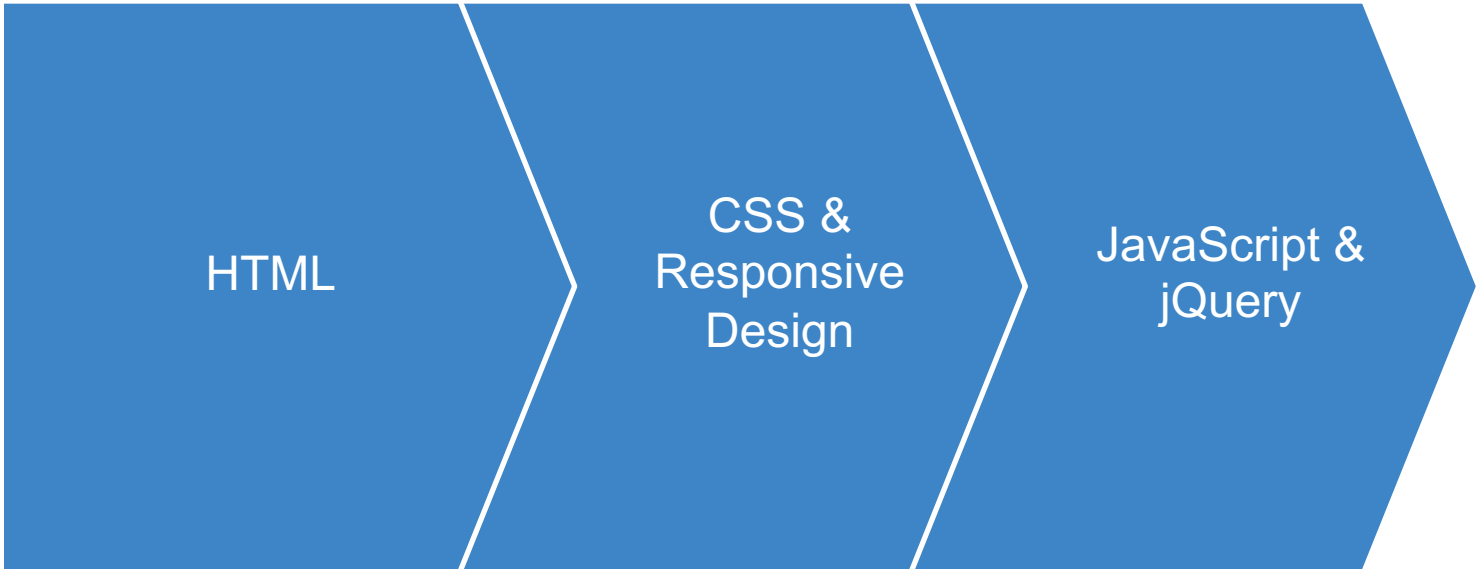
Course Map

Getting Started	Front End	Back End	Advanced Topics
			
Introduction	HTML	Web Frameworks	APIs
Fundamentals	CSS	Architecture	Visualization
Git	Responsive Design	Database Design	Security
Command Line	JavaScript & jQuery		Deployment

