Case Study: Web Development

Document Object Model

Event Handling with Javascript

Modern Web Interfaces

Recall: Desktop + Mobile

- Desktop (*Java*)
  - Keyboard / Mouse
  - Large-ish Screens
- Mobile (*Android*)
  - Touch / Soft Keyboard
  - Small Screens
- Web
  - Constraints unknown
  - Concerned with handling <u>all of these</u> ☹

Challenges and Constraints

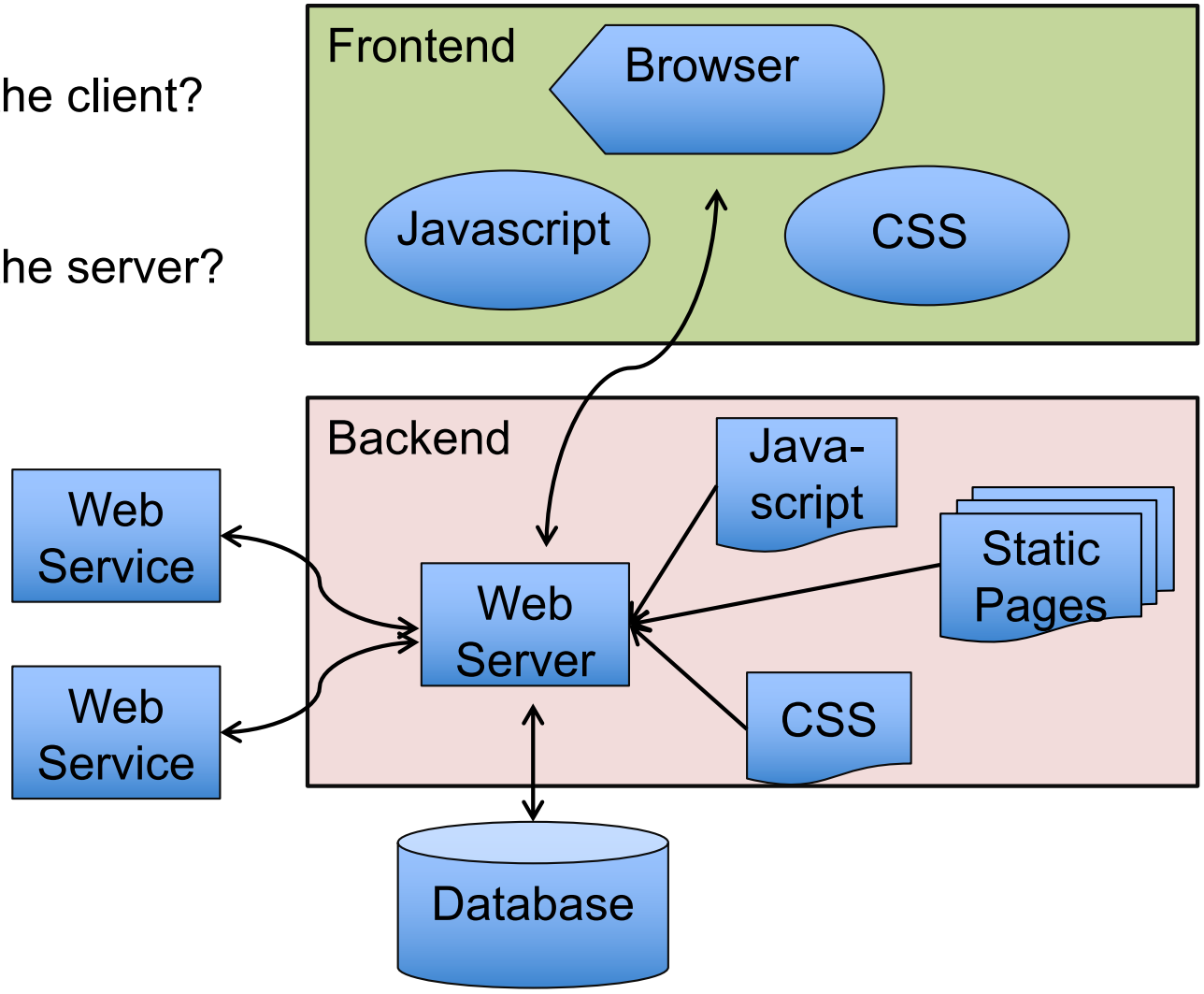
- Effective web interfaces can be challenging to design.
  - Key: Targeting multiple *browsers*
- Supporting uniform interactions on all platforms
  - Differences in what feels “natural” across devices
- Limitations of Device + Browser + Input Mechanisms
  - varying screen sizes
  - varying memory constraints
  - varying processing power
  - varying network speed (problematic for big UIs)

Web Architecture

Web Architecture

Frontend
- What happens on the client?

Backend
- What happens on the server?



Other Stuff:
- Load balancing
- Security
- Etc.

Browers

- How do you support the unknown?

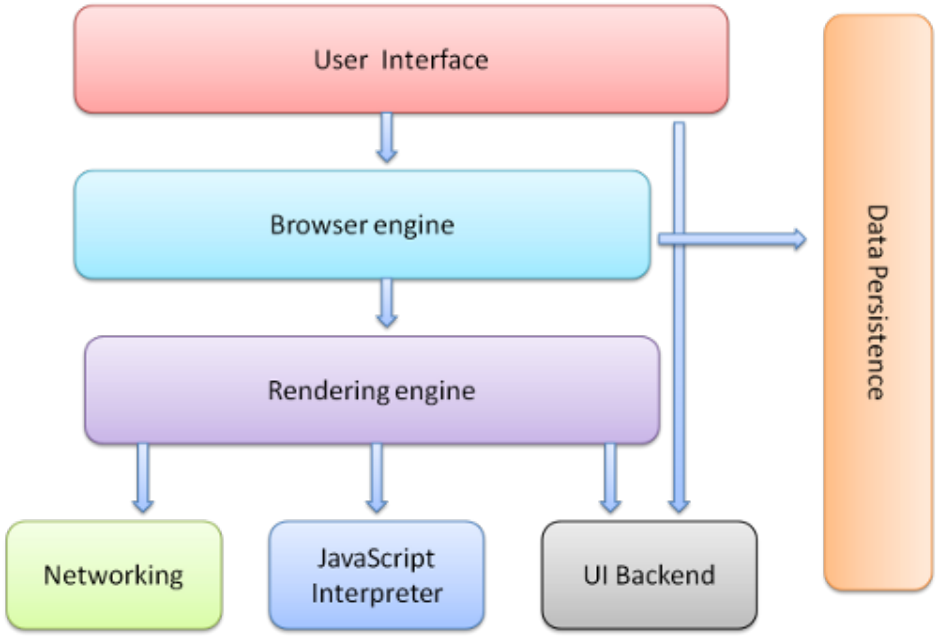
- Browsers
  - Intermediary between web resource + device
  - Task: Retrieve and render a web resource
    - e.g. a web page, an image, etc

- Browsers are on every platform
  - Desktop (Windows, Mac, *nix)
  - Mobile (Android, iOS, etc)
  - Even the Nintendo Switch

Application Model

- Devices don’t render web apps; Browsers do!

