LAST WEEK ON IO LAB

If you haven't done these things already, please do them before we begin today's lecture

Install Firebug and Greasemonkey.

Complete the online skills assessment.

Join the iolab@ischool mailing list.

You can find links to help with all of these on the course website at
http://courses.ischool.berkeley.edu/i290-4/f09/

INFORMATION ORGANIZATION LAB

Faculty: Bob Glushko
Student Instructors: Nick Doty & Ryan Greenberg

A warning: Today is going to be a full-on crash course in the web client technologies we'll be using for much of this class (HTML/CSS and Javascript/JQuery) followed by actually building on that with web browser extensions like Greasemonkey. This should begin to explain the basics behind the tutorial you worked on this past week and the demo that Ryan did last week, but it may still be a lot to take in and we'll be talking a lot today. We promise that you'll have time to learn this stuff beyond just today and that we won't be talking as much every class.

OFFICE HOURS

Room 210

Nick

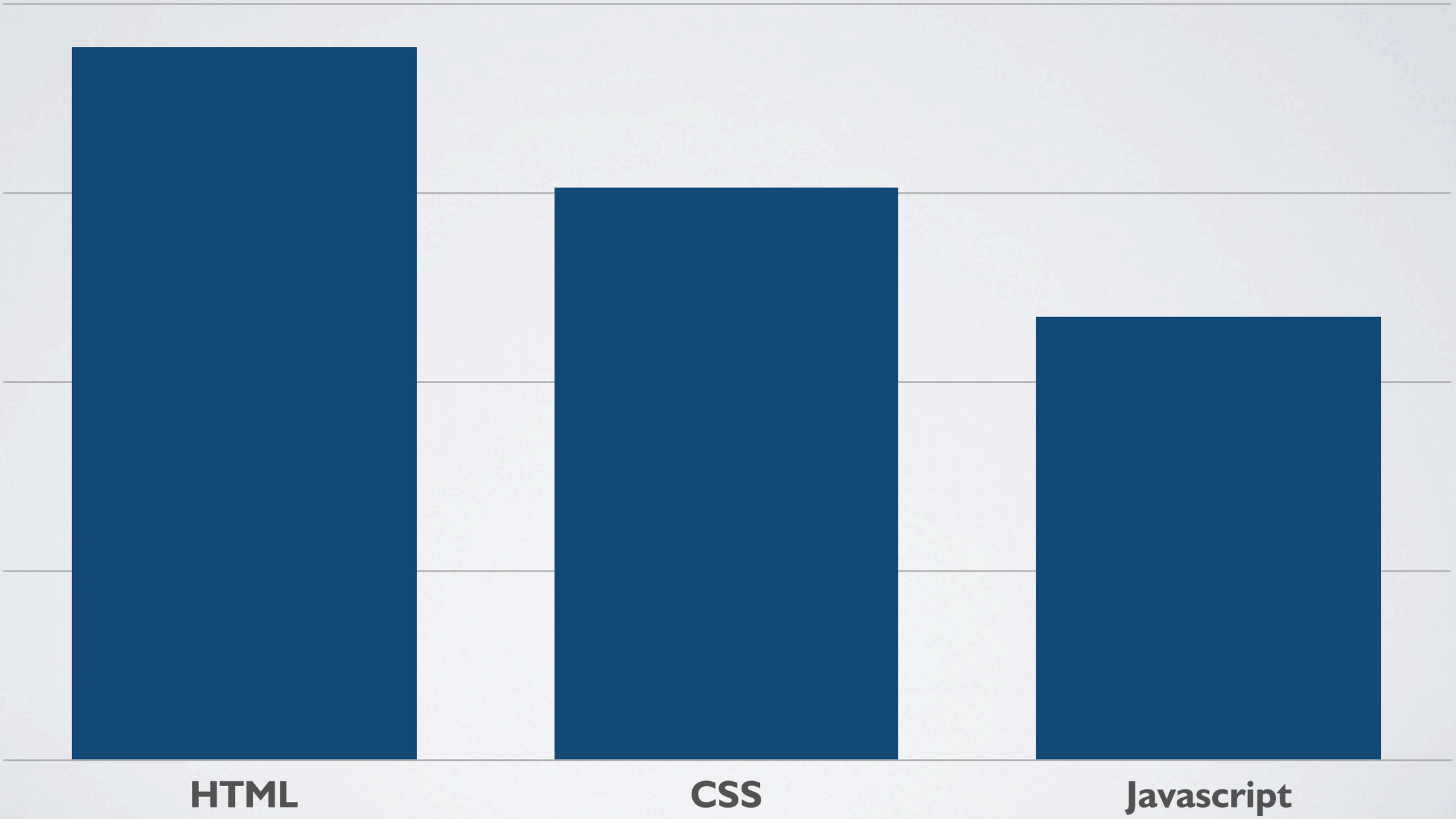
Friday
1:00-2:00

Ryan

Wednesday
1:00–2:00

And by appointment.
Email ryan@ischool or npdoty@ischool to schedule.

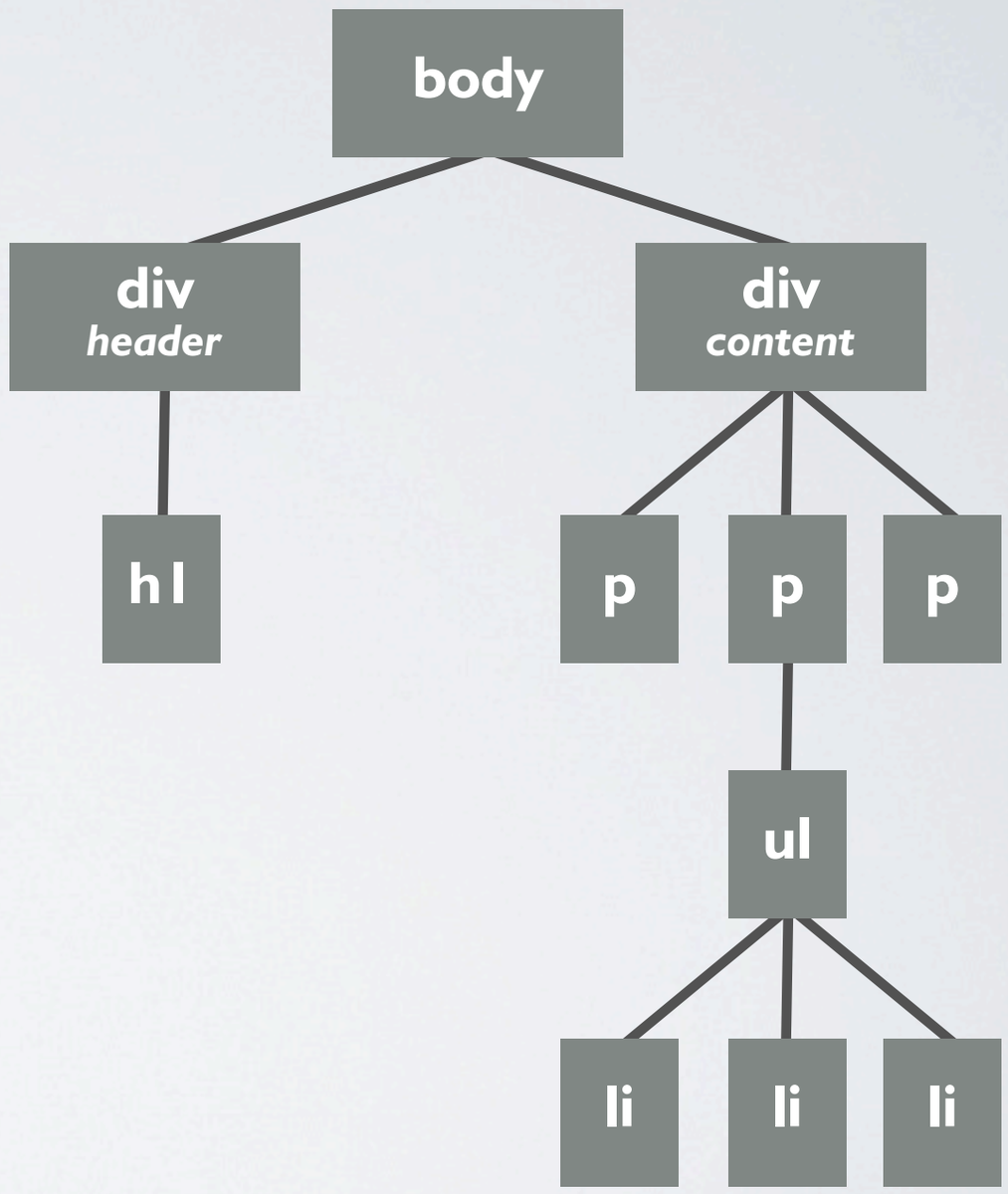
WHAT WE KNOW



Based on the results of the unscientific survey you completed last week.

