Week 6: Advanced Topics in CSS & Javascript and Bootstrap

INFSCI 2560
Web Standards & Technologies

Topics

- CSS - advanced topics
- Javascript - advanced topics
- **Break**
- Bootstrap Overview
- Activity

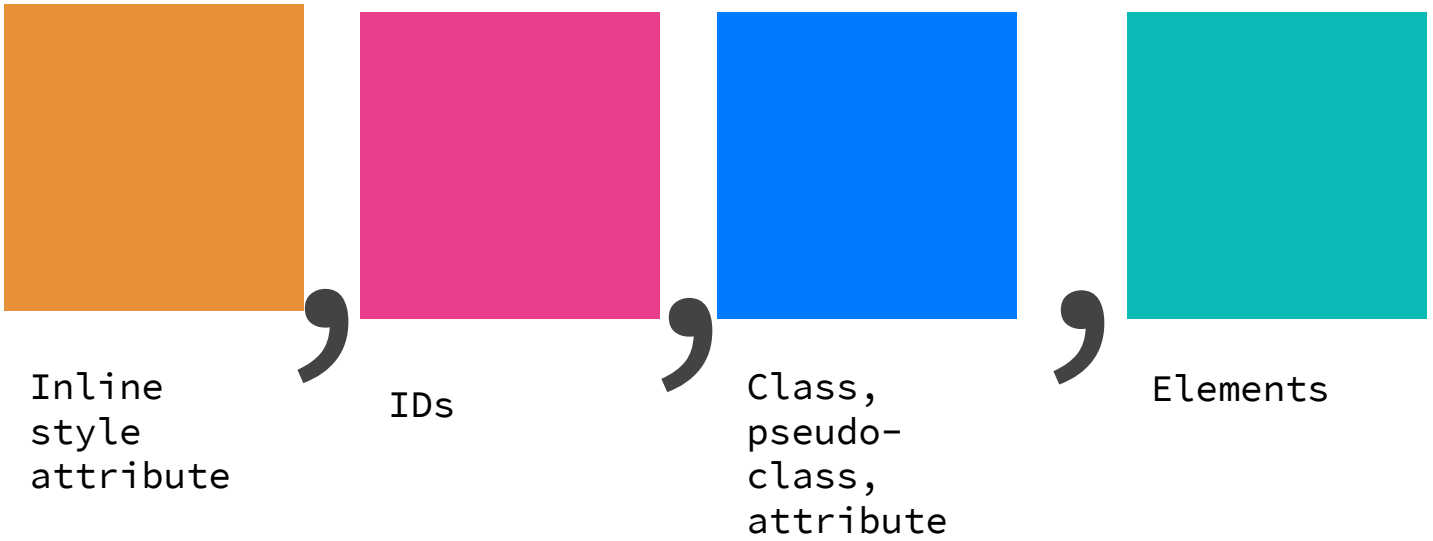
CSS: Rule of Specificity

- When selectors have an equal specificity value, the latest rule is the one that counts.
- When selectors have an unequal specificity value, the more specific rule is the one that counts.
- Rules with more specific selectors have a greater specificity
  - Type selectors (least specific)
  - Class selectors
  - ID selector (most specific)
- `!important` - overrides any other declarations.

What this means:

- The latest rule in the CSS counts
- Embedded styles override external styles
- Class selectors are more specific than element selectors
- If two selectors apply to the same element, the one with higher specificity wins.

Calculating the Rule of Specificity



Most specific > Least specific

The Universal Selector (*)

*, body * and similar selectors have zero specificity.

0,0,0,0

Inherited values also have 0 specificity

Elements and Pseudo-Elements

All element selectors

Includes pseudo-elements like :before and :after.

0,0,0,1

```
h1 { color: red }
<h1>Text is red</h1>
```

Classes, pseudo-classes and attributes

This group includes `.classes`, `[attributes]` and pseudo-classes such as `:hover`, `:focus` etc.

0,0,1,0

```
.red { color: red }
<div class="red">Text is red</div>
```

IDs

ID is an identifier for your page elements, such as #red.