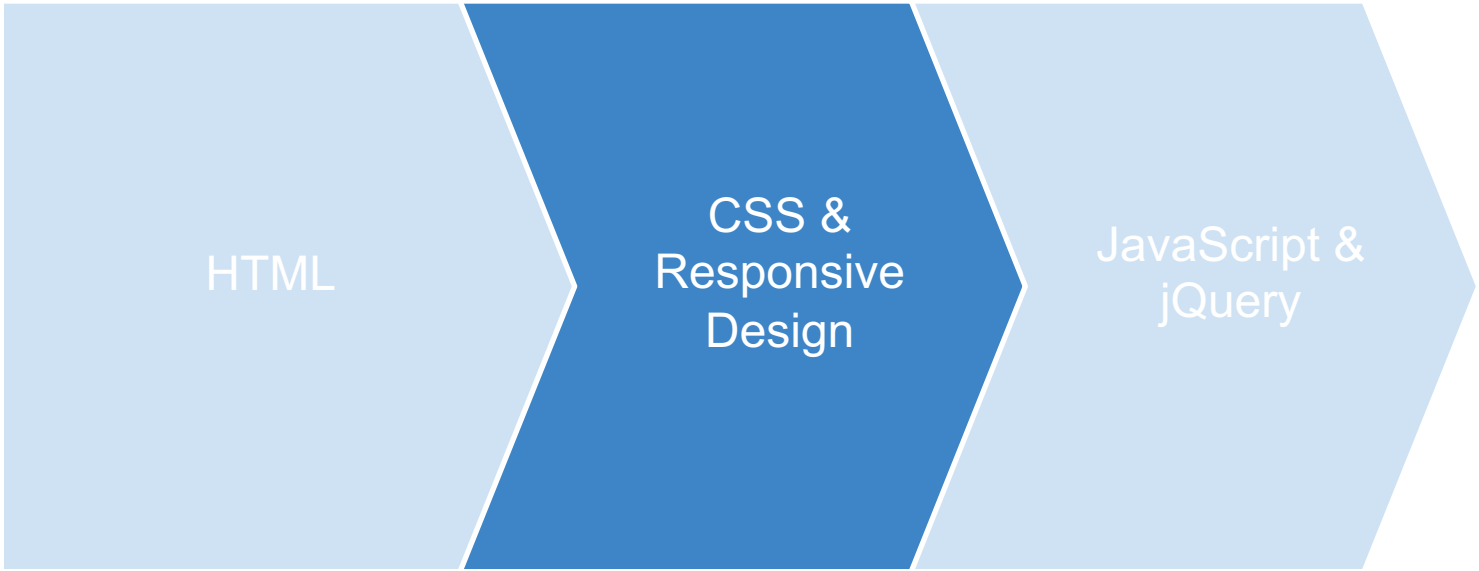
Course Map

Getting Started	Front End	Back End	Advanced Topics
			
Fundamentals	HTML	Web Frameworks	APIs
Git	CSS	Architecture	Visualization
Command Line	Responsive Design	Database Design	Security
	JavaScript & jQuery		Deployment

Front End Module



Front End Module - CSS



CSS

Cascading

Style

Sheets

Architectural Thinking

If architected correctly, the same underlying content can be presented in any number of ways using CSS

Check Out [CSS Zen Garden]

CSS - A Set of Appearance Rules

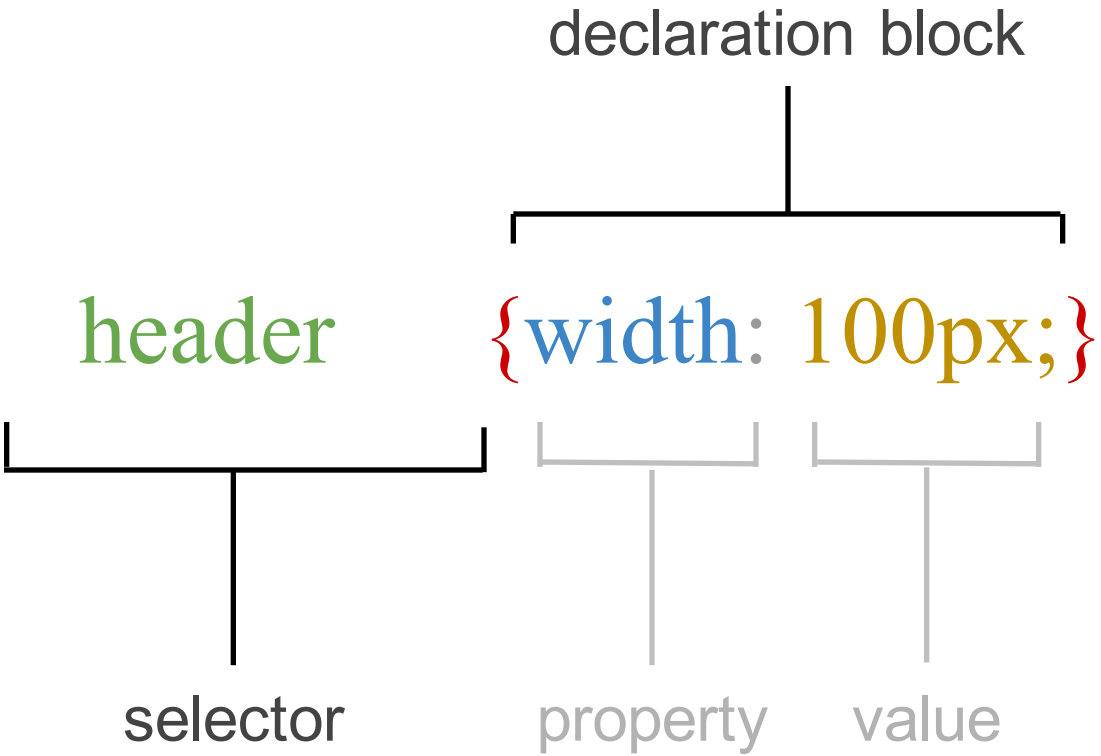
Layout & Positioning

Sizing & Spacing

Colors & Fonts

Animation

CSS - Syntax


declaration block
header
{width: 100px;}
selector
property
value

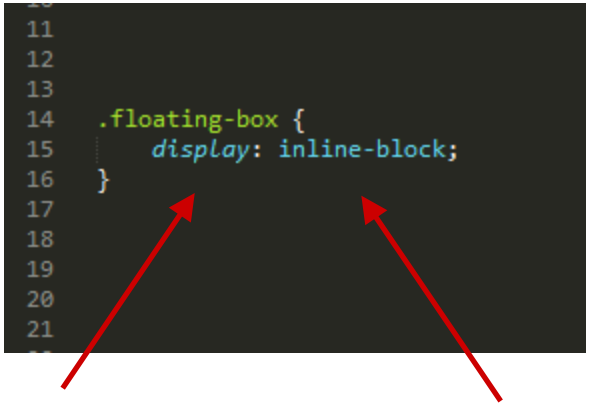
CSS - Getting a Handle with Selectors

- Selectors target individual or multiple elements for styling

Selector	Meaning	Example
Type Selector	Matches all element types	**h1, h2, h3 { }** Targets the <h1>, <h2>, and <h3>
Class Selector	Matches an element whose class attribute has a value that matches the one specified after the period (or full stop) symbol	**.note { }** Targets any element whose class attribute has a value of note **p.note { }** Targets only <p> elements whose class attribute has a value of note
Id Selector	Matches an element whose id attribute has a value that matches the one specified after the pound or hash symbol	**#introduction { }** Targest the element whose id attribute has a value of introduction

