



**FULL SAIL**  
UNIVERSITY

# programming for web applications 2

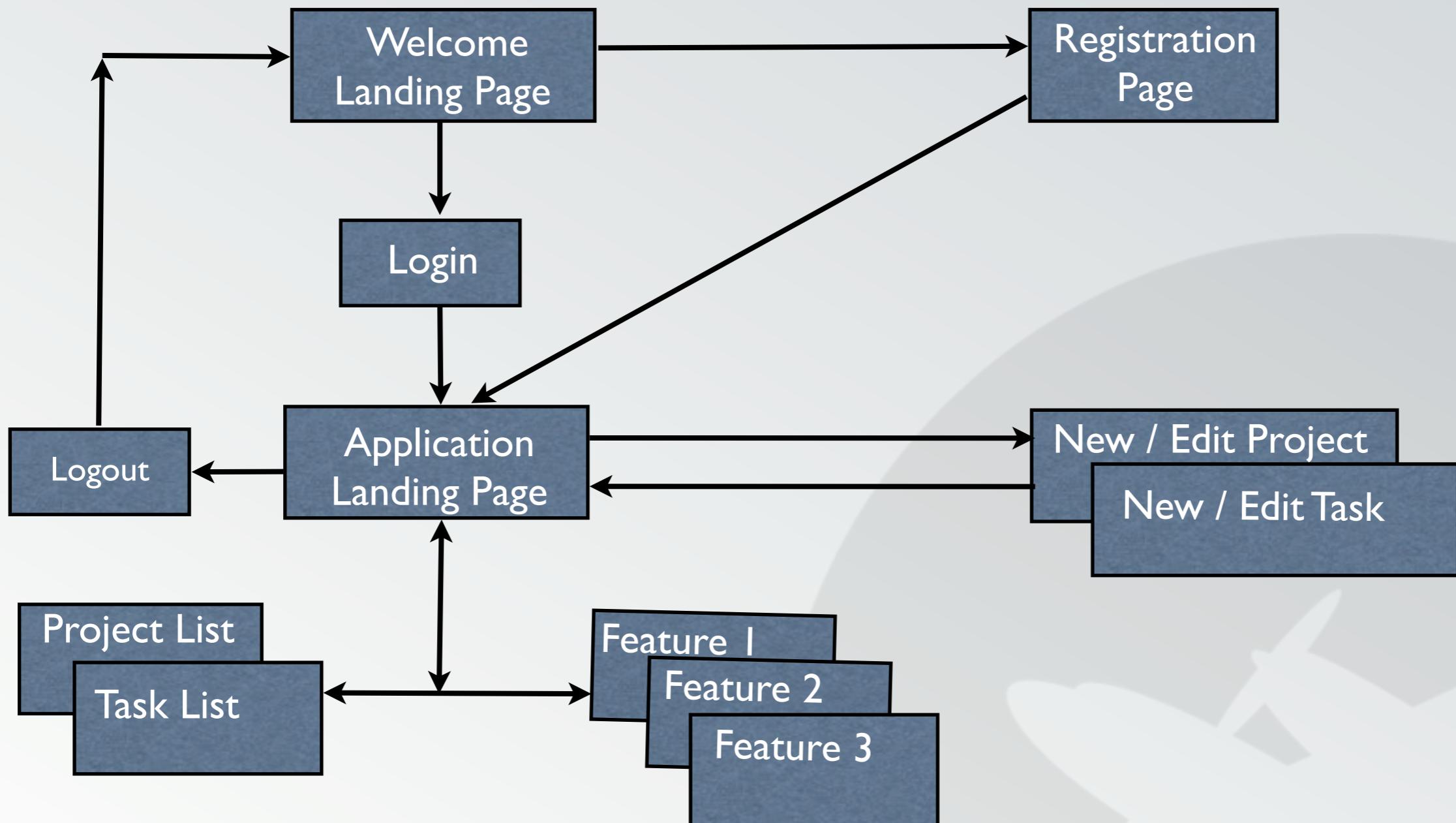


javascript **libraries**

# PWA-2 DUE Dates

Item	Due Dates
Branding / Logo	08/05/13 - After Lab on the First Day
Project Pitch	08/09/13 - Before Lecture on 3
Creative Brief - Finished Document	08/12/13 - Before Lecture 4
Site Prototype ( <i>html/css</i> )	08/16/13 - After Last Lab of the 2nd Week
Development Milestone ( <i>javascript</i> )	08/19/13 - Due End of Lab 7
Inclusion of 5 media center items	08/28/13 - Last Day of Class After Lab
Aesthetics & Usability ( <i>finished site</i> )	08/28/13 - Last Day of Class After Lab
Functionality ( <i>finished site</i> )	08/28/13 - Last Day of Class After Lab
Professionalism	The duration of the course
Class Participation	The duration of the course

# Basic Project Flowchart Example



# JavaScript Libraries

- JavaScript is a fast language, but the DOM is a third-party (inconsistent) browser API.

## DOM API

```
var items = document.qSA('#nav li');
ryu(items).each(function(){
  this.style.display = 'block';
});
```

```
<ul id="nav">
  <li>Home</li>
  <li>Blog</li>
</ul>
```

# JavaScript Libraries

- JavaScript is a fast language, but the DOM is a third-party (inconsistent) browser API.