0,1,0,0

```
#red { color: red }
<div id="red">Text is red</div>
```

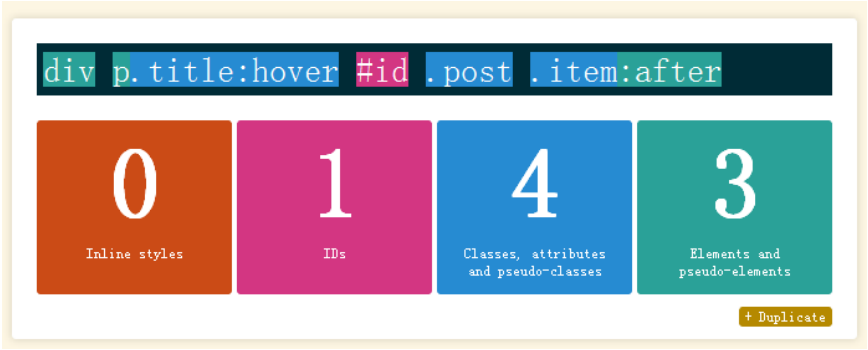
Inline styles

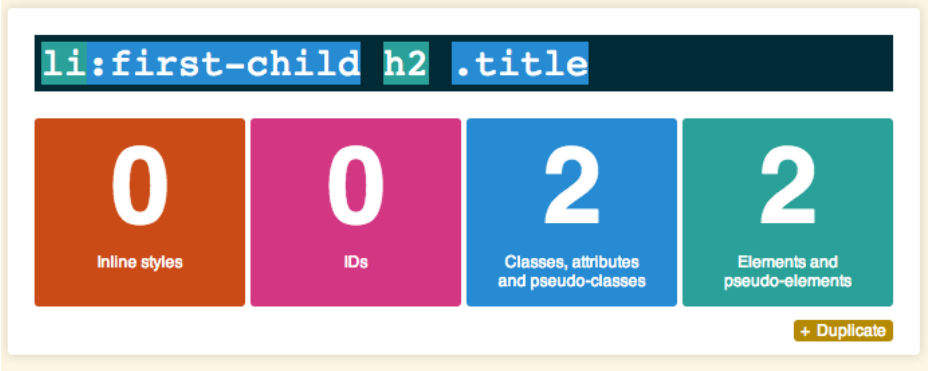
An inline style lives within your HTML document. It is attached directly to the element to be styled.

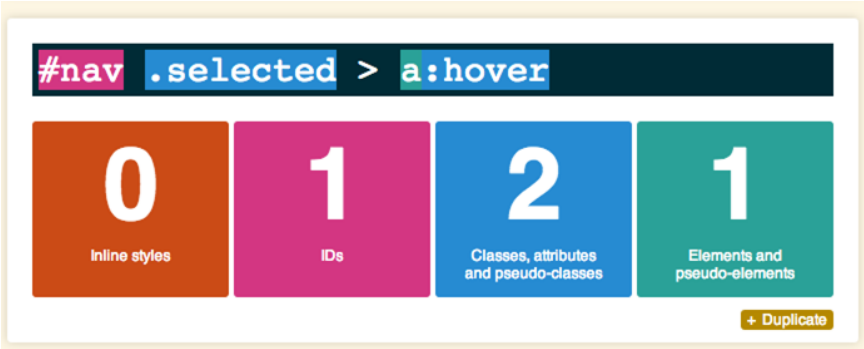
1,0,0,0

```
<div style="color: red">
Text is red</div>
```

Specificity Examples







CSS Resources

Avoid using !important

Great article explaining rule of specificity and another one

Digging Deeper with JavaScript

Defining Variables & Constants

`var` is the most common declarative keyword. A variable declared with the var keyword is available from the function it is declared in.

`let` allows you to declare block-level variables. The declared variable is available from the block it is enclosed in.

`const` allows you to declare variables whose values are never intended to change. The variable is available from the block it is declared in.