CSS - Properties & Values

Once you've targeted one or more elements with a selector, you can style the element(s) by setting the values of their properties



Here, we're selecting the 'display' property to style

Here, we're setting the 'display' property's value to 'inline-block'

CSS – The Last Rule… Rule

- Cascade of files
  - Later files loaded in HTML will take precedence over earlier files
- Cascade of declarations
  - Later selectors in CSS file will take precedence over earlier selectors. Unless a more specific selector is used - more on this in a moment
- Cascade of properties within declaration
  - Later declared property-values take precedence over earlier declared property-values

CSS - Cascade of Declaration Example

```
14  .floating-box {
15      display: inline-block;
16  }
17
18  #spinner {
19      width: 40px;
20      margin-left: auto;
21      margin-right: auto;
22      margin-top: 48%;
23  }
24
25  #loading-mask {
26      position: absolute;
27      top: 0;
28      left: 0;
29      height: 539px;
30  }
31
32  .floating-box {
33      display: block;
34      width: 100px;
35      height: 90px;
36      display: inline;
37  }
38
```

The properties declared in the second .floating-box declaration will override any corresponding properties declared in the first .floating-box declaration

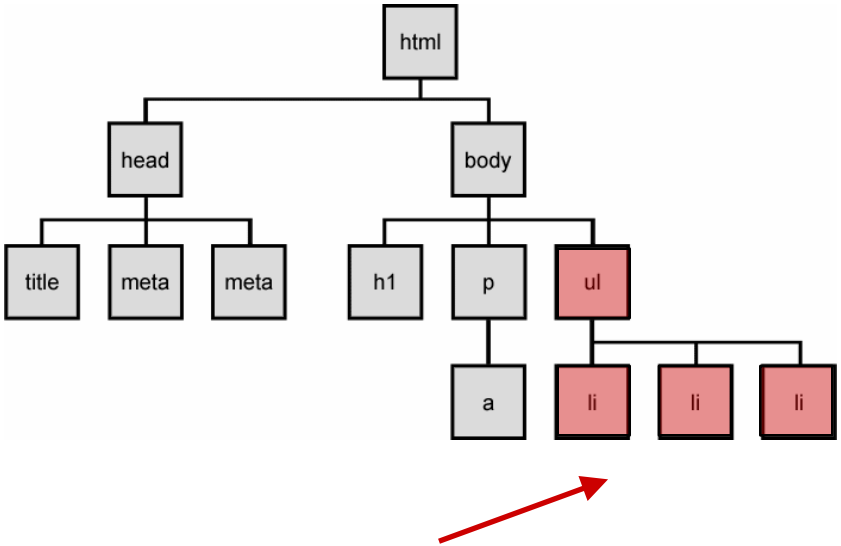
CSS - Cascade of Properties Example

```
14  .floating-box {
15      display: inline-block;
16  }
17
18  #spinner {
19      width: 40px;
20      margin-left: auto;
21      margin-right: auto;
22      margin-top: 48%;
23  }
24
25  #loading-mask {
26      position: absolute;
27      top: 0;
28      left: 0;
29      height: 539px;
30  }
31
32  .floating-box {
33      display: block;
34      width: 100px;
35      height: 90px;
36      display: inline;
37  }
38
```

The second display property declaration will override the first display property declaration

CSS - Inheritance

- Separate from the Cascade
- Similar to genetics
  - Parent passes on traits to children
  - In CSS, parent elements *usually* pass on properties to child elements
  - Not all properties, such as background-image are inherited



In this case, if we set the font-size of the ul element to 1.5 em, the font size of the li children (and their children) would also be 1.5 em

More on inheritance here

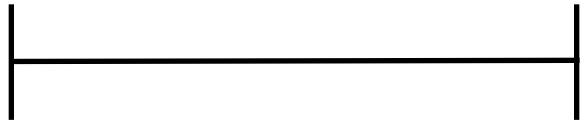
List of CSS inheritance values

CSS - Specificity

More-specific selectors will override cascade precedence

In other words, no matter the order in the cascade:

element type < class < ID



less specific

more specific

Example

```
* {
  font-family: Arial, Verdana, sans-serif;}
h1 {
  font-family: "Courier New", monospace;}
i {
  color: green;}
i {
  color: red;}
b {
  color: pink;}
p b {
  color: blue !important;}
p b {
  color: violet;}
p#intro {
  font-size: 100%;}
p {
  font-size: 75%;}
```

CSS - The Display Property

- Important because it determines how elements are rendered
- Four common values:
  - Block
  - Inline
  - Inline-Block
  - None (prevents element & children from rendering)