- Browsers have 7 components
  1. User Interface
  2. Browser Engine
  3. Rendering Engine
  4. Networking Layer
  5. UI Backend
  6. Data Persistence Layer

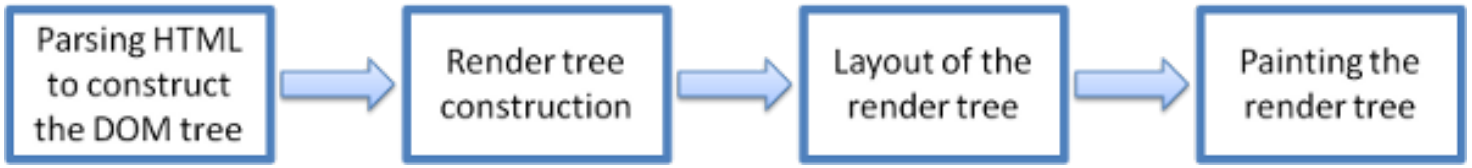


https://taligarsiel.com/Projects/howbrowserswork1.htm

HyperText Markup Language (HTML)

HTML & the DOM

- Hypertext Mark-up Language (HTML)
  - defined 1993, *World Wide Web Consortium* (W3C)
  - analogous to UI elements provided by a GUI toolkit
- The Document Object Model (DOM)
  - defines the structure of the HTML document as well as the behavior of the objects it contains.
  - resembles an interactor tree



Browser Pipeline for Rendering the DOM Tree

https://taligarsiel.com/Projects/howbrowserswork1.htm

A Simple Example

```
<!DOCTYPE html>
<html lang="en">
        <head>
                <meta charset="utf-8"/>
                <title>…</title>
                <link href="css/some-stylesheet.css"  rel="stylesheet"/>
                <script src="scripts/some-script.js"></script>
        </head>

        <body>
                <!-- page content (this is an HTML comment, by the way) -->
        </body>
</html>
```

An example HTML document with most required tags.

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Simple HTML</title>
    </head>
    <body>
        <h1>Simple HTML</h1>

        <p><a href="http://www.lipsum.com">Lorem ipsum</a> d
        do eiusmod tempor incididunt ut labore et dolore mag
        exercitation ullamco...</p>

        <p>Sed ut perspiciatis unde omnis iste natus error s
        totam rem aperiam, eaque ipsa quae ab illo inventore
        sunt explicabo.</p>

        <h2>Some Lists</h2>
        <ol>
            <li>Lorem <em>ipsum</em></li>
            <li>dolor sit amet</li>
            <li>consectetur adipisicing elit</li>
        </ol>

        <ul>
            <li>Lorem <em>ipsum</em></li>
            <li>dolor sit amet</li>
            <li>consectetur adipisicing elit</li>
        </ul>

        <h2>A table</h2>
        <table>
            <tr><td>Intro</td><td>Lorem ipsum dolar sit amet
            <tr><td>Body</td><td>sed do eiusmod tempor incid
            ad minim veniam, quis nostrud exercitation ullam
            <tr><td>Conclusion</td><td>quis nostrud exercita
        </table>