DOCUMENT OBJECT MODEL

```
<html>
<body>
<div id="header">
    <h1>Document Object Model</h1>
</div>
<div id="content">
    <p>This is my first paragraph</p>
    <p>My second paragraph has a list:
        <ul>
            <li>Item One</li>
            <li>Item Two</li>
            <li>Item Three</li>
        </ul>
    </p>
    <p>This is the third paragraph</p>
</div>
</body>
</html>
```



Here on the right side is a short HTML document. When your web browser loads this document, it generates a representation called the Document Object Model.
If your code is properly indented, your can see that the hierarchy of the DOM corresponds to the indentation level.

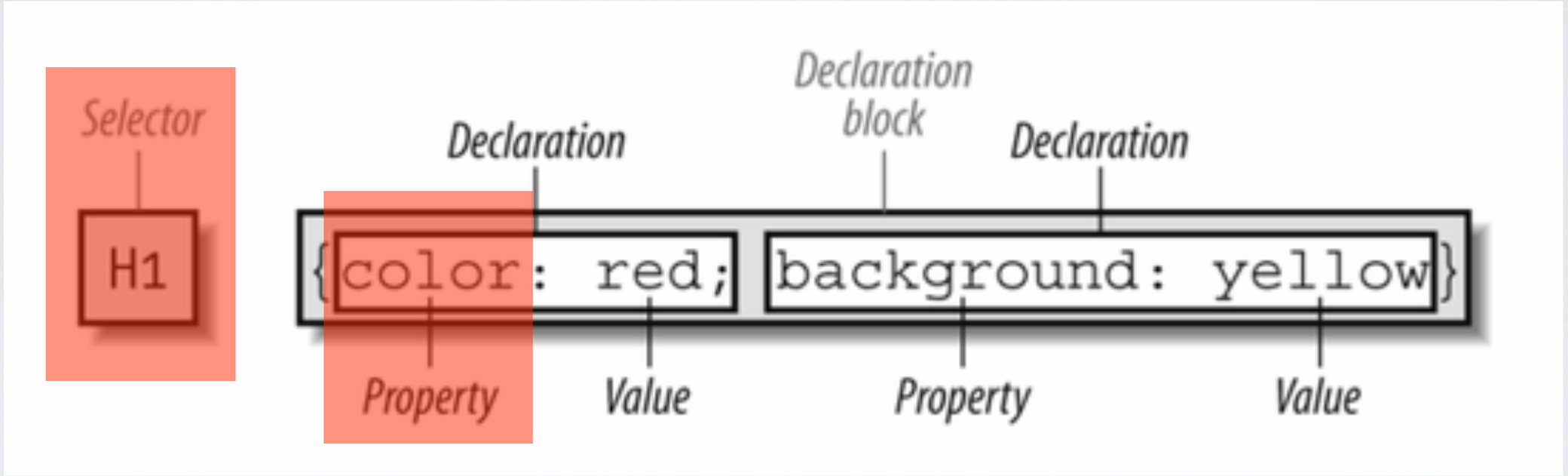
The big idea here is that the left is just a bunch of text. On the right there is a **collection of objects** that you can manipulate and change.

See http://en.wikipedia.org/wiki/Document_Object_Model for more information.

CASCADING STYLE SHEETS

Separate presentation from structure and content.
If you want to be impressed by what's possible with CSS, see http://csszengarden.com.

RULE STRUCTURE



From *CSS: The Definitive Guide*

A stylesheet consists of a series of rules. Here you see the structure of a style rule. You start with a selector, which specifies what elements in the DOM you want this rule to apply to. Then you write one or more declarations to apply styles to that selection. Declarations are separated by semi-colons.