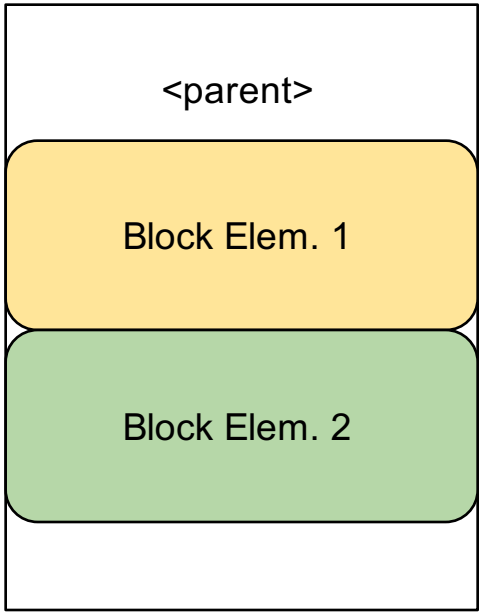
CSS - Block Elements

Take up width of parent container by default

Start on new line

Can wrap around other block or inline elements

div, h1-h6, p, & form are block by default



CSS - Inline Elements

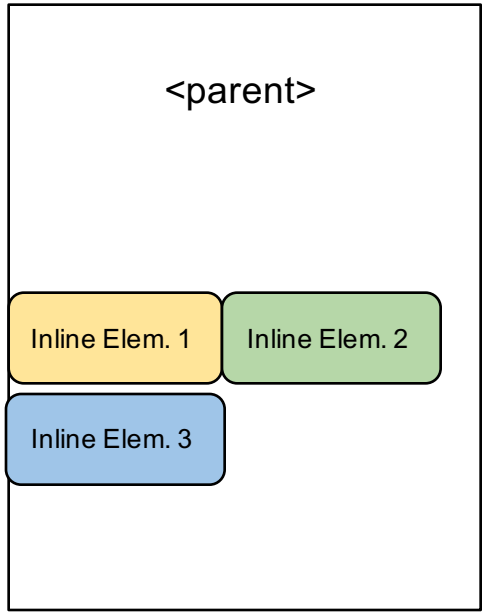
Do not accept width, height, or vertical margins

Width & height based on content

Positioned side-by-side

Except for anchor elements, should not wrap around block elements

span, a, & img are inline by default

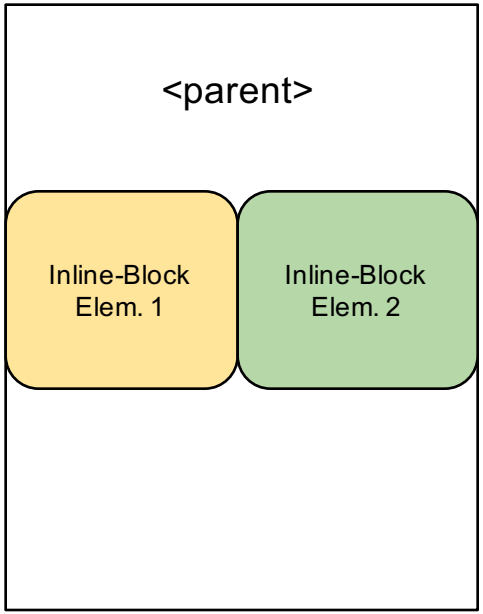


CSS - Inline-Block Elements

Can have a specified width & height, like block elements

Can be on same line as each other, like inline elements

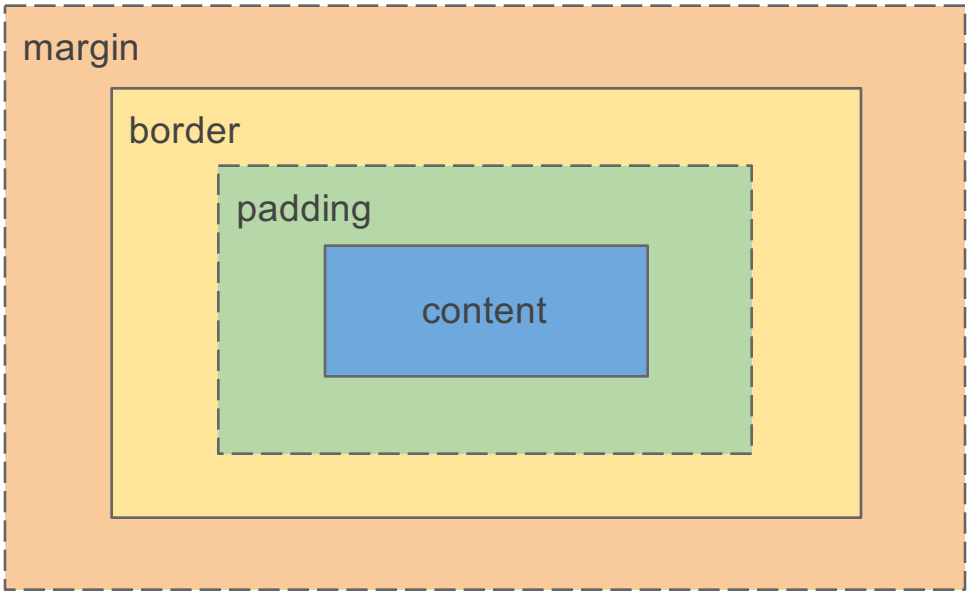
Great for creating grids



CSS - The Box Model

- Every html element is treated as a rectangular box

- Each box is made up of four parts, which can be styled
  - Content (Width & Height)
  - Padding
  - Border
  - Margin