        <h2>An Image</h2>
        <img src="cs649.jpg" alt="Cicero" />
    </body>
</html>
```

Simple HTML

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco...

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Some Lists

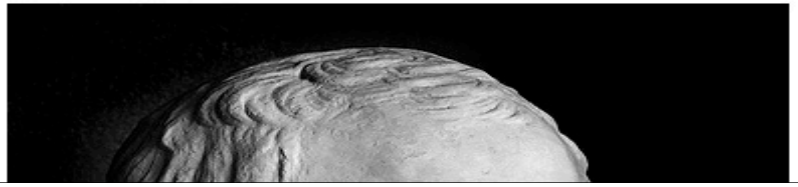
1. Lorem *ipsum*
2. dolor sit amet
3. consectetur adipisicing elit

- Lorem *ipsum*
- dolor sit amet
- consectetur adipisicing elit

A table

Intro	Lorem ipsum dolar sit amet, consectetur adipisicing elit...
Body	sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
Conclusion	quis nostrud exercitation

An Image



Fundamental Tags

…	Headings (also h2…h6)
…	Paragraphs
…	"Anchor" to reference another document; a hyperlink.
… …	Lists. Use for unordered lists and for ordered lists.
	Images
<table> <tr><td>…</td><td>…</td> <tr><td>…</td><td>…</td> </table>	A 2 x 2 table. Related tags include <thead>, <tbody>, and <tfoot> to define headers, footers, and the body. <th> renders cell titles.
…	Apply CSS to a section of the document
…	Apply CSS to a span of characters
br, hr, code, pre, dl/dt/dd	

Tag Attributes

- Tags can have attributes:
  - Defining a URL:
    - Lorem ipsum
  - Defining an Image:
    -

- Universal Attributes

Attribute	Meaning
id	Assigns a unique identifier to the element.
class	Assigns one or more classifications to the element.
style	Apply in-line CSS styles to the element.
title	Provide a title or advisory information about the element.

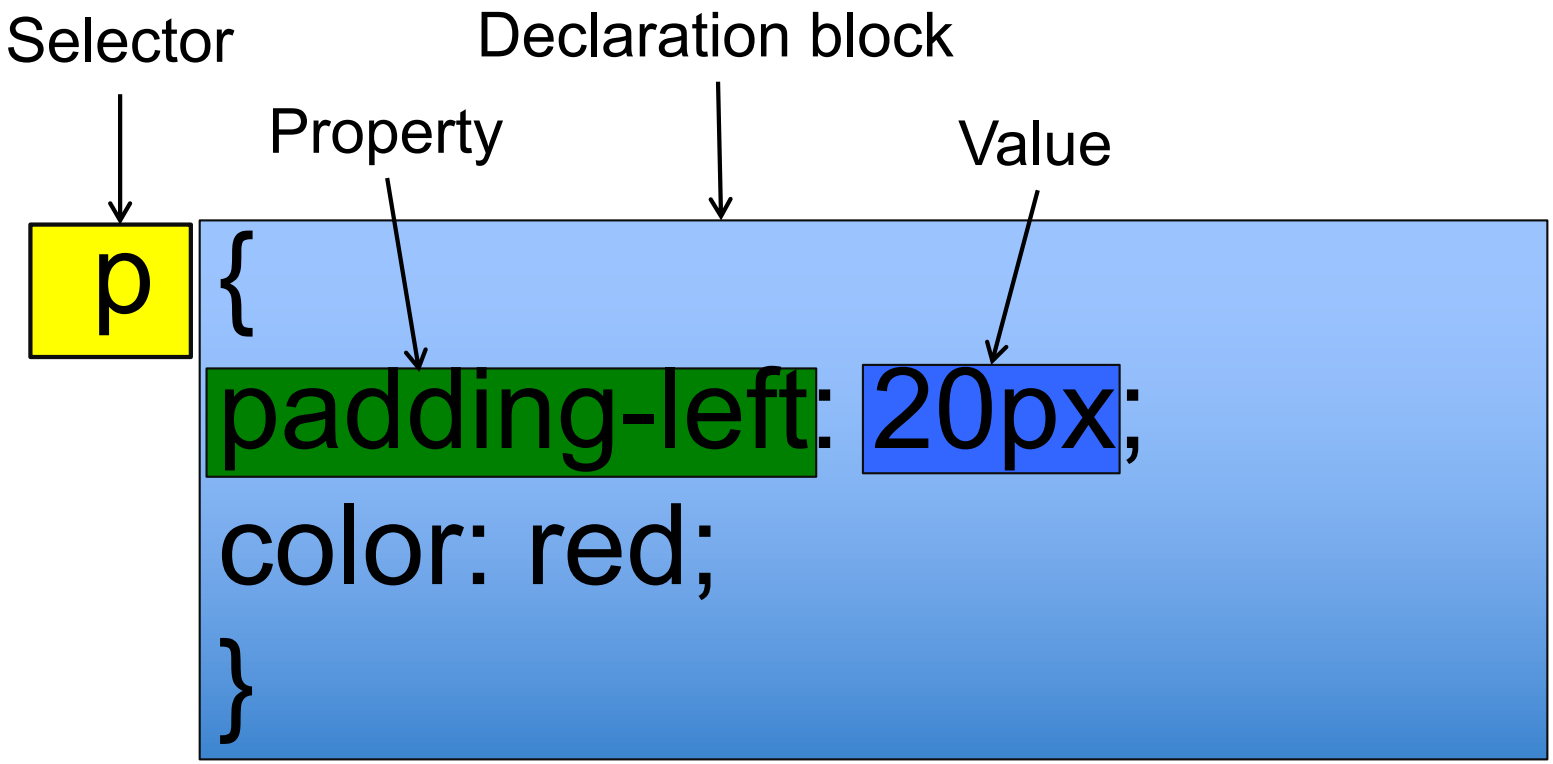
Cascading Style Sheets

CSS: Cascading Style Sheets

- Motivation: Plain HTML is ugly
  - Need to define how HTML is displayed.
  - Separate presentation from underlying content
    - Replace early HTML elements (e.g. <font>, <color>)

- CSS: Usage
  - Apply properties to specific elements in the HTML document
  - Can be in the HTML document or a separate file

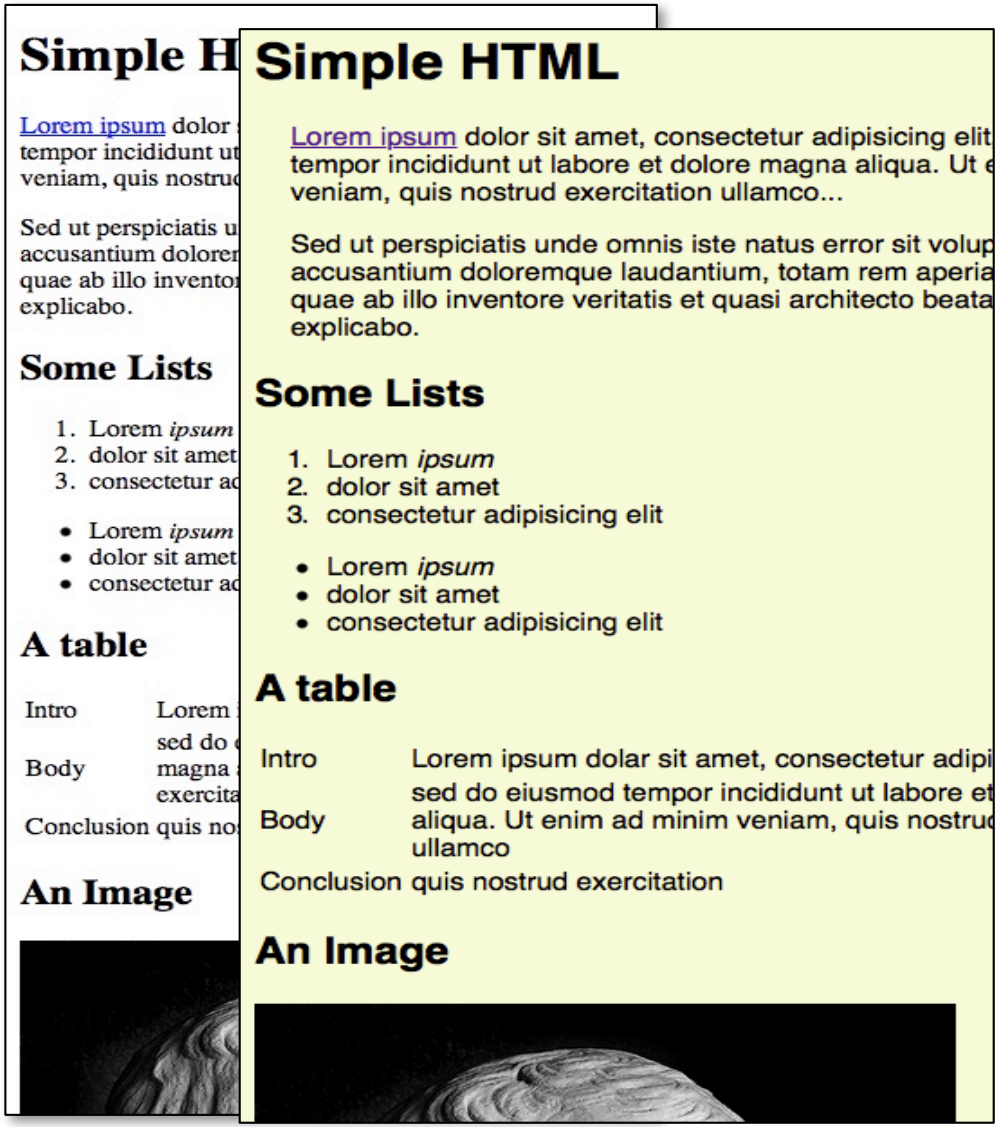
CSS Rule Example



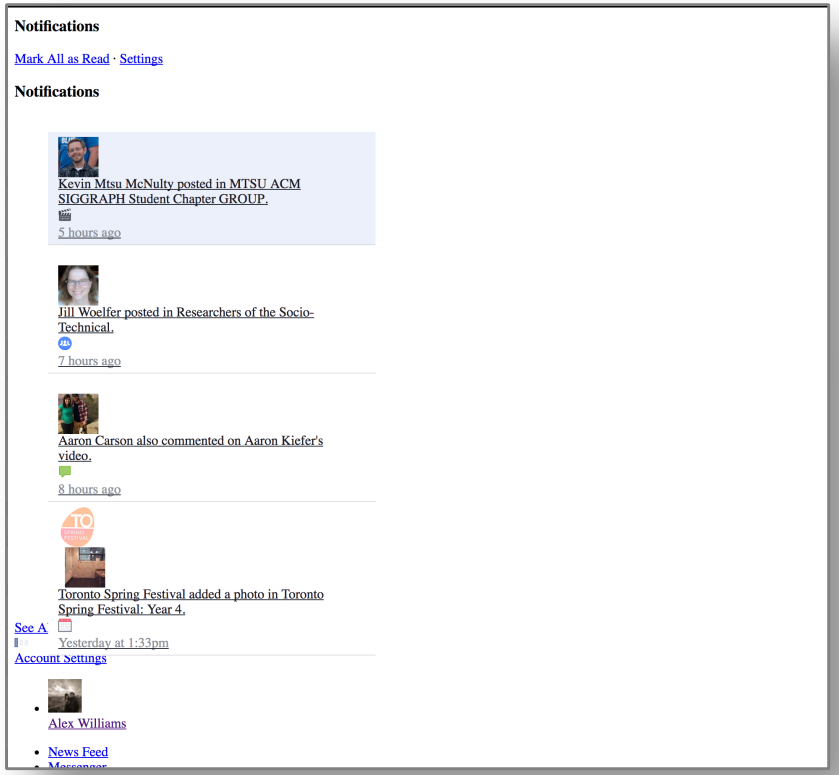
Selectors