## DOM API

```
var items = document.querySelectorAll('#nav li');
ryu(items).each(function(){
  this.style.display = 'block';
});
```

```
<ul id="nav">
  <li>Home</li>
  <li>Blog</li>
</ul>
```

jQuery

```
$('#nav li').show();|
```

# Top JavaScript Libraries

- ❖ Last month you explored the concept of creating a library of code.
- ❖ Libraries exist to provide frameworks of reusable code, for the largest amount of common developer needs.
- ❖ Some of the top libraries are:
- ❖ *Dojo, Prototype, Mootools, YUI, Ext, jQuery*

## ‣ Dojo

- Dojo was one of the **first large-scale javascript libraries**, startup around 2004
- Dojo is **considered a Toolkit**, in that it **focuses on widgets**. Most of the widgets provided simulate popular needs in Rich Internet Applications, such as:
  - **form validation, tree displays, dynamic charts, calendar and time selectors**, etc
- Dojo also provides methods for tying into Flash players, managing local data storage, auto-code compression, and Adobe AIR integration.
- Dojo is sponsored by AOL, IBM, Sun, and Sitepen
- <http://dojotoolkit.org/>

## ‣ Dojo

- Like most modern libraries, **Dojo uses a *modular system***, through which the library's core can be extended by including optional js files.
- ***free license ... paid support via Sitepen***
- *Has been gaining alot of steam lately*



## ▶ Prototype

- ▶ Prototype was the **first library to bring massive popularity to the library concept**, with startup around 2005.
- ▶ Prototype is stand-alone, focused on providing **utility**.
- ▶ *Utility frameworks* tend to provide **methods for extremely common needs**, such as accessing and manipulating the DOM/CSS and performing AJAX calls.
- ▶ Prototype focuses on *mimicking* Classes, something that JavaScript normally lacks. Most functionality and plugins for Prototype are based on their class constructor.
- ▶ <http://prototypejs.org/>
- ▶ **Free license**

- ▶ Prototype / script.aculo.us

- ▶ Prototype is also the base of the popular effects library called **scriptaculous**, now on its newest beta as **scripty2**.
- ▶ Provides a massive set of pre-written animation effects and controls:
  - ▶ draggables/droppables, sortables, autocomplete, sliders
  - ▶ appear, slidedown/up, fade, appear, grow, fold, highlight, morph, move, parallel, puff, pulsate, scale, scrollto, shake, shrink, squish, tween, etc..
- ▶ *Was used for the original Lightbox javascript plugin.*
- ▶ <http://script.aculo.us/>
- ▶ **Free license**



## ► Mootools

- Mootools is another stand-alone, very similar to Prototype. It provides **utility functionality for DOM and AJAX scripting, and effects.**
- While Scriptaculous focuses on pre-built effects, Mootool's "FX" effects provides core methods that can be combined, and has the smoothest speeds.
- "FX" was also one of the **first to release color-animation, css-morphing effects, and easing options.**

## ‣ Mootools

- Mootools also provides **widgets**, such as:
- tooltips, slider controls, sortables, drag/drop, accordion, form autocomplete, and mouse effects.
- Mootools was also the **first to release a modular downloader**. Their website **allows you to pick and choose which effects and components you want, and then builds and minifies a custom js file for you**.
- <http://mootools.net/>
- **Core (with no extras) is only 4kb**
- **free license**





## ‣ Ext

- Ext is a highly detailed framework for RIA development. It was **originally developed for use only in Yahoo's YUI**, but now provides tie-ins to jQuery and Prototype, and also has a standalone version.
- Ext's focus is on providing RIA-style widgets for Graphical User Interfaces (known as GUIs). Some of the GUI components are:
  - modal dialog windows, toolbars, tabs, menu systems, regional panel layouts, grid layouts, tree menus, client data management, etc.
- <http://www.sencha.com/products/extjs/>
- ***commercial license***

## ‣ YUI

- Yahoo's javascript UI library (as YUI), is an extensive library **aimed at being all-encompassing**. It provides several utility, effects, and RIA methods such as we've already discussed.
- YUI uses a modular class system for all of its namespacing, similar to Dojo but more intense. While making **YUI highly customizable**, this also means there are dozens of individual files that must be combined for any piece of utility or widget. *YUI 3 makes this process easier.*
- On the YUI team is JavaScript legend **Douglas Crockford**, who has a popular open source compression tool, the “YUI Compressor”.

## ‣ YUI

- YUI's primary focus is open source, highly documented sharing of javascript ideas, and **pushing the boundaries of what's possible in js.**
- Some of their innovations worth looking at: **YQL, YSlow, YUI Theater**
- The **YUI Theater** is a public storage for dozens of lectures and conference talks by the biggest advocates of JavaScript development. *I would recommend watching any videos by Douglas Crockford or John Resig.*
- <http://yuilibrary.com/>
- **free license**