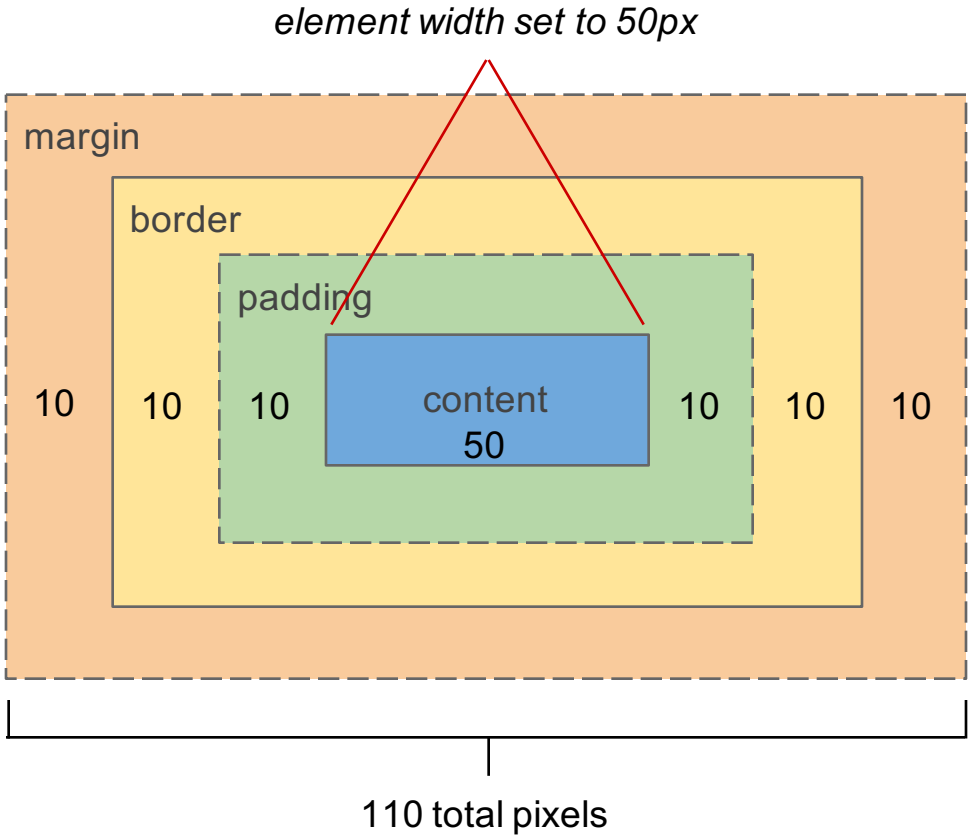


CSS - 3 Different Box Types

- content-box
  - default box type
- padding-box
- border-box
  - generally easiest to work with

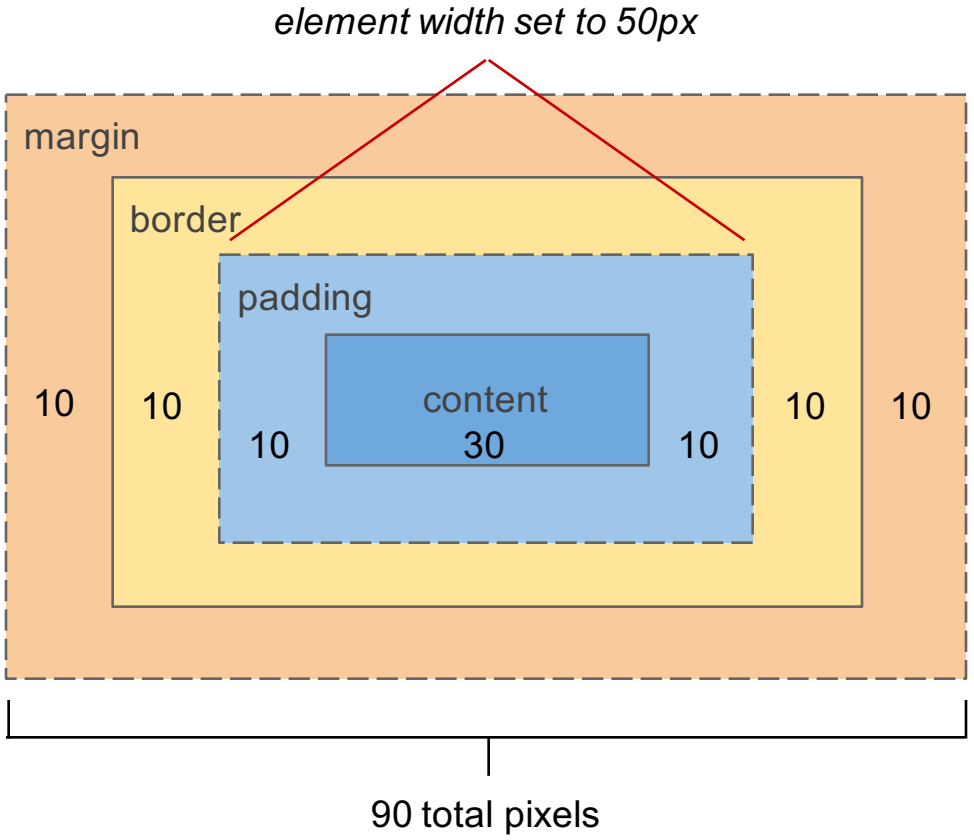
CSS - Content Box

- All four parts of the box added together to get the element's actual total width & height