```css
* {
  font-family: Helvetica, Arial, sans-serif;
}

body {
  background-color: #F9FAD9;
}

p {
  padding-left: 20px;
}
```

```
<head>
   <meta charset="utf-8"/>
   <title>…</title>
   <link href="css/some-stylesheet.css”
         rel="stylesheet"/>
   <script src="scripts/some-script.js”>
   </script>
</head>
```

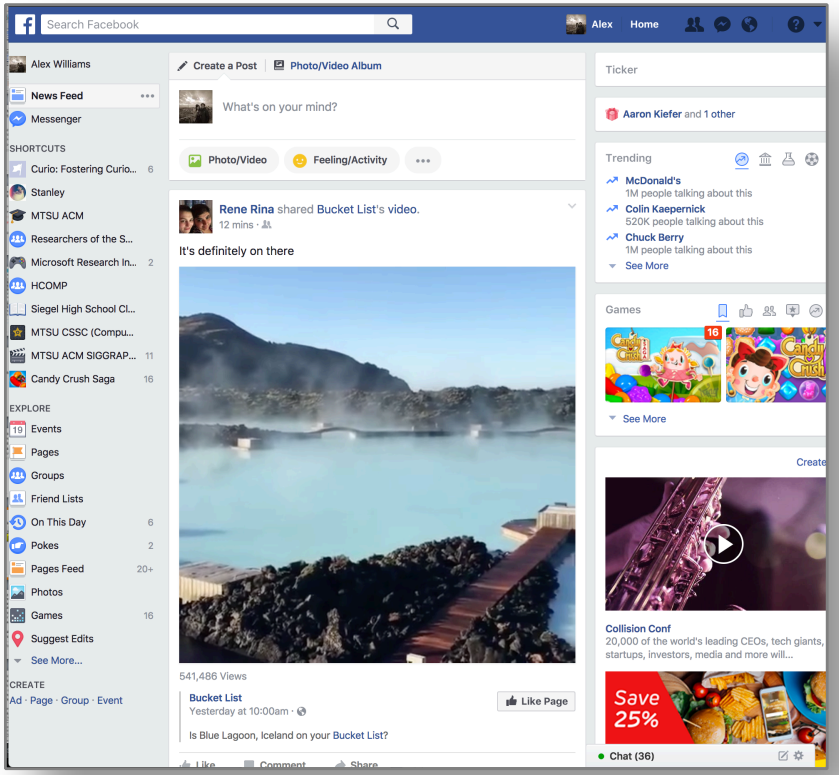


Web without CSS?



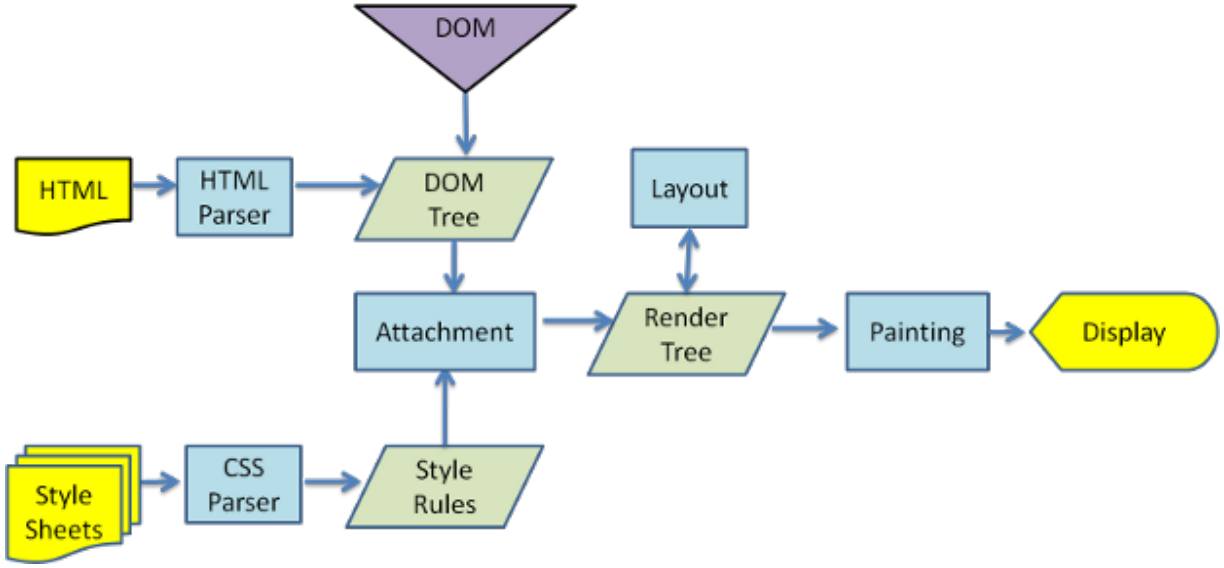
CSS

Apply



How do browsers account for CSS in rendering?

- Browser Rendering Process
  1. Build Render Tree based on HTML + CSS
  2. Exact coordinates determined by Layout process
  3. Render Tree is traversed, and UI Backend paints each node



https://taligarsiel.com/Projects/howbrowserswork1.htm

CSS: UI Frameworks

- Writing styles from scratch is time-consuming.
  - Styling *every* HTML* element?
  - Displaying content based on device constraints?

- Web UI Frameworks
  - Provide a “theme” for HTML elements
  - A CSS file of pre-configured styling
  - Adjust appearance and limited behaviour

- Examples
  - Bootstrap (*Twitter Bootstrap)*
  - Materialize

Bootstrap Example