Don't use `var` and try to use `const` wherever possible (i.e. when things doing change)

Common JavaScript String Operations

```
var str = 'The quick brown fox jumps over the lazy dog.'
```

```
Str.length //44

str.includes(“fox”) //true

str.indexOf(val) //16

var arr = str.split(‘ ‘)

//[‘The’, ‘quick’, ‘brown’, ...‘dog’]

str.split(‘ ‘,2)

//[‘The’, ‘quick’]
```

```
var regex = /dog/gi;
str.replace(regEx, monkey)

str.toLowerCase();

str.toUpperCase();
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/RegExp

Common Javascript Array Methods

```
const ages = [19, 20, 22, 16, 12];
```

```
ages.length   //5

console.log(...ages);
//19 20 22 16 12

ages.includes("30")   //false

ages.some(function(person){
            return person >18;
        });) //true
```

```
ages.every((person) => person >18);
//false

ages.concat(moreAges)

filteredAges = arr.filter(person) =>
person >18) // [19, 20, 22]

ages.map(age => age + 1)
// [20, 21, 23, 17, 13]
```

Debugging with JavaScript

Printing to the console

`Console.log()`

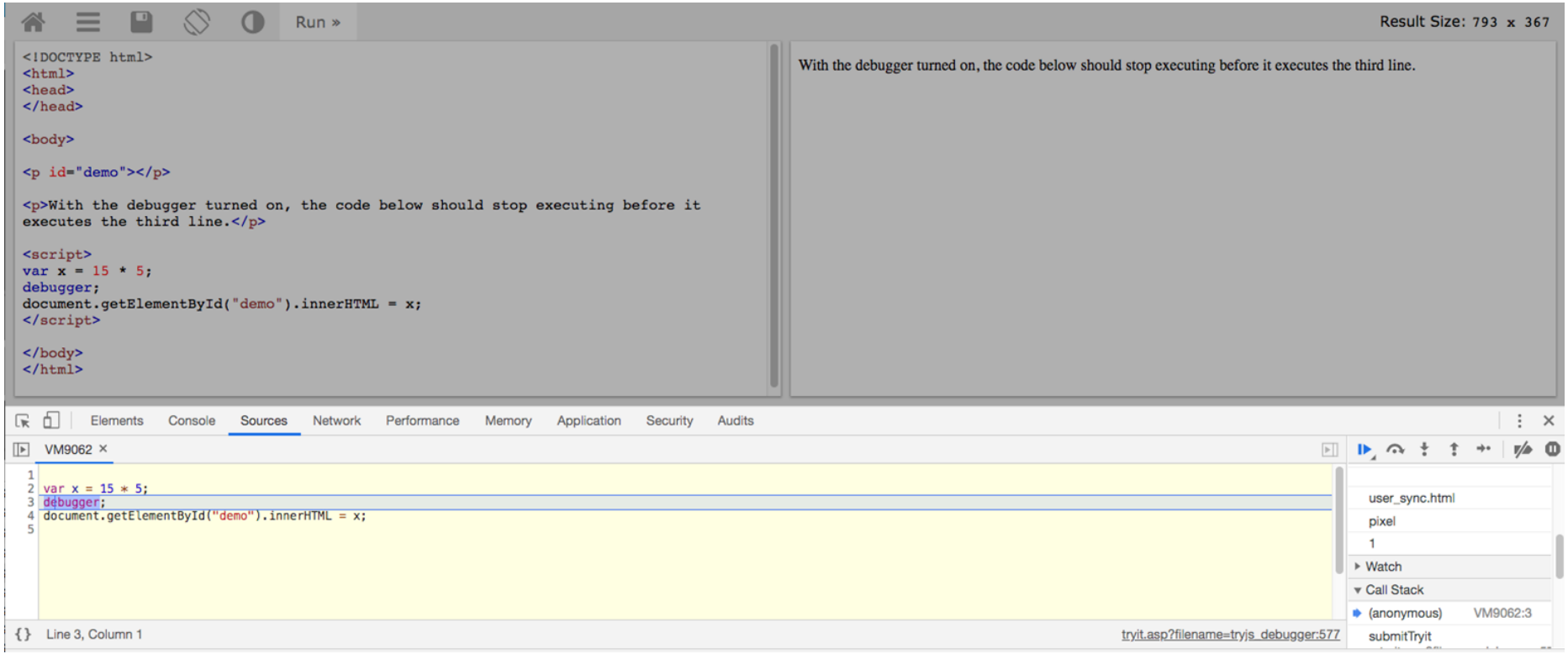
`Console.error()`

The **debugger** Keyword - invokes any available debugging functionality, such as setting a breakpoint. If no debugging functionality is available, this statement has no effect.