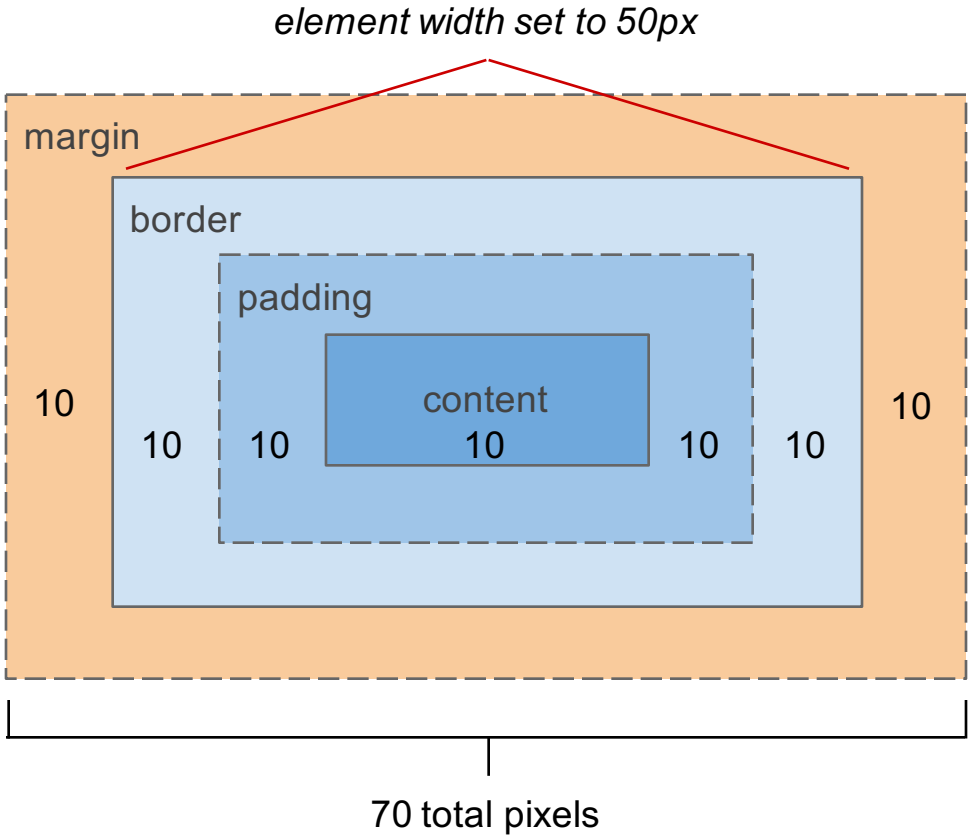
CSS - Padding Box

- The padding is included in the element's width value
- As the padding increases in size, the content decreases in size
- Border and margin are added after to get the element's actual total width & height



CSS - Border Box

- The padding and border are included in the element's width value
- As the padding or border increases in size, the content decreases in size
- Margin is added after to get the element's actual total width & height



CSS - Absolute vs Relative Values

- Pixels and points are absolute
  - Precise but not scalable
  - 1 pixel is 1 dot on the screen
  - 1 point is 1/72 inch (traditionally used in print media)
- Percentages & Ems are relative
  - Percentage - dependent on parent container/element
  - Em - dependent on font size
  - Scalable but not as precise
  - Good for responsive design (more on that in the next lecture)

More on percentages & ems

More on pixels & points

CSS - Positioning & Layouts

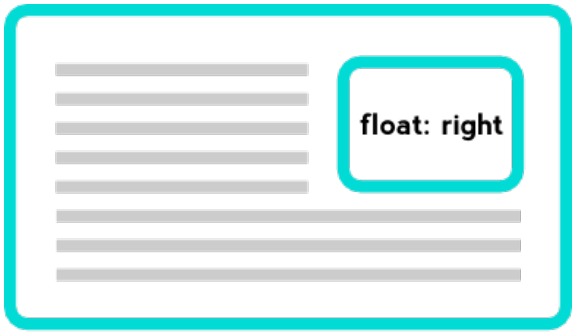
- Most web pages are arranged with elements positioned in a grid layout
  - Layouts are great because they can be reused
- One option is to use **Float**
- Another option is to use **Inline-Block**
- Floats are generally better for positioning individual or groups of elements on the page, while inline-block is generally better for creating a grid layout
  - But there is no perfect answer!

A good explanation of float vs in-line block

A great resource to learn more about layouts

CSS - Float

- Originally designed with traditional print-like layouts in mind
  - Not designed for creating grid layouts
- Floating an element removes it from the normal flow of the HTML
  - Text wraps around the floated element
- However, floats cause display issues requiring ‘creative’ workarounds
  - Using the :after pseudo-element is least messy clear fix method



Clearfix methods

CSS - Inline-Block

- All the benefits of being able to use margins, padding, width, height, and vertical-align
- No weird collapsing or clearing issues, but still not perfect
    - Horizontal spacing issue between elements

How to fix the horizontal spacing issues with inline-block

CSS - Layout Exercise - You Try It!