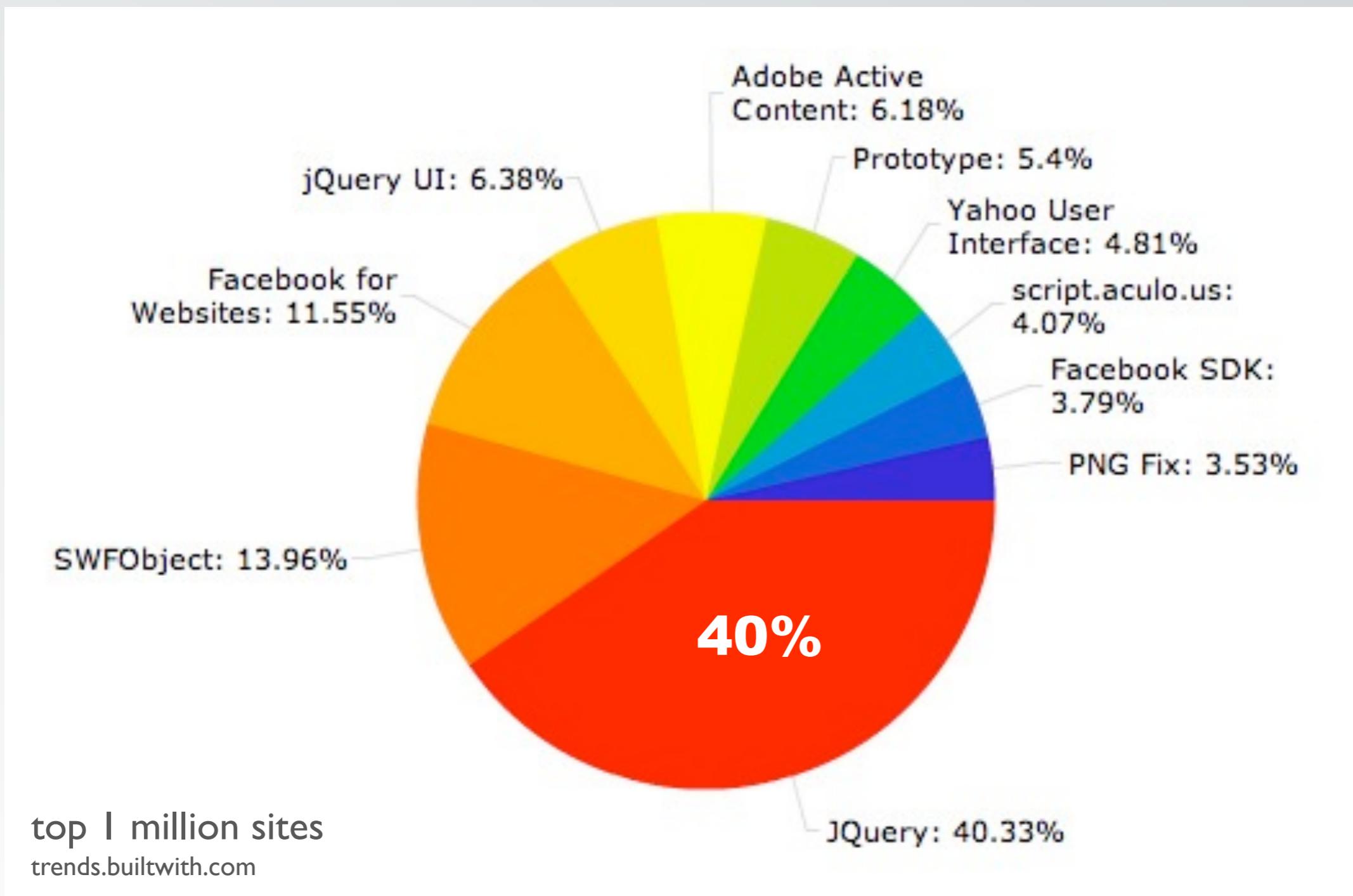




## › jQuery

- › And finally, the Library we'll be focusing on this month is jQuery.
- › **Founded by John Resig of the Mozilla Corporation**, jQuery is a relatively **simple and efficient library**, and currently the **most widely used among developers and designers**.
- › Supporters: Mozilla, Wordpress, Microsoft (.NET), Google.
- › <http://jquery.com/>
- › ***free license, open source on github***

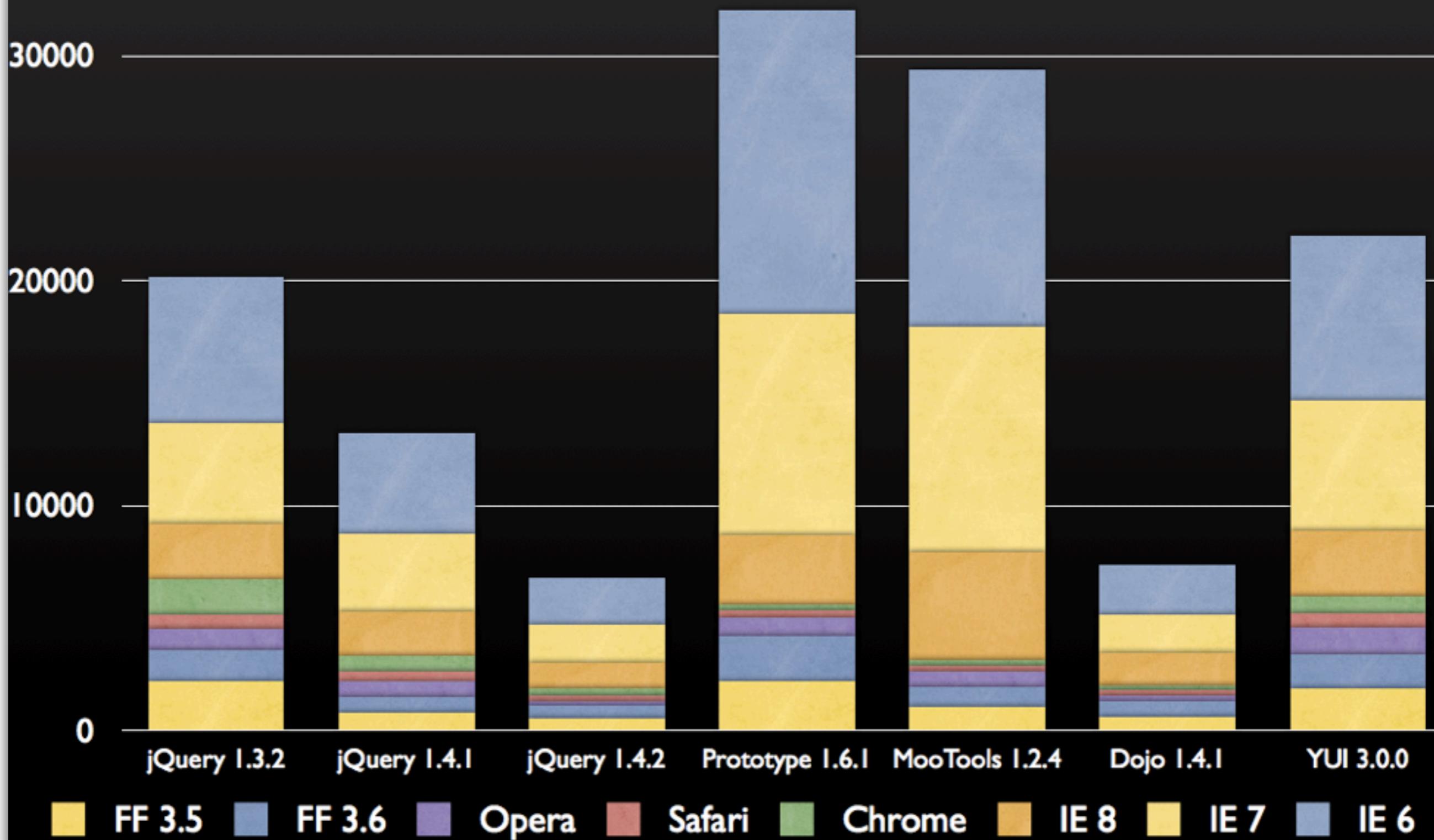
# so why jQuery?