```
console.log('abc', 123, true);
// Output:
// abc 123 true
```

```
console.log('Test: %s %j', 123, 'abc');
// Output:
// Test: 123 "abc"
```

Debugger Example



Object Oriented JavaScript

- EMCAScript 2015 introduces the **`class`** keyword to help create objects
- **`class`** keyword - some syntactic sugar for creating an object (rather than writing functions)
  - Use the **`new`** keyword to create an instance
- ***Constructor*** - a special method for creating and initializing an object created with a class. ***THERE CAN BE ONLY ONE*** special method with the name "constructor" in a class.
- **`extends`** - used in the class declaration to specify inheritance
- **`super()`** - calls the constructor function of the parent (super) class

Example

```
class Person {
  constructor(first, last, age, gender, interests) {
    this.name = {
      first,
      last
    };
    this.age = age;
    this.gender = gender;
    this.interests = interests;
  }

  greeting() {
    console.log(`Hi! I'm ${this.name.first}`);
  };

  farewell() {
    console.log(`${this.name.first} has left the building. Bye for now!`);
  };
}
```

Example

```
class Person {
  constructor(first, last, age, gender, interests) {
    this.name = {
      first,
      last
    }
    console.log(`${this.name.first} has left the building. Bye for now!`);
  };
}
```

```
let han = new Person('Han', 'Solo', 25, 'male', ['Smuggling']);
han.greeting();
// Hi! I'm Han

let leia = new Person('Leia', 'Organa', 19, 'female', ['Government']);
leia.farewell();
// Leia has left the building. Bye for now
```

The *this* keyword

- Represents the containing object
  - or undefined or the window object if there is not a clear containing object
- ES5 introduced the bind() method to set the value of a function's this regardless of how it's called, and ES2015 introduced arrow functions which don't provide their own this binding (it retains the this value of the enclosing lexical context).

```
const test = {
  prop: 42,
  func: function() {
    return this.prop;
  },
};

console.log(test.func());
// expected output: 42
```

Arrow Functions

- A concise syntax for writing anonymous functions
- Don't rebind *this*
  - Useful for React

```
var elements = [  'Hydrogen',  'Helium',  'Lithium',
'Beryllium'];

// This statement returns the array: [8, 6, 7, 9]
elements.map(function(element) {
  return element.length;
});

// The regular function above can be written as the
arrow function
elements.map((element) => {
  return element.length;
}); // [8, 6, 7, 9]