- We’re going to create a layout using float and inline-block
- Pull files from Git - they’re in the CSS folder
- Follow along in class

CSS - The Position Property

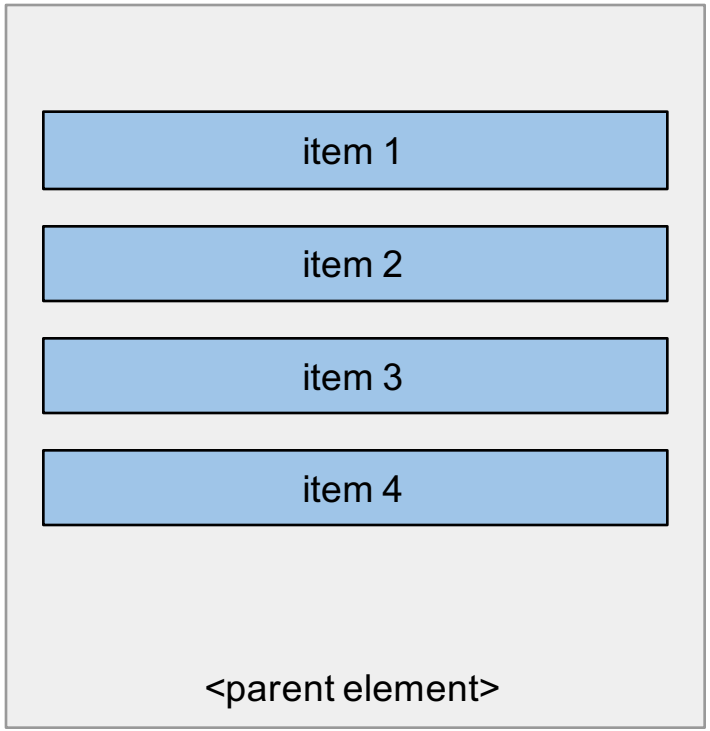
- Use the **Position** property to help layout and position elements
- Default Position value is **Static**, but can also be **Inherit**, **Fixed**, **Relative**, or **Absolute**
- Static, Inherit, and Fixed are pretty straightforward
  - **Static** - element is in normal flow of document, will match HTML layout
  - **Inherit** - element inherits positioning from parent element
  - **Fixed** - element is positioned in the exact same spot in the window, regardless of other content

CSS - Relative vs. Absolute Position

- **Relative** & **Absolute** used to position elements with precision
- **Relative** - element is positioned with respect to itself
- **Absolute** - element is positioned with respect to its closest parent with Relative or Absolute positioning

CSS - Static Position Example

Imagine we have 4 element items stacked on top of each other

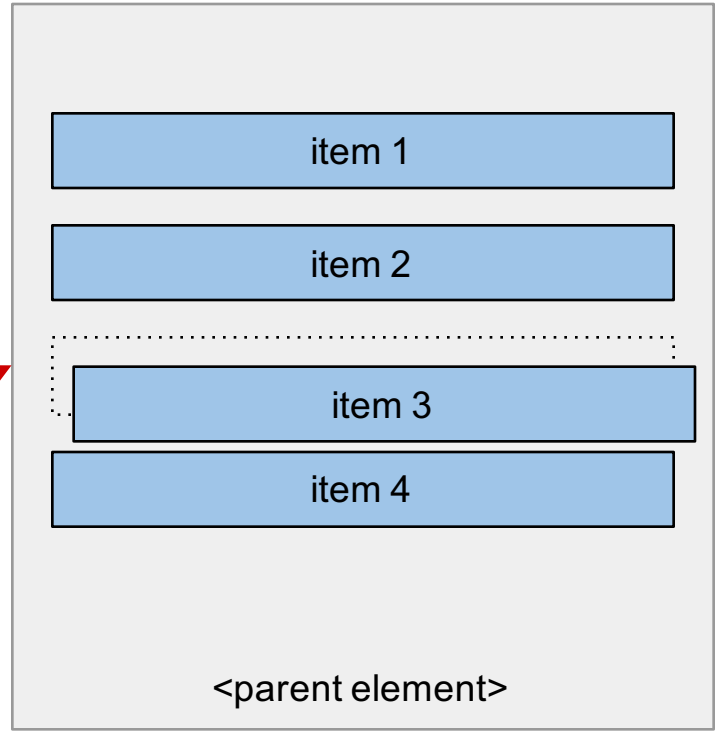


CSS - Relative Position Example

If we set item 3's position to 'relative' and move it 10px from the left and 10px from the top...

```
12
13
14  #item3 {
15      position: relative;
16      left: 10px;
17      top: 10px;
18  }
19
20
21
22
23
```