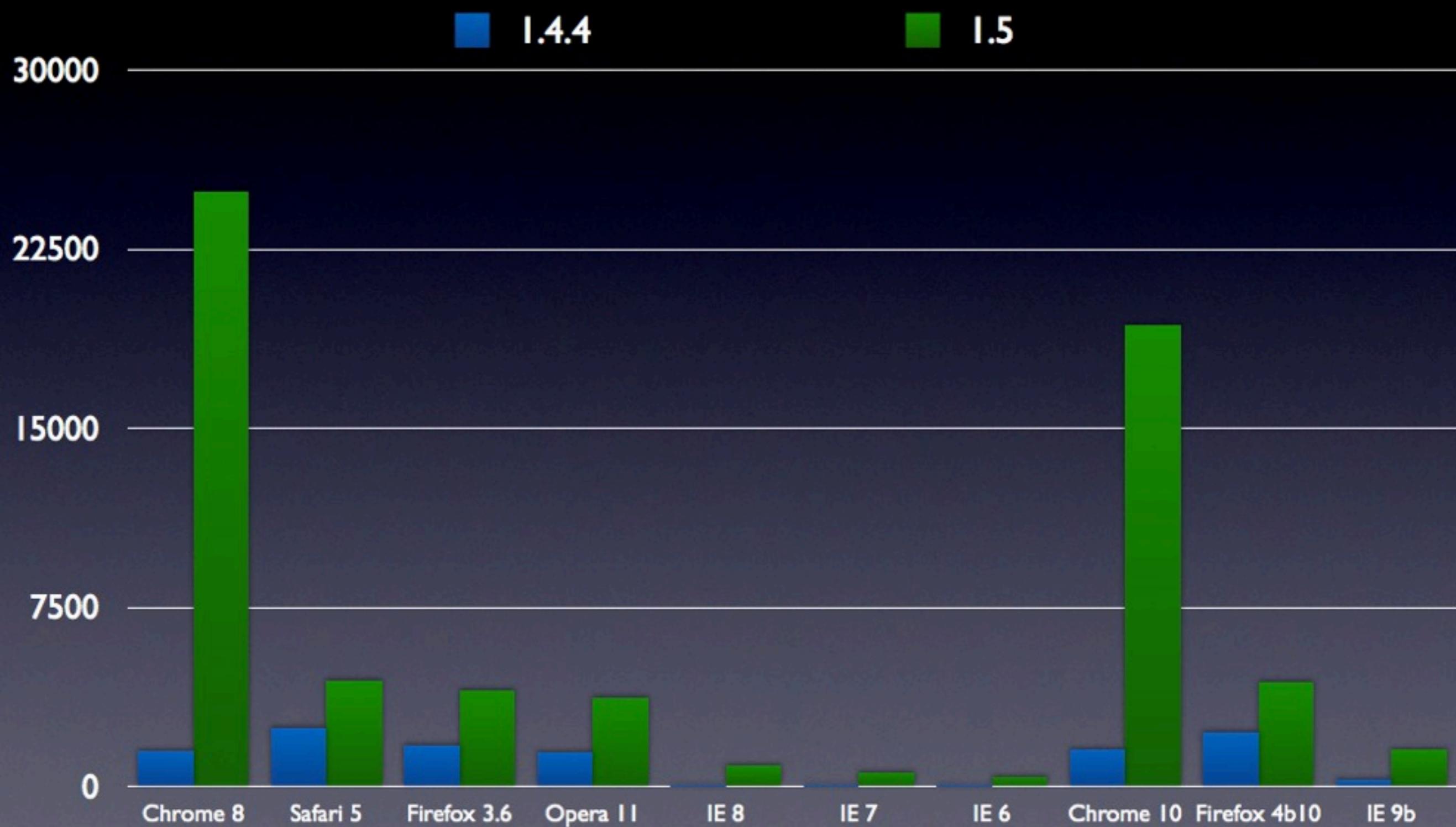
# Taskspeed Results (Time in ms, Smaller is Better)

<http://www.domassistant.com/slickspeed/>



# .children() Performance in jQuery 1.5

(Browser by # of iterations per second, higher is better.)



# let's see some code

*a simplified library comparison*

# W3C JavaScript w/o Any Library

- ❖ Using JavaScript to apply event handlers using modern specs

```
window.addEventListener('DOMContentLoaded', function() {
  document.getElementById('somelink')
    .addEventListener('click', myFn);
});
```

# Mootools

```
window.addEvent('domready', function(){
  $('mylink').addEvent('click', myFn);
});
```



# Prototype

```
document.observe('dom:loaded', function(){
  $('somelink').observe('click', myFn, false);
});
```



# Dojo

```
dojo.addOnLoad(function(){
    dojo.connect(dojo.byId('somelink'), 'click', myFn);
});
```



# Yahoo YUI 2

```
YAHOO.util.Event.onDOMReady(function() {  
    YAHOO.util.Event.addListener(  
        YAHOO.util.Dom.get('somelink'), 'click', myFn, this);  
    );  
});
```



FULL SAIL  
UNIVERSITY

# Yahoo YUI 3

```
YUI().use('*', function(Y){  
    Y.on('domready', function(){  
        Y.on('click', myFn, '#somelink');  
    }, Y);  
});
```



# jQuery

```
$(document).ready(function() {  
    $('#somelink').on('click', myFn);  
});
```





- **Compact:** jQuery weighs in at 72kb (*minified*) or 32kb (*minified/gzip*)
- **Simplicity:** all of jQuery's methods are easy to catch on to:
  - *jQuery.css()*
  - *jQuery.ajax()*
  - *jQuery.html()*
  - *jQuery.animate()*
- **Efficiency:** jQuery highly utilizes OOP coding structures for efficient code, which can turn several lines of code into 1.

go to: [jquery.com](http://jquery.com)

