ECS 189H
WEB PROGRAMMING

4/14

CSS override rules

- CSS stands for "Cascading Style Sheets"
- What this means is that you can link several CSS style sheets to one Web page; we've already done this with "reset.css" and our page-specific css files.
- It is easy to end up with multiple contradictory values for a single property.
- How to settle conflicts?

CSS override rules

- First: use the most specific specification:

```
p.bird {
    background-color: lightblue;
}

p {
  background-color: pink;
}
```

CSS override rules

- Second: Use last specified property

```
p.bird {
    background-color: lightblue;
}
p.bird {
  background-color: pink;
}
```

- This is why media query has to come after default

Dynamic Web pages

- Media queries respond to change in viewport size, but we often want to change the page in response to user interaction.
- Could always request a new Web page from the server after user input. What is the problem with that?

Dynamic Web pages

- Media queries respond to change in viewport size, but we often want to change the page in response to user interaction.
- Could always request a new Web page from the server after user input. What is the problem with that?
- Not a good idea because 1) it is slow and 2) it makes a flash.
- We make dynamic pages with Javascript.

Javascript

- Javascript is a language, interpreted by the browser, that can be used to change the Web page (add or remove elements, change properties like color, etc).

Object-oriented programming

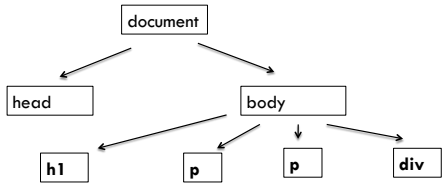
- Javascript is an object-oriented language
- An object is a collection of data, grouped together with functions that act on that data
- Functions belonging to objects are called methods
- Data belonging to objects are called properties

```
/* obj is an object */
obj.someData   /* a property */
obj.aFunction()  /* a method */
```

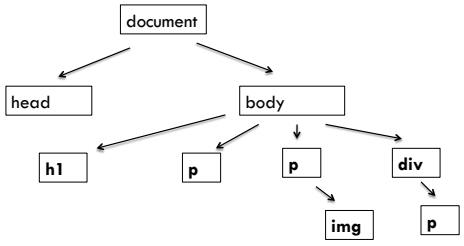
DOM

- Most importantly, the HTML document is an object that is available for the Javascript to modify.
- It has a lot of built-in methods.
- It's properties are all of the elements
- This is the DOM – the Document Object Model

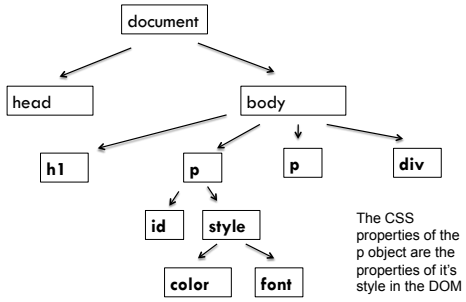
The DOM is a tree



The DOM is a tree



The DOM is a tree



Javascript Example with Button

- The HTML:

```
<p id="theText"> Here is a paragraph. </p>
<button onclick="buttonAction()">click me</button>
```

- "button" is an HTML element
- It's onclick property defines the Javascript function that runs when the button is clicked.

Where is the Javascript?

- Just like css, let's always put Javascript in it's own file.
- Links to scripts can go anywhere in an HTML file.
- Good to link them somewhere consistent. Turns out the best place to put them is after all the HTML.

```
…
<script src="js/actions.js"> </script>
</body>
```

The actual script

```
/* the action taken by the button */
function buttonAction() {
   var demoPgh = document.getElementById("theText");
   demoPgh.textContents = "Now it is different!";
}
```

- Whitespace does not matter
- Lines end with semi-colons
- Curley brackets group blocks of code
- It looks like C or Java. It isn't ☺

The actual script

```
/* the action taken by the button */
function buttonAction() {
   var demoPgh = document.getElementById("theText");
   demoPgh.textContents = "Now it is different!";
}
```

- "document" is the DOM object corresponding to our whole HTML document
- getElementById is a **very useful** built-in method

The actual script

```
/* the action taken by the button */
function buttonAction() {
   var demoPgh = document.getElementById("theText");
   demoPgh.textContents = "Now it is different!";
}
```

- "demoPgh" is a new variable we're defining.
- It's value is the paragraph object with id "theText"
- textContents is one of its properties, containing the text of the paragraph.

Javascript runs in the browser

- The Javascript code is run by the browser while executing the HTML file, as soon as it sees the <script> tag.
- But the text of the paragraph does not change when the HTML file is executed – it does not change until the button is pushed. Why?

Javascript runs in the browser

- The Javascript code is run by the browser while executing the HTML file, as soon as it sees the <script> tag.
- But the text of the paragraph does not change when the HTML file is executed – it does not change until the button is pushed. Why?
- This code just **defines a function;** the function is not run until it is called.
- The function is run when the button is pushed

```
onclick="buttonAction()"
```

Javascript outside of functions

- Does get run immediately when the page is loaded.

```
var demoPgh2 = document.getElementById("theText");
demoPgh2.style.color = "red";
```

- demoPgh2 is a new variable containing the same paragraph
- All elements have a property "style" containing the properties assigned by CSS.
- The color property is changed right after the page is loaded

Placement of the <script> tag

- So why is it a good idea to put the script tag at the end of the HTML file?

Placement of the <script> tag

- So why is it a good idea to put the script tag at the end of the HTML file?
- All the DOM elements it might want to change, using code outside of any function, will have been defined already, and whatever values were assigned by the .css files will have been set.

Strings

- Strings are text data.
- Can use either single or double quotes, or both:

  " 'Have another banana,' she said. "
- Use '\n' for newline and '\t' for tab
- Use + for string concatenation

```
var code = "32";
var outStr = "Today's code is: "+code;
// outStr contains "Today's code is: 32"
```

Numbers

- Only floating point, although may be written differently; there are no integers!
- Conversion is automatic!
- This can lead to some interesting behavior, ie:

```
var a = 5 * "2.0";
// a = 10 – the string became a number

var b = 5 + "2.0";
// b = "52.0"  …  why?
```

Explicit conversion

- To prevent errors, best to explicitly convert:

```
var b = Number("2.0")+5;

var m = Number("cow");  // m contains NaN
```

- The value NaN means "not a number"
- Can also convert explicitly to String

```
var m = String(3)+2  // m contains "32"
```