SELECTORS

	CSS	HTML
Type (Tag)	p	<p>
Id	#header	id="header"
Class	.author	class="author"
Descendent	div p	<p>
Grouping	h1, h2	<h1> *and* <h2>

Who can explain the difference between IDs and classes? IDs are unique, only occur once on the page. Classes are recurring elements.
Both can add semantic meaning to the page.
For a complete list of selectors in CSS2, see http://www.w3.org/TR/CSS2/selector.html.
For a list of all the selectors that jQuery can use (which are a lot more than CSS2), see http://docs.jquery.com/Selectors.

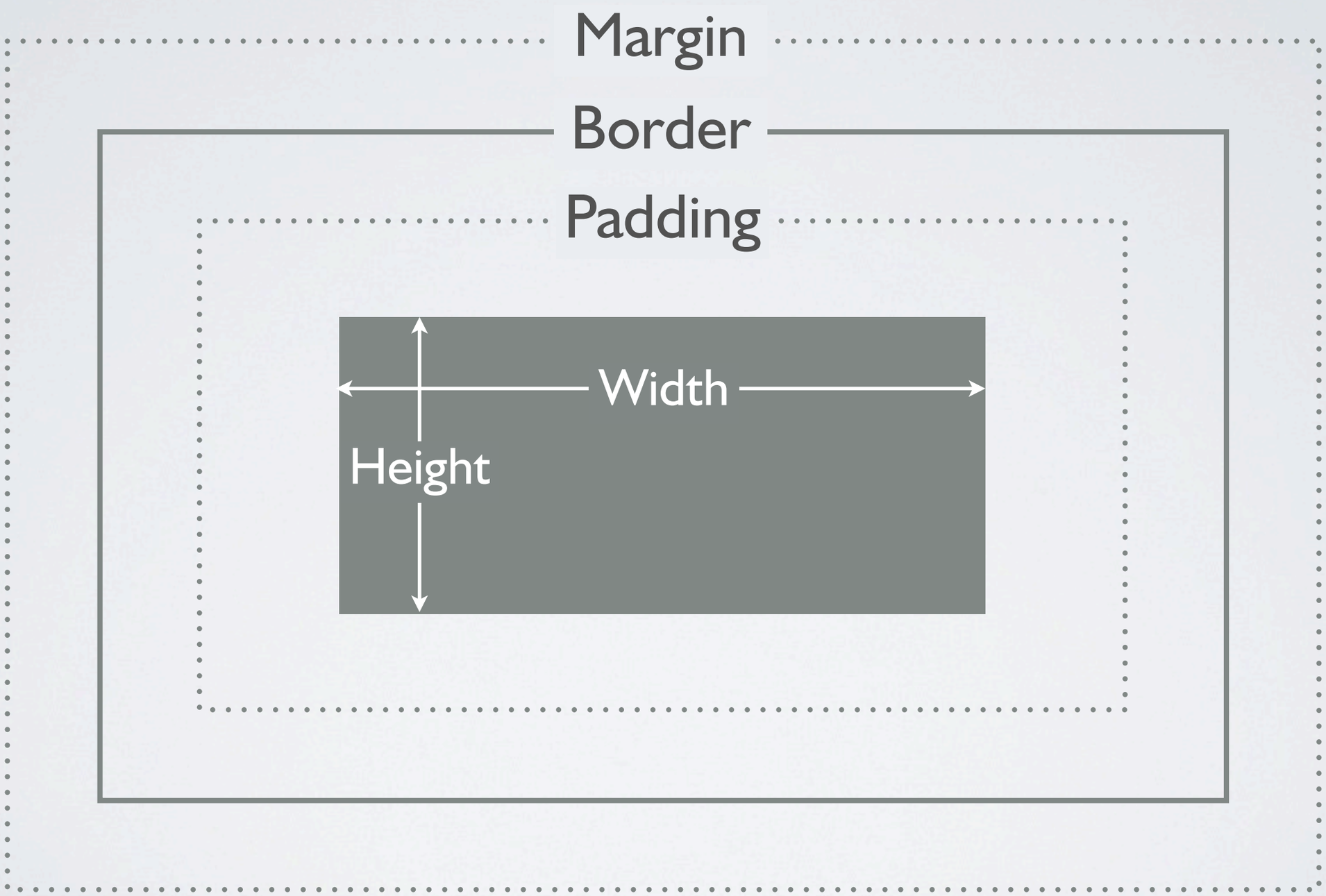
COMMON PROPERTIES

font-family	color	border	display
margin	font-size	width	padding
background	position	text-align	float

See http://htmldog.com/reference/cssproperties/ for a good list of CSS2 properties.