The screenshot shows the official jQuery website. On the left, there's a dark sidebar with the text "jQuery is a new kind of JavaScript Library." followed by a detailed description of what jQuery does. Below this, three features are listed with checkmarks: "Lightweight Footprint", "CSS3 Compliant", and "Cross-browser". On the right, there's a large button labeled "GRAB THE LATEST VERSION!" with a sub-section titled "CHOOSE YOUR COMPRESSION LEVEL:" containing two options: "PRODUCTION (24KB, Minified and Gzipped)" and "DEVELOPMENT (155KB, Uncompressed Code)". Below these is a prominent "Download( jQuery );" button with a downward arrow icon. At the bottom of this section, it says "Current Release: v1.4.2".

download both (*minified and uncompressed*)

# jQuery hosted by Google CDN

- Bookmark the following url:

<http://ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js>

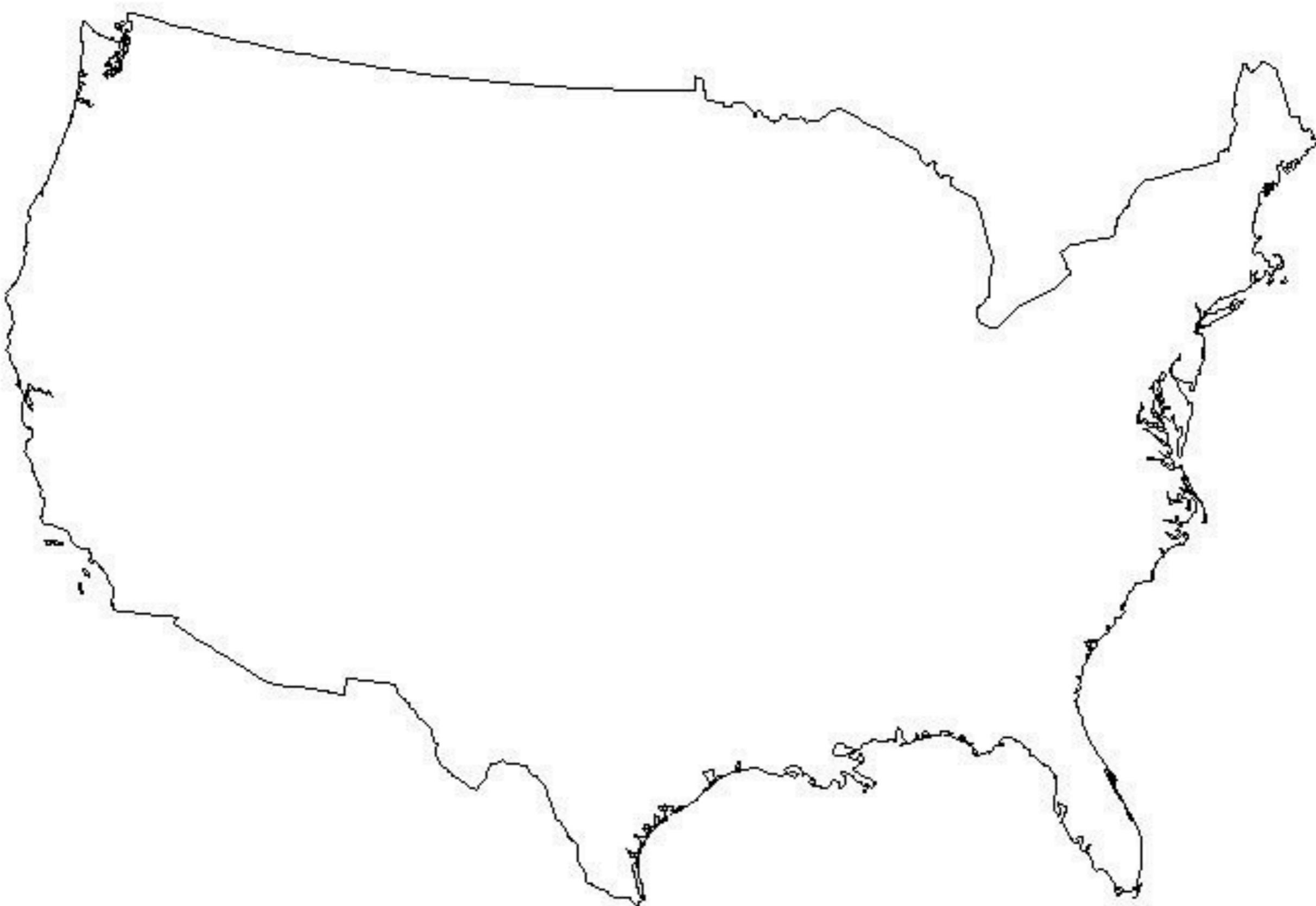
---

- *Has the benefit of **worldwide caching***
- *Required for final turn-in*

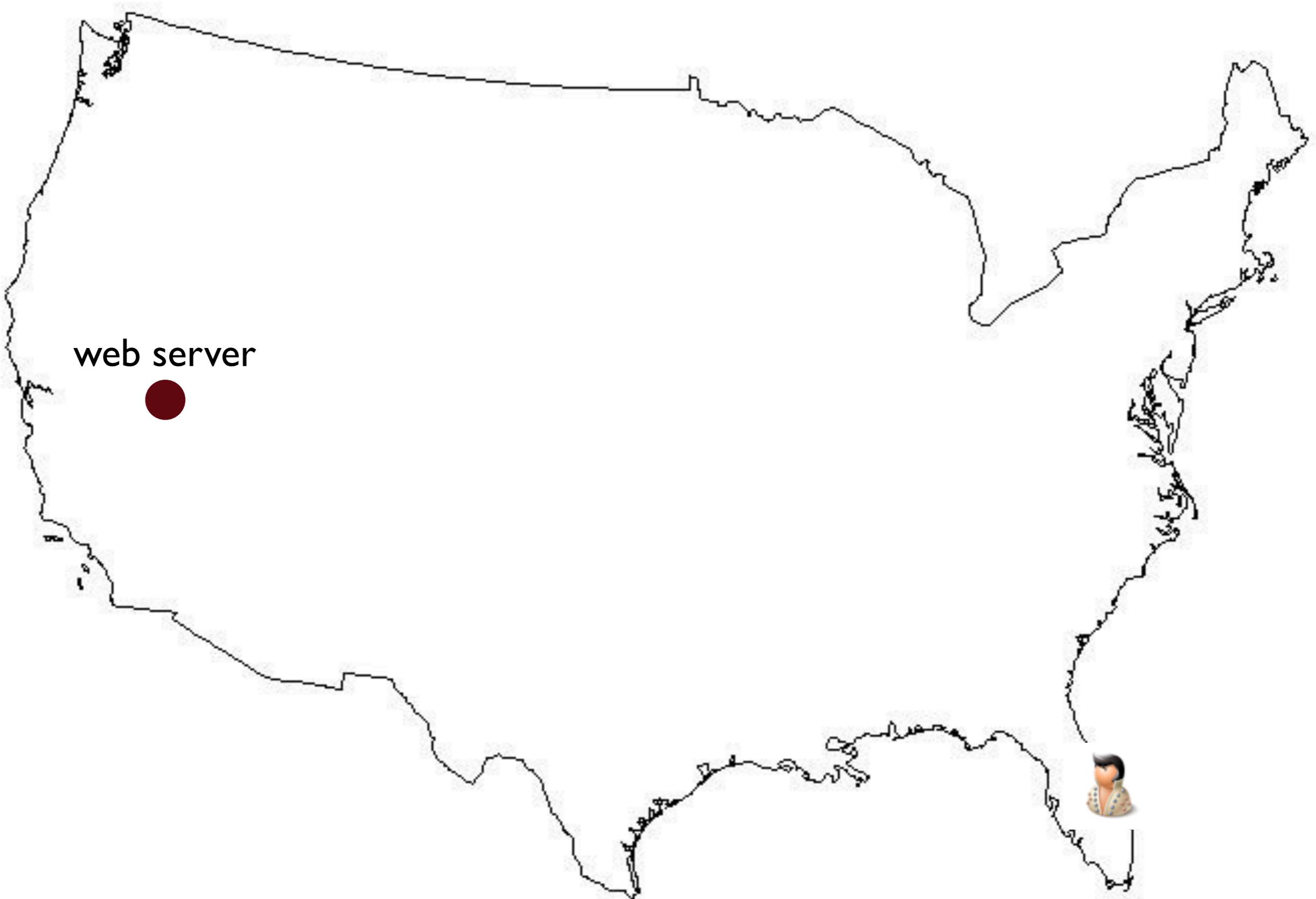
# Content Delivery Networks

- One of the problems with web hosting tends to be bandwidth and the sheer number of requests that a server has to handle.
  - *Especially in large websites, this can cause noticeable server strain, reducing bandwidth speeds for the end user.*
- **A CDN is a network of servers that share your content, allowing you to push off some of the strain of your own server.** Most are pay-services, such as Amazon's CloudFront CDN, or Rackspace.