// When the only statement in an arrow function is
`return`, we can remove `return` and remove
// the surrounding curly brackets
elements.map(element => element.length); // [8, 6, 7, 9]
```

Modules

- ***Destructuring*** - The destructuring assignment syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.
- ***Export*** - used when creating JavaScript modules to export functions, objects, or primitive values from the module so they can be used by other programs with the import statement.
- ***Import*** - used to import bindings which are exported by another module.

More JS Resources

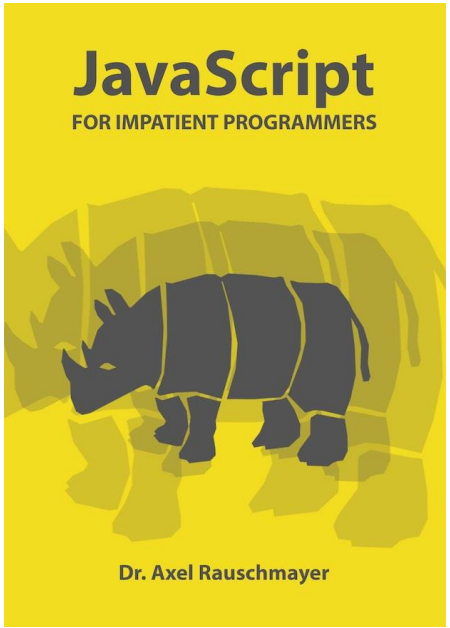
These JavaScript methods will boost your skills in just a few minutes

Most Common Javascript Methods and Gotchas

MDN documentation on Arrays and Strings

Introduction to Object Oriented JavaScript on MDN

Eloquent Javascript by Marijn Haverbeke (free online)



Questions?

Break

CSS Frameworks

Making accessible, responsive, usable, and visually appealing websites is a lot of work

There are lots of quirks

Not all browsers implement the standards in the same way

Or at all

Popular Frameworks

- PureCSS
- Foundation
- Bootstrap

Bootstrap

- Open source HTML, CSS and JavaScript Framework used for front-end web development
- Originally Twitter Bootstrap
- Created as an internal tool with the aim of eliminating inconsistencies among developers using different tools. Soon, many developers started contributing to the project and then it was released as an open-source project.

Example websites built with Bootstrap

Advantages

- Able to create mobile-first, responsive websites *quickly and easily.*
- Free
- Very popular (large community)
- Easy to get started
- Easily customizable
- Excellent documentation

Disadvantages

- You can use it without actually understanding CSS.
- Without customization, your site may look like other Bootstrap websites
- Large file size

Bootstrap Overview

This is a QUICK introduction to the key features of Bootstrap. You are expected to read the documentation/tutorial to fully understand Bootstrap and complete the activity.

Bootstrap Overview

Containers

- Most basic layout element
- Required
- Responsive, fixed-width or fluid-width