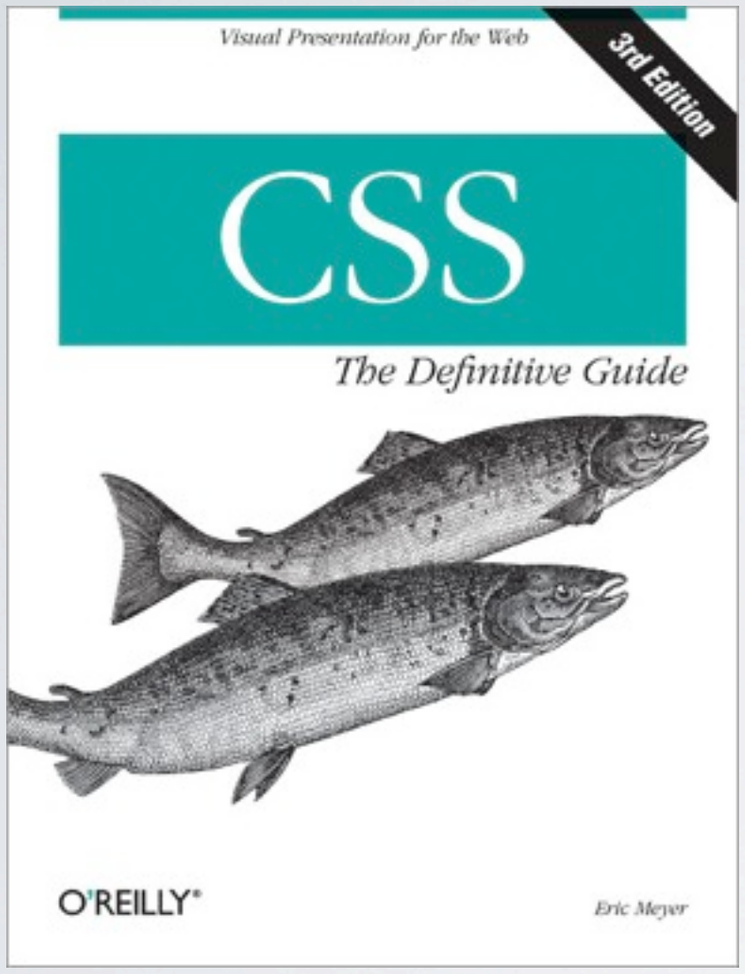
Let's do some examples.
(1) I want to make the blog
(2) Align the text in the header: #header { text-align: center; }
(3) Make every author's name's much bigger. .author
(4) I want to make the titles of the blog entries blue Papyrus.