```
<div class="row">
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
  <div class="col-md-1">.col-md-1</div>
</div>
<div class="row">
  <div class="col-md-8">.col-md-8</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
  <div class="col-md-4">.col-md-4</div>
</div>
<div class="row">
  <div class="col-md-6">.col-md-6</div>
  <div class="col-md-6">.col-md-6</div>
</div>
```

HTML

.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1	.col-md-1
.col-md-8									.col-md-4		
.col-md-4				.col-md-4					.col-md-4		
.col-md-6						.col-md-6					

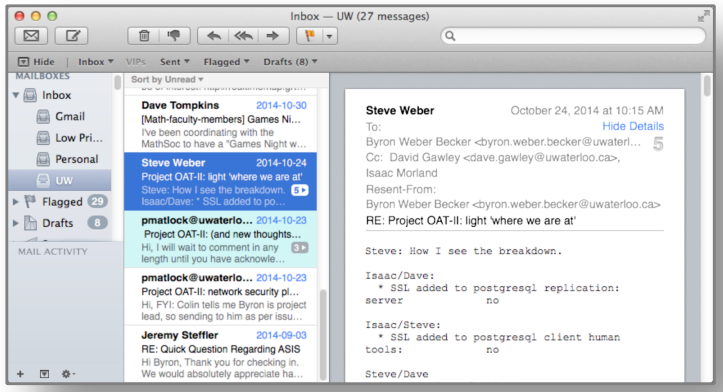
Rendered HTML

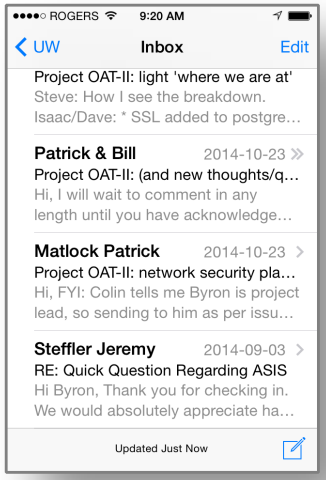
Responsive Design

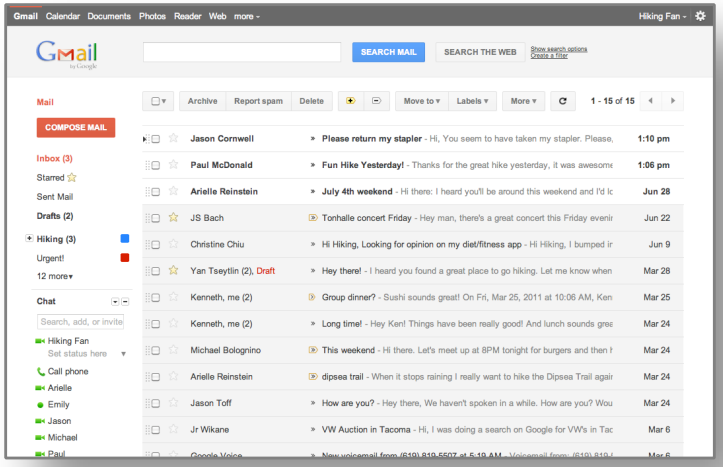
"Rather than tailoring disconnected designs to each of an ever-increasing number of web devices, we can treat them as facets of the same experience. We can design for an optimal viewing experience, but embed standards-based technologies into our designs to make them not only more flexible, but more adaptive to the media that renders them. In short, we need to practice responsive web design."

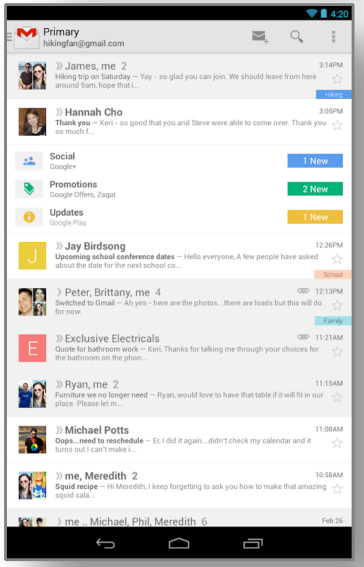
– Ethan Marcotte, *A List Apart*

Recall: Desktop vs. Mobile

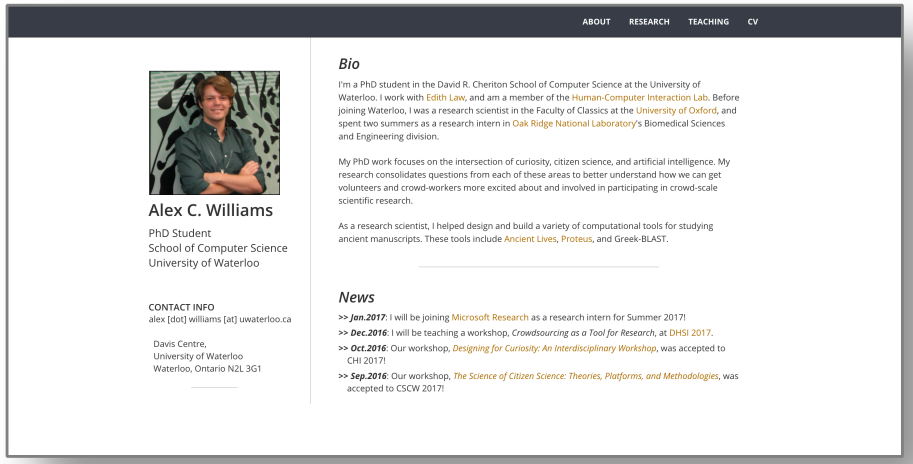




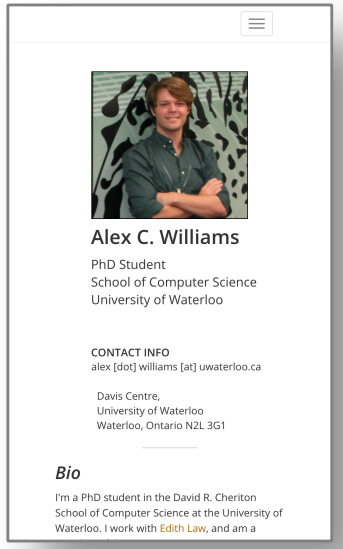




Constraints: Dynamic Layouts



Large Window



Small Window

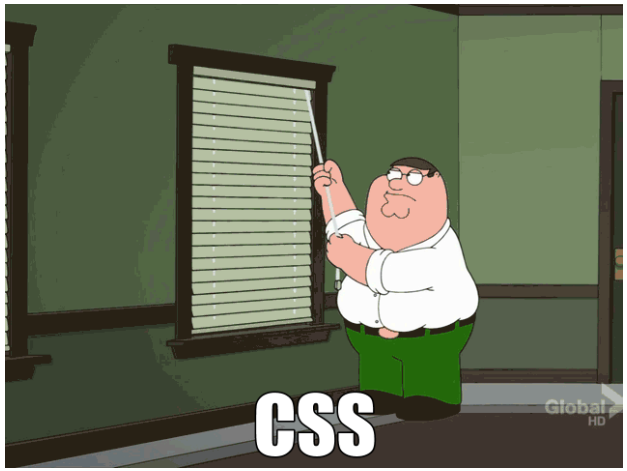
- UI Toolkits can automate layout management!
- “Responsive Layout”: superset of dynamic/adaptive layout that offers specific layouts based on device constraints (and further allows them to be adjusted)