Item 3 repositioned with respect to original, 'static' position of itself

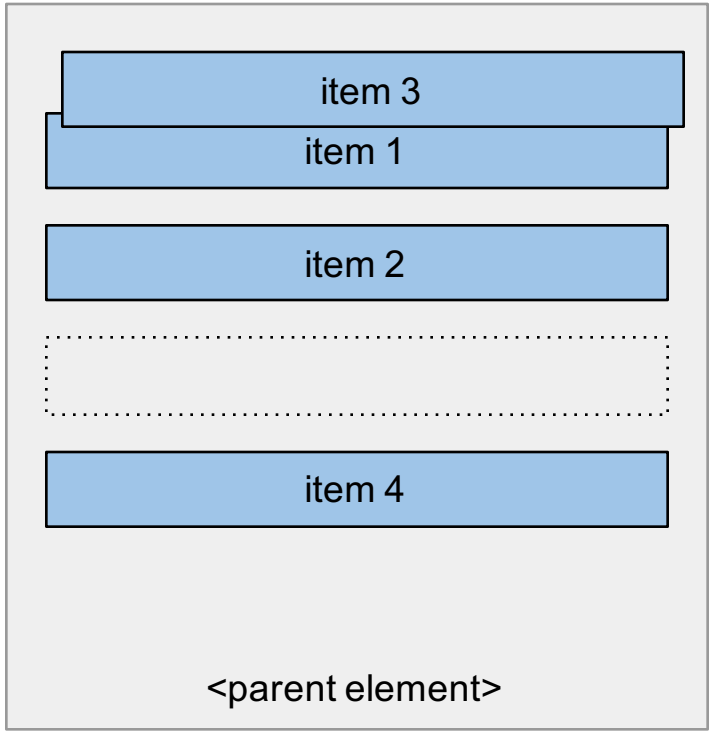


CSS - Absolute Position Example

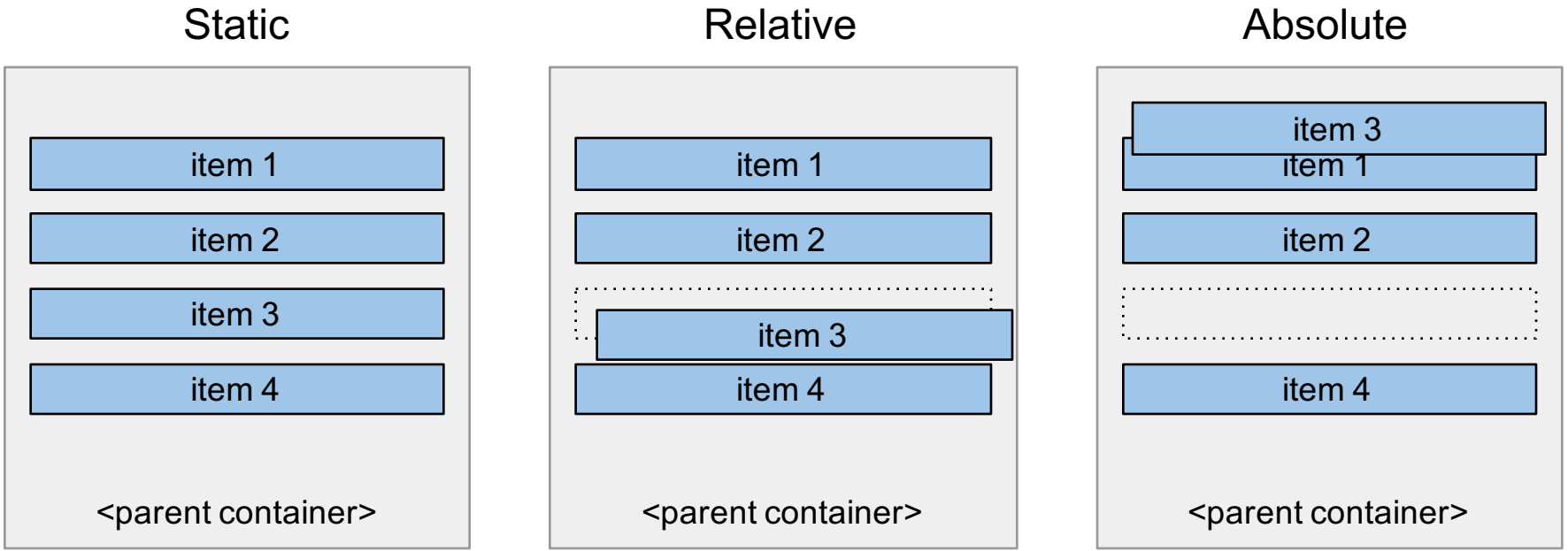
Now, if we set item 3's position to 'absolute'...

```
12
13
14  #item3 {
15      position: absolute;
16      left: 10px;
17      top: 10px;
18  }
19
20
21
22
23
```

Item 3 repositioned with respect to parent element



CSS - Static vs Relative vs Absolute



CSS - When to Use Different Position Values

- Each Position value has its advantages
  - Fixed - good for non-moving items, like menu bars
  - Relative - good for smaller adjustments to an element
  - Absolute - good for when you want an element to always be in the same position relative to its parent

A comprehensive guide to alignment and centering

CSS - Complex Selection

- Combine different selectors to target a specific element or group of elements
- Spaces between selectors indicate **descendant selection**
  - Ex: .container p {width: 60%} selects all paragraphs within elements with the ‘container’ class and sets their width to 60%
- Selectors without spaces indicate a **combination**
  - Ex: p.sale-item {color: red} selects all p elements that have the ‘sale-item’ class and sets their text color to red

CSS - More Best Practices

- Reset
  - Browsers have different styling defaults
  - Need to reset/recalibrate styles so we know what to expect
  - http://meyerweb.com/eric/tools/css/reset/ is a good one
- Reuse
  - This is more system-level thinking
  - If certain styles are being repeated, consolidate into a new class if possible

More on writing good selectors

CSS - In-Class Workshop

- Continue styling HTML code from the layout example earlier in class
  - You can use the same html file. However, create a new copy of the grid-style.css file (rename the css file) and link to it within your HTML file (see image below on how to do this)
  - Change the the typography, font colors, and font sizes
  - Change rest of the site's color scheme so it doesn't look so horrible
  - Fix the 'Felines in a Grid' header so that it doesn't move from the **bottom** of the window
  - Bonus: Use inline-block instead of float for the classes: **.other-news-container** and **.main-container**

Add this line of code to the 'head' section of your HTML file. The href attribute value should match the name of your CSS file.