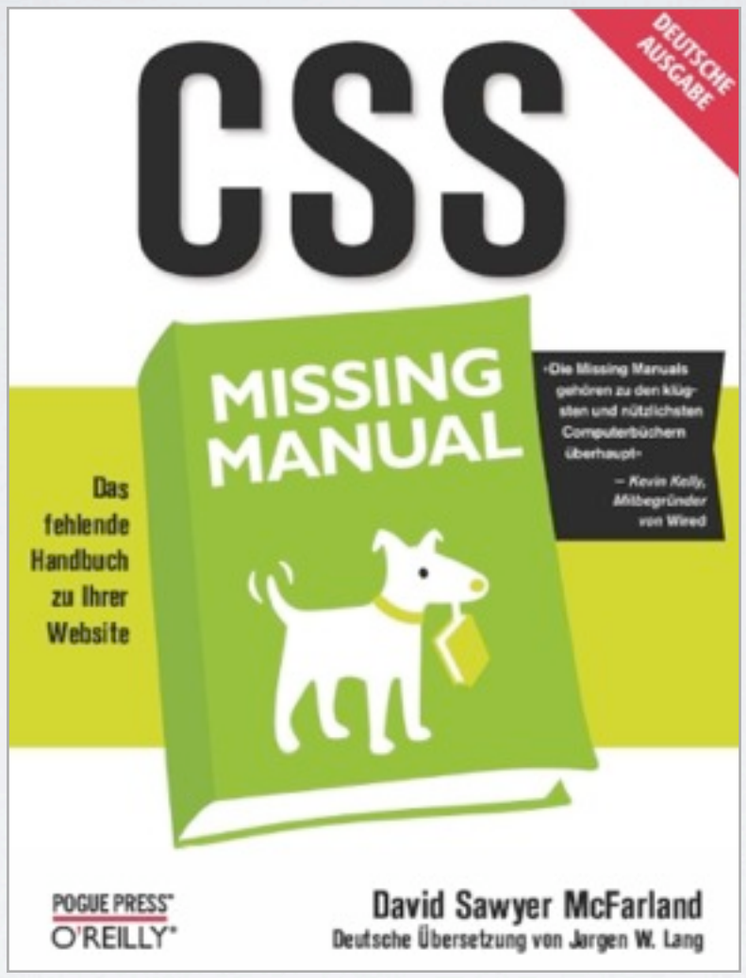
THE BOX MODEL



Every object on the page consists of a rectangular box. The content area, plus padding, border, margin. When you set the width of an element using CSS, that is the width of the content area, not the entire box. If you set {width: 500px; padding: 20px; border: 1px solid black}, the box will be 542px wide: 500, plus 20 padding on each side, plus 1 border on each side.

CSS RESOURCES





Available free for students at http://proquest.safaribooksonline.com.

CSS Definitive Guide: http://proquest.safaribooksonline.com/0596527330
CSS Missing Manual:

Heads-First XHTML & CSS: http://proquest.safaribooksonline.com/059610197X/hfhtmlcss-CHP-8?imagepage=285
“Expanding your vocabulary” http://proquest.safaribooksonline.com/059610197X/hfhtmlcss-CHP-8?imagepage=285

JAVASCRIPT CRASH COURSE

Everyone open up a copy of Firefox and Firebug. If you would like, you can also use Safari's web inspector. Some things in Safari are much better, but I'll be using Firebug. Cross-platform, has a few more developed features.

FIRST THINGS FIRST

JavaScript is a high-level, object-oriented language used most often in web browsers.

You can write comments in your code with // or /* */

A semi-colon goes at the end of every statement.

It's a dynamic, scripting language. Prototype-based inheritance, not class-based. See Douglas Crockford's explanation for more information: http://javascript.crockford.com/prototypal.html

VARIABLES

29

Numbers

'Bob'

Strings

true

Boolean

['Bob', 'John', 'Coye', 'Deirdre']

Arrays

{'name': 'Arnold', 'weight': 240}

Objects

Variables can be of different types. We're going to cover these basic data types.

VARIABLES

```
var stateName = 'California';
```

You use the word ‘var’ to declare a variable. You don’t have to say what type of variable it is--variables in JavaScript are untyped. Convention is to use camelCase.

STRINGS

A sequence of characters.
Use single- or double-quotes to indicate a string.

Examples

```
var myName = "Larry";
myName → "Larry"
myName.length → 5
myName.toUpperCase() → "LARRY"
myName.indexOf('a') → 1
```

ARRAYS

An ordered collection of elements.
Use square brackets to indicate an array.

Examples

```
var myArray = ['dog', 'fish', 'cat'];
myArray.length → 3
myArray[0] → ['dog']
myArray.push('horse') → myArray == ['dog', 'fish', 'cat', 'horse']
myArray.indexOf('fish') → 1
myArray.sort() → ['cat', 'dog', 'fish'];
```

OBJECTS

A collection of key-value pairs or named properties.
Use braces to indicate an object.

Examples

```
var person = {'name': 'Arnold', 'weight': 240, 'height': 6.2 }
person.name → "Arnold"
person.height → 6.2
person.wife = 'Maria';
person.wife → 'Maria'
person['wife'] → 'Maria'
```

The most confusing thing about objects in JavaScript is that they're used for so many different things. First, they fill the role of the data structure: hashes/dictionaries (in Python)/ associative arrays. Second, objects are naturally used for JavaScript's object-oriented programming. Third, JavaScript objects are also the basis for JSON.

You can access the properties of an object using the dot notation or bracket notation.

FUNCTIONS

```
function add(x, y) {
    return x + y;
}
add(2,4) → 6
```

```
var add = function(x, y) {
    return x + y;
}
```

JavaScript functions are defined with the keyword **function** and they **return** a value. Functions can have names, as in the top example, or they can be anonymous.

BROWSER FUNCTIONS

alert('…')

confirm('…')

prompt('…')

console.log('…')

console.log is better to use when debugging because alert (1) doesn't give you any history, (2) you have to click each button. That said, there are times when alert('Testing!') is simply more convenient.