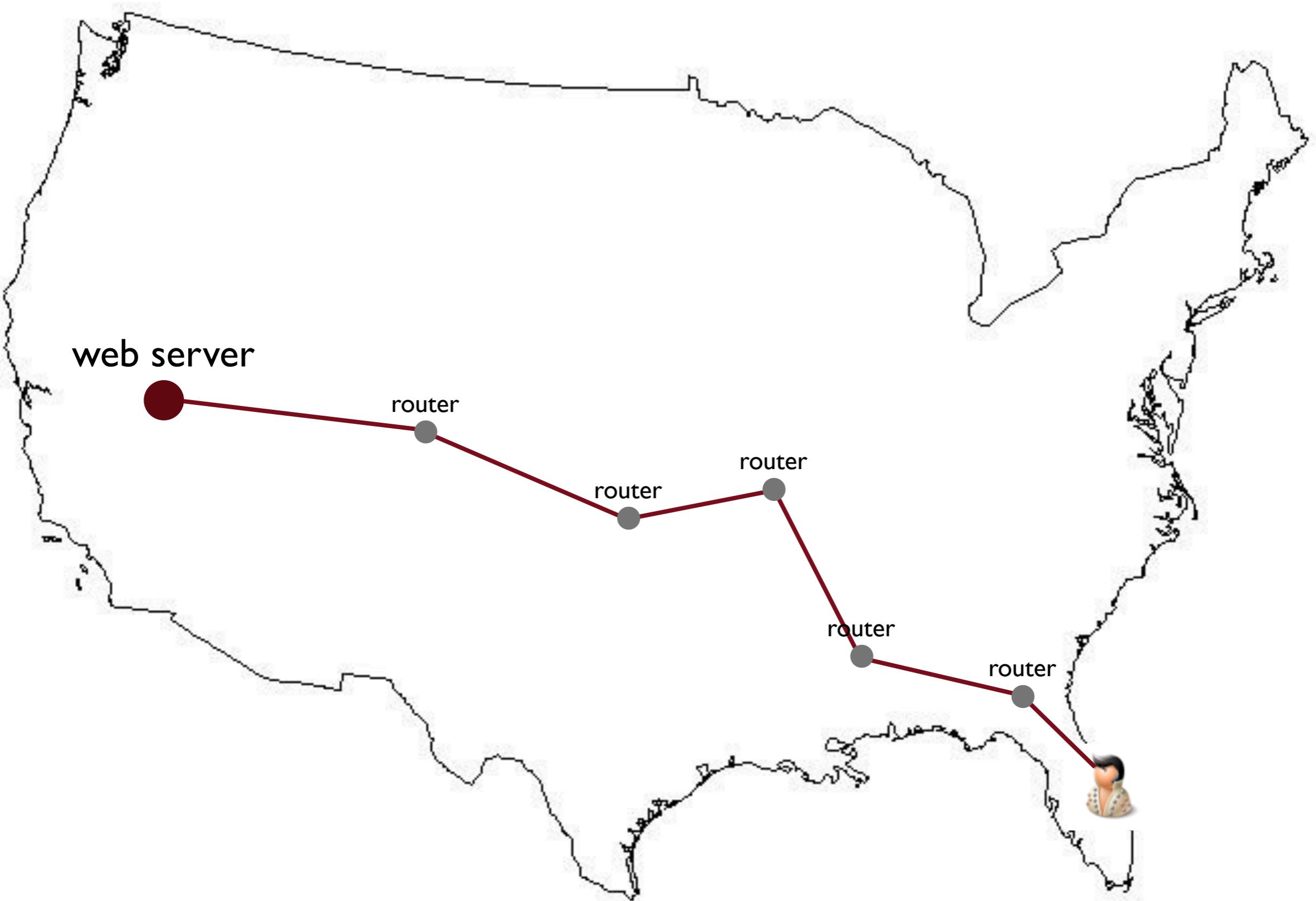
# Normal Web Request



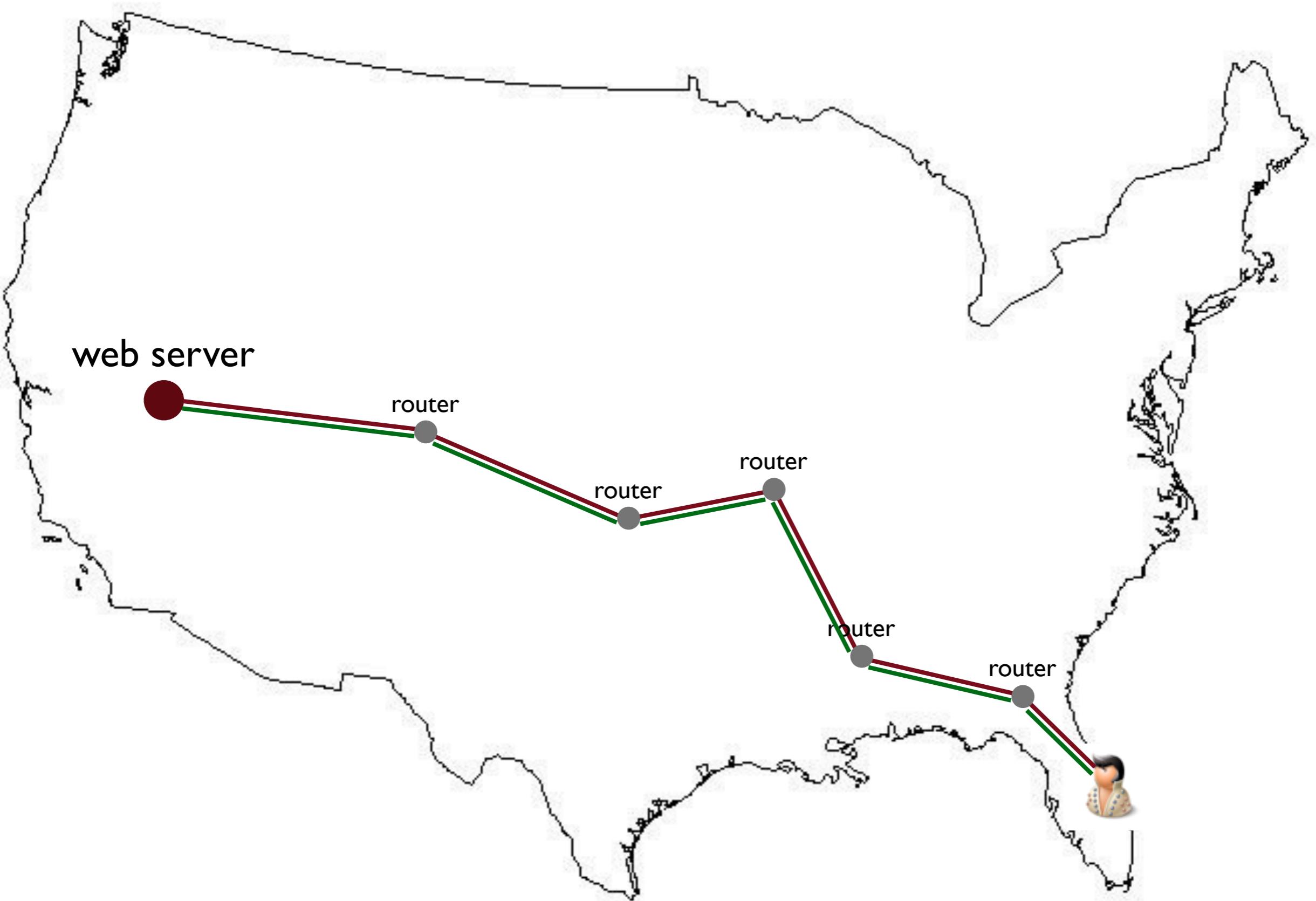