CSS: UI Toolkits

- If toolkits are so great, why doesn't everyone use them?

- Pros:
  - You write minimal CSS! ☺

- Cons
  - Your UI doesn't look *unique.*
  - You have to use the *entire* toolkit.
  - Possible conflicts with existing CSS.
    - Multiple toolkits? Yikes.



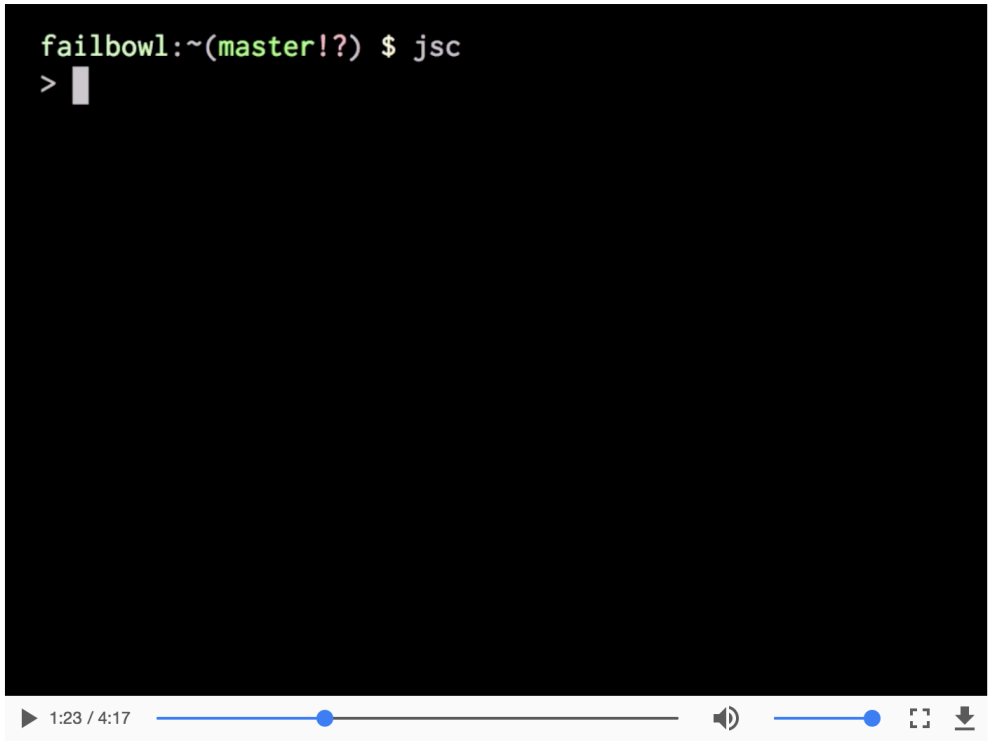
Javascript

Capturing events with jQuery

MVC in Web UIs

Wat

A lightning talk by Gary Bernhardt from CodeMash 2012



https://www.destroyallsoftware.com/talks/wat

1:20 – 4:00

Javascript: Overview

- Developed by Netscape, 1995.
  - **No relation to Java whatsoever**.
  - Became “standard” language for browsers

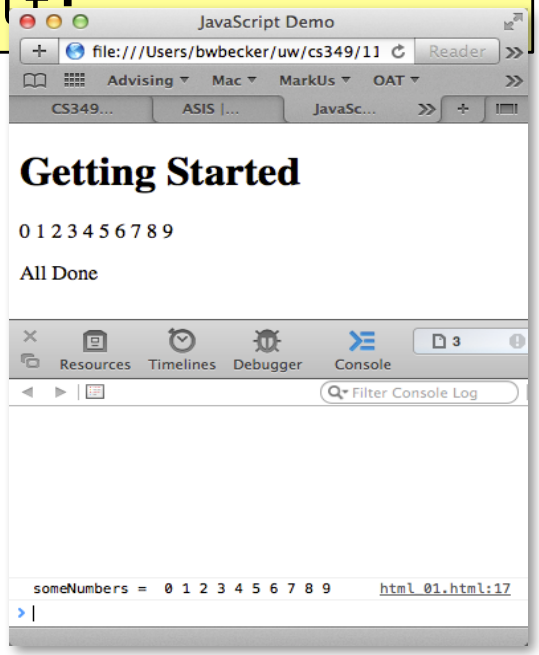
- Characteristics:
  - Interpreted
  - Dynamically typed
  - C-like syntax
  - Inheritance is via prototypes, not classes
  - There are a bunch of “gotchas” (understatement)

- Key Use: Javascript is the event-handling arm of Web UIs.

Getting Started

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>JavaScript Demo</title>
    <link href="html_01.css"
    rel="stylesheet" type="text/css">
    <script src="html_01.js"
    type="text/javascript"></script>
  </head>

  <body>
    <h1>Getting Started</h1>

    <script>
      var someNumbers = numbers(10);
      document.write(someNumbers);
      console.log("someNumbers = " + someNumbers);
    </script>

    <p>All Done</p>
  </body>
</html>
```

```
function numbers(max) {
   var result = "";
   for (i = 0; i < max; i++) {
      result = result + " " +
i;
   }
   return result;
}
```