```
<div class="container">
  <!-- Content here -->
</div>
```

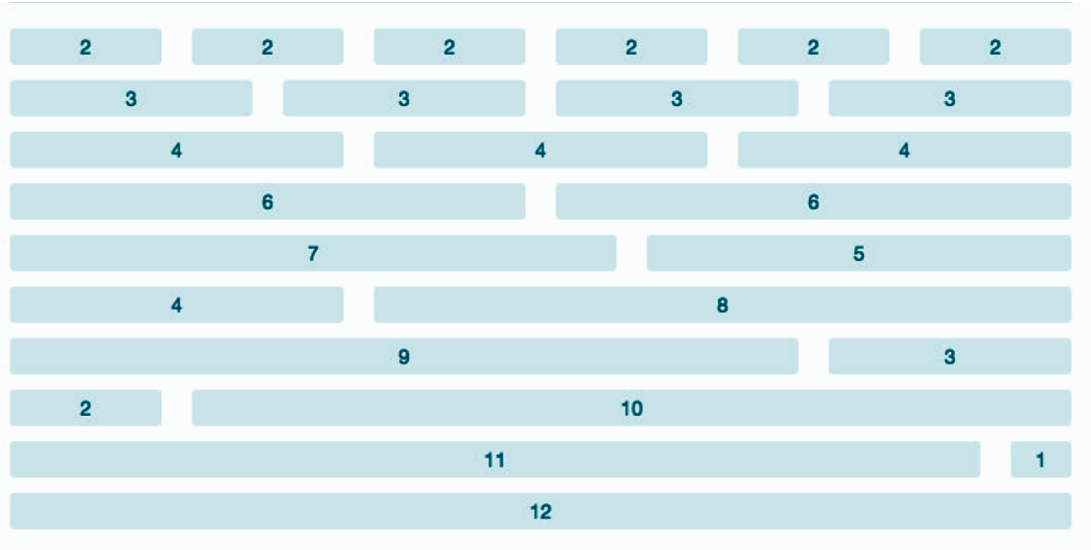
Responsive Breakpoints

- 576px - small devices (phones)
- 768 - medium devices (tablets)
- 992 - large devices (desktop)
- 1200 - extra large devices

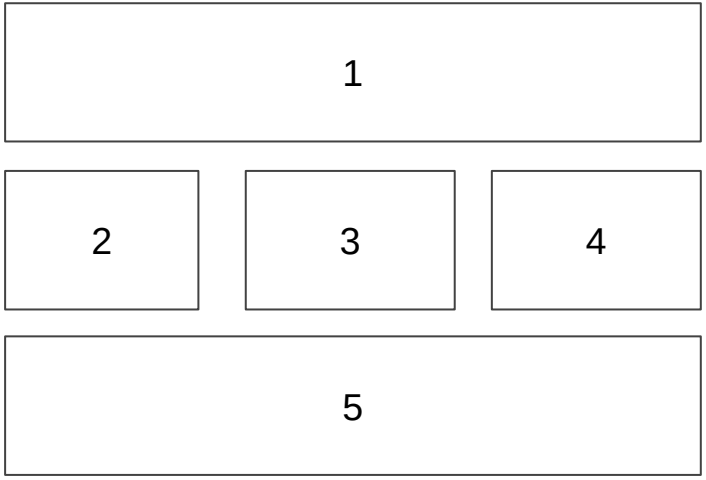
https://getbootstrap.com/docs/4.2/layout/overview/

Grid System

- Fully responsive
- Rows - container for columns
- Columns
- Max 12 columns per row



Basic Bootstrap Layout

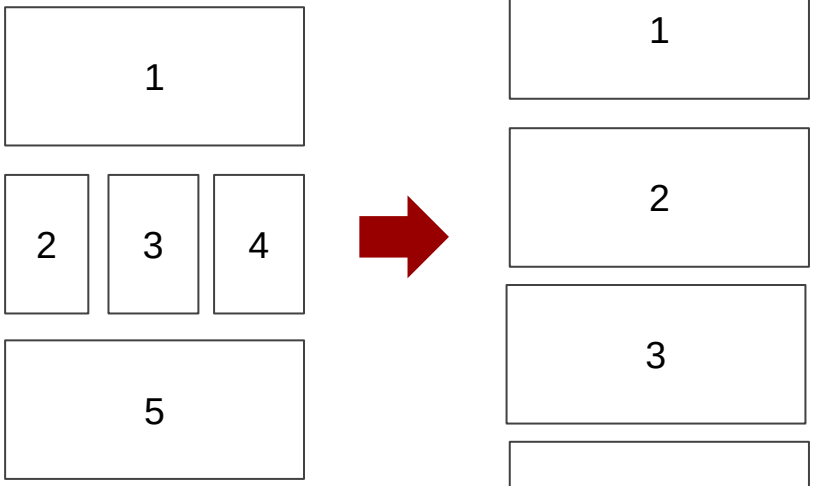



```
<div class =”container”>
          <div class = “row”>
                    <div class = “col”>1
</div>
          </div>

     <div class = “row”>
                    <div class = “col”>2
</div>
                    <div class = “col”>3
</div>
                    <div class = “col”>4
</div>
          </div>
     <div class = “row”>
                    <div class = “col”>5
</div>
          </div>

</div>
```

Making it responsive




```
<div class =”container”>
          <div class = “row”>
                    <div class = “col”>1
</div>
          </div>

     <div class = “row”>
                    <div class = “col-
sm”>2 </div>
                    <div class = “col-
sm”>3 </div>
                    <div class = “col-
sm”>4 </div>
          </div>
     <div class = “row”>
                    <div class = “col”>5
</div>
          </div>

</div>
```

Activity 6 - Bootstrapping

This week you will re-style a page you made for assignment 1 using Bootstrap

1. Review the Bootstrap documentation
2. Remix or copy content from Assignment 1 into a new Glitch Project
3. Bootstrap it!
    a. Wrap content in a container class
    b. add grid columns with row classes
    c. Add three Bootstrap components
4. Make sure the page is readable on Mobile

Don't forget to change the name of the glitch project to "<pittid>-Activity7"!!