CONTROL STRUCTURES

```
if (3 > 2) {
    alert('3 is greater than 2');
}
```

```
for (var i=0; i < myArray.length; i++) {
    myArray[i];
}
```

The for loop in JavaScript is a standard C-style for loop. The first statement (here var i=0) sets the starting condition. The second statement (i < myArray.length) sets how long the loop will run--until this condition is false. The third statement (i++) says what will happen at the end of each loop.

+

JQUERY

CSS meets JavaScript

jQuery is a JavaScript library (intro. 2006) written by John Resig. When learning: great when you can apply something you know to something else. A lot of JS in browser has to do with selecting objects from the DOM. And we already have something to do that...CSS!

NEW HOTNESS *vs.* OLD AND BUSTED

(But with JavaScript instead of computers)

Javascript has been around for awhile. Tutorials on the internet are old. Used to put your Javascript inline with your HTML. onclick...onload...document.writeln(). Like CSS, the current best practice is to separate your Javascript from the HTML. We don't use onevent anymore.

Also, using JavaScript in browsers has often been onerous. To change all the elements with a certain class, you used to write document.getElementsByTagName('*') ... and a bunch of other stuff. But new libraries like jQuery let you do this more efficiently: $('.name-of-class')

JQUERY

Using jQuery involves two steps:

- Selects objects from the DOM using CSS selectors.
- Do something with the selected elements.

MAIN JQUERY OPERATIONS

- **Attributes**: Changing existing elements.
- **Traversing**: Moving from selected elements in the DOM to others.
- **Manipulating**: Inserting or removing elements.
- **Events**: Attaching functions to events in the browser.

The jQuery Documentation (http://docs.jquery.com), which is well organized and written, uses these names as well. Examples:
- Attributes: $('h1').text() gives you the text from the selected elements. $('h1').text('New Text Here') sets the text in the selected elements.
- Traversing: Moves from selected elements elsewhere in the DOM.
- Manipulating: $('p').after('This text will be added after every paragraph'); $('body').prepend('This will appear at the top of the page')
- Events: You select the elements you want to attach an event to and provide an anonymous function: $('h2').click(function() { alert('You clicked on an h2'); });

WEB BROWSER EXTENSIONS

Greasemonkey and Jetpack and bears, oh my!

A general overview: we'll be looking at a class of tools that extend the functionality of a web browser, either to change the browser chrome (like the status bar or the menu options) or modify an existing webpage.

EXTEND WHAT?

- Browser chrome
- Page content
- Page style
- Page behavior

Chrome: an image in the status bar that lets you know when you have new email
Content: removing an advertisement or adding a map
Style: giving Craiglist any kind of style at all
Behavior: make a button

Greasemonkey

Mozilla Jetpack

Firefox Extensions

Chrome Extensions

Relative advantages and disadvantages. The left-hand column is fairly mature compared to the right. The top row is fairly light-weight (for development and installation) than the bottom row. Firefox Extensions have the best performance but are the hardest to develop. We'll use Greasemonkey here -- it's easy to develop, easy to install, and fairly widespread. But these others have their advantages -- Jetpack makes it particularly easy to add to the browser's chrome and Firefox gives you a lot of power that Greasemonkey doesn't have -- if you want to use one, go for it! But Greasemonkey is a good start for us and development is just like modifying an existing webpage.

Anyone know of others? (Safari Saft, IE Activities....)

GOOD FOR THE BROWSERS

GOOD FOR US

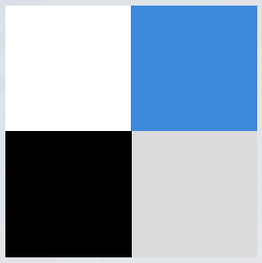
For the browsers:
Let users customize and extend the browser, but keep the core small.

For us:
Let us prototype website and browser features without building either from scratch.

LET'S TRY IT

To the browser / text editor!

FOR NEXT WEEK



For practice, make sure you can build the Delicious Trailmaker all by yourself. Add a feature to it.

Write your first Greasemonkey script. Come with questions next class.

Decide on an idea for Project 1.

You can find links to help with all of these on the course website at
http://courses.ischool.berkeley.edu/i290-4/f09/