Event Listeners

Let's say we want to attach a listener to a `<button>` element.

- Most browsers support different ways of attaching listeners.

Level	Method

DOM Level 0	`<button onclick="alert('hello');">`
	`  Say Hello!`
	`</button>`

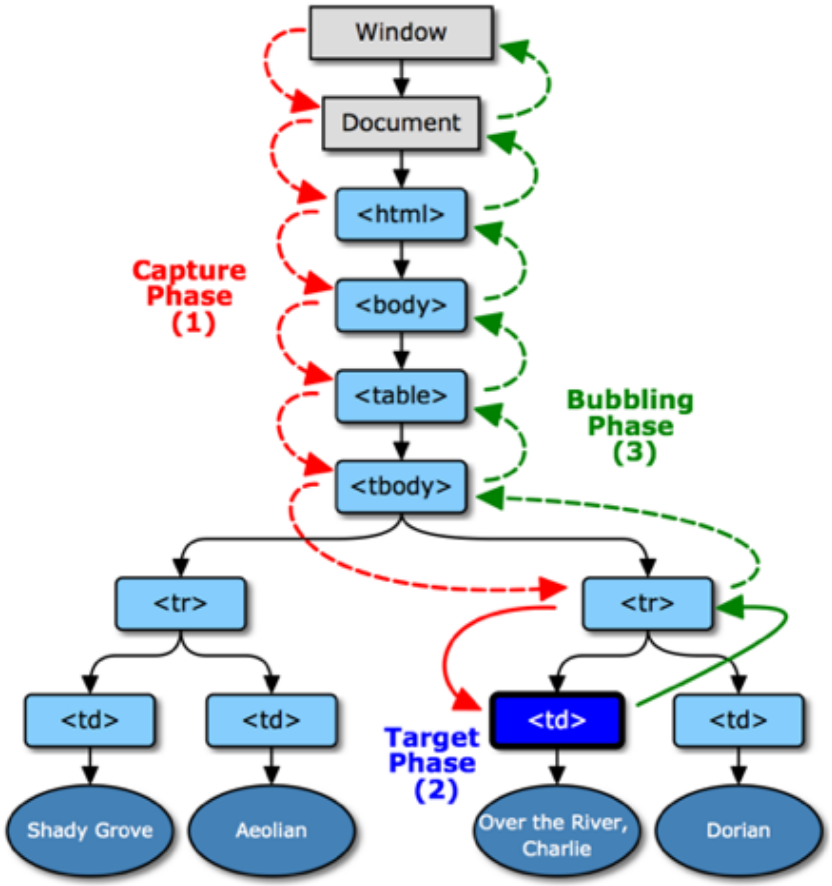
DOM Level 1	`document.getElementById('myButton').onclick =  function()`
	`{alert('Hello!');`
	`}`

DOM Level 2	`var el = document.getElementById('myButton')`
	
	`el.addEventListener('click', function() {`
	`  alert('Hello');`
	`}, false);`

… But, how does Javascript route the event to the element?

Event Propagation

- Events are dispatched to a target through a propagation path. Events register where in the path they wish to be called.
  1. **Capture**: the event travels from the root to the target's parent.
  2. **Target**: the event arrives at the final target.
  3. **Bubble**: the event propagates in reverse order, starting with the target's parents and ending with the root.



Case Study: jQuery

- "Vanilla" Javascript is pretty verbose
  - Time-consuming to write
  - Error-prone
  - Solution: Wrap the terrible parts with something nicer!

- jQuery is a set of libraries that simplifies:
  - Traversing the DOM
  - Animating elements in the DOM
  - Handling events on the DOM
  - AJAX: Asynchronous interaction between client and server

- We only explore event handling here.

Compare and Contrast

Task: Alert “Hello!” when the <button> with id=myButton is clicked.

```
var el = document.getElementById('myButton')

el.addEventListener('click', function() {
    alert('Hello');
}, false);
```

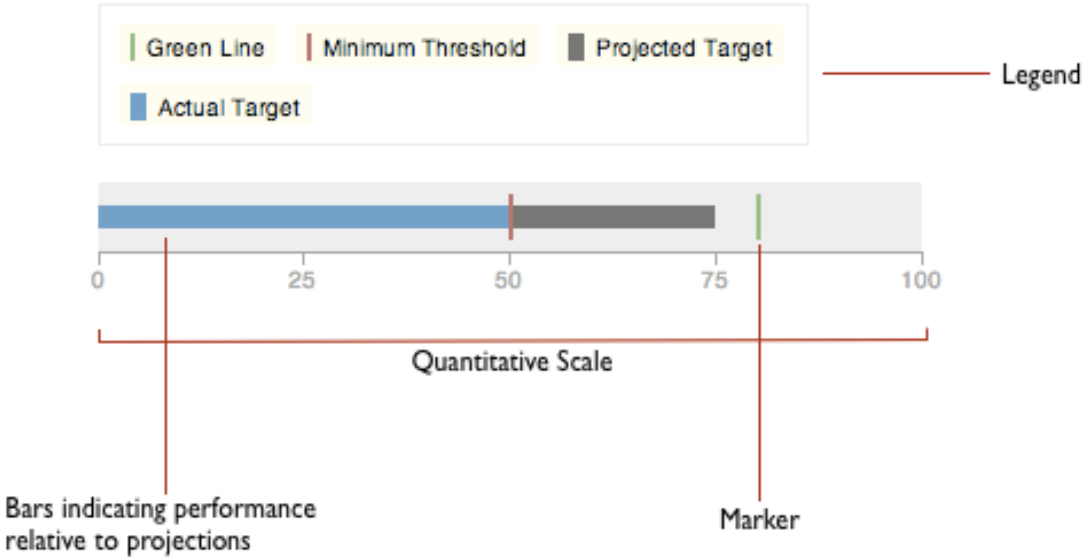
Vanilla Javascript

```
$("#myButton").click(function(){
    alert('Hello');
});
```

jQuery

- jQuery UI
  - A jQuery extension for UI development
  - Widget Factory

- Example: Building a “Bullet Chart”