# Normal Web Request



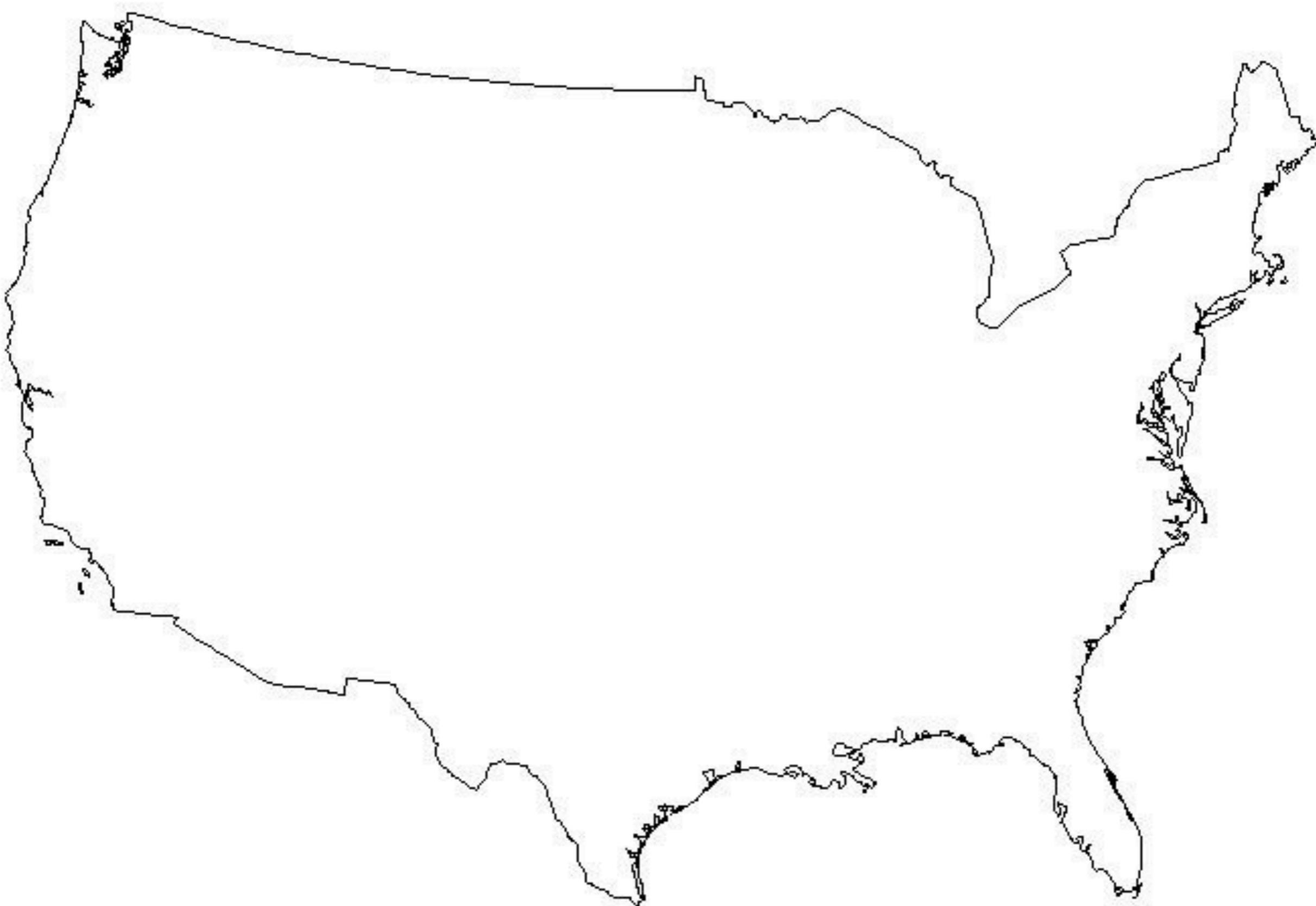
# Normal Web Request



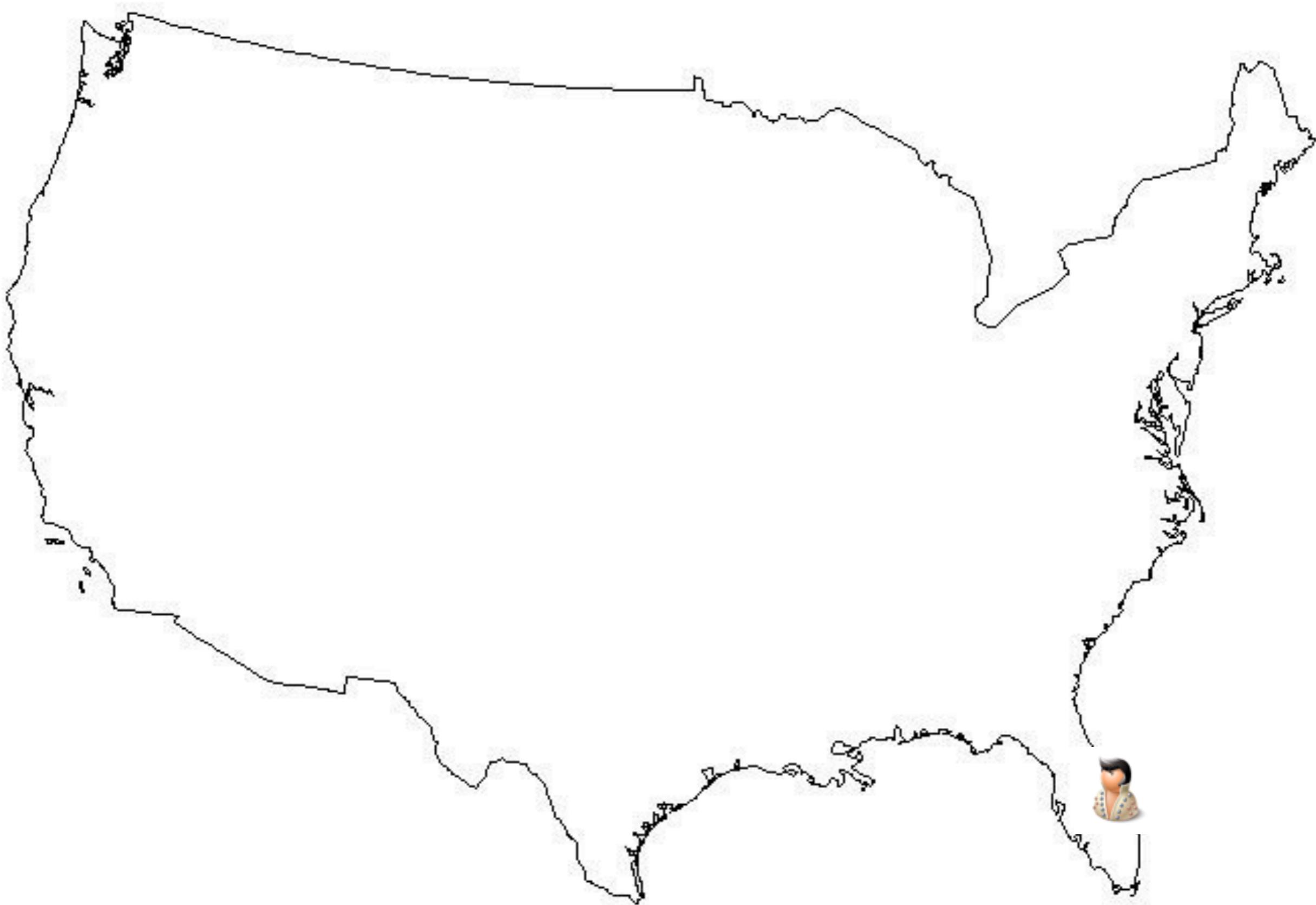
# Normal Web Request