- jQuery UI
  - A jQuery extension for UI development
  - Widget Factory

- Example: Building a “Bullet Chart”

```
$.widget('nt.bulletchart', {
  options: {},

  _create: function () {},
  _destroy: function () {},


  _setOption: function (key, value) {}
});
```

jQuery Widget Skeleton

- jQuery UI
  - A jQuery extension for UI development
  - Widget Factory

- Example: Building a “Bullet Chart”

```
  $.widget('nt.bulletchart', {
    options: {
      // percentage: 0 - 100
      size: 100,

      //  [{ title: 'Sample Bar', value: 75, css: '' }],
      bars: [],

      //  [{ title: 'Sample Marker', value: 50, css: '' }],
      markers: [],

      // ticks -- percent values
      ticks: [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
    },

    ...
}
```

Define Instance Variables

- jQuery UI
  - A jQuery extension for UI development
  - Widget Factory

- Example: Building a “Bullet Chart”

```
_create: function () {
  this.element.addClass('bullet-chart');

  // chart container
  this._container = $('<div class="chart-container"></div>')
    .appendTo(this.element);

  this._setOptions({
    'size': this.options.size,
    'ticks': this.options.ticks,
    'bars': this.options.bars,
    'markers': this.options.markers
  });

}
```

Define an Init / Constructor

- jQuery UI
  - A jQuery extension for UI development
  - Widget Factory

- Example: Building a “Bullet Chart”

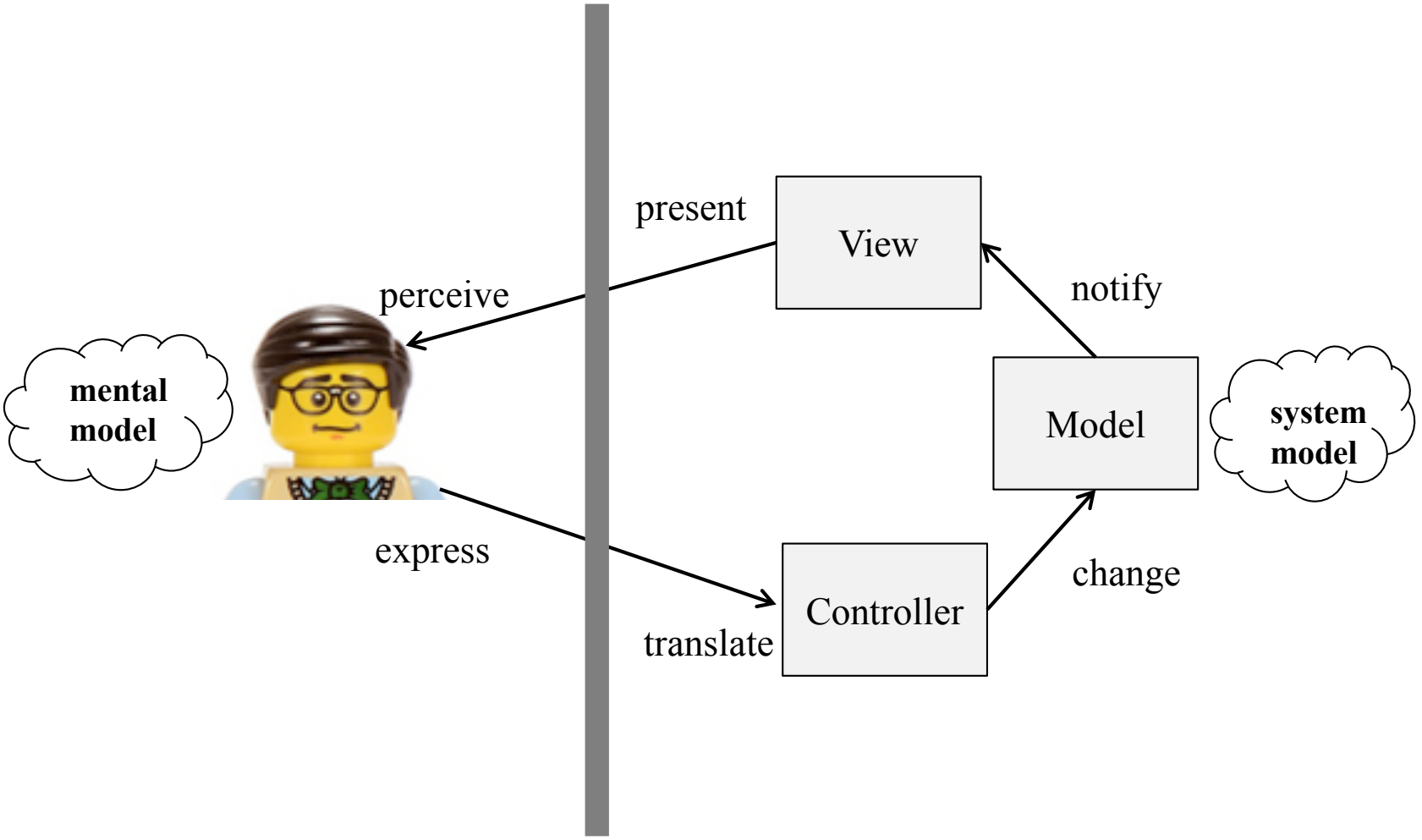
```
_destroy: function () {
  this.element.removeClass('bullet-chart');
  this.element.empty();
},
```

Define a Destructor

Case Study: jQuery

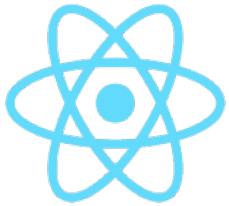
- There are clear benefits!
- Pros:
  - Interacting with the DOM is painless.
  - Can create full-fledged widgets
    - e.g. provided by jQuery UI
  - Others related to AJAX, Animation, etc
  - Extendable!
- Cons:
  - Performance
  - You aren't *really* learning Javascript
  - Black box: debugging JQuery?

MVC on the Web?



MVC Frameworks

- By itself, jQuery doesn't *really* facilitate any form of MVC
- Many MVC frameworks exist (wrapping jQuery):
  - Backbone.js
  - Spine.js
  - Ember.js
  - AngularJS
  - React.js
  - … The list goes on!



React

METEOR

- Each one does "MVC" differently ☹

Summary

- Document Object Model (DOM)
  - Browser Rendering Engine
    - parses HTML to create the DOM
    - utilizes CSS to figure out where DOM nodes should be painted
- Event Handling
  - Managed by Javascript's 3-Step Propagation
  - jQuery can help you write concisely + faster!
- MVC on the web is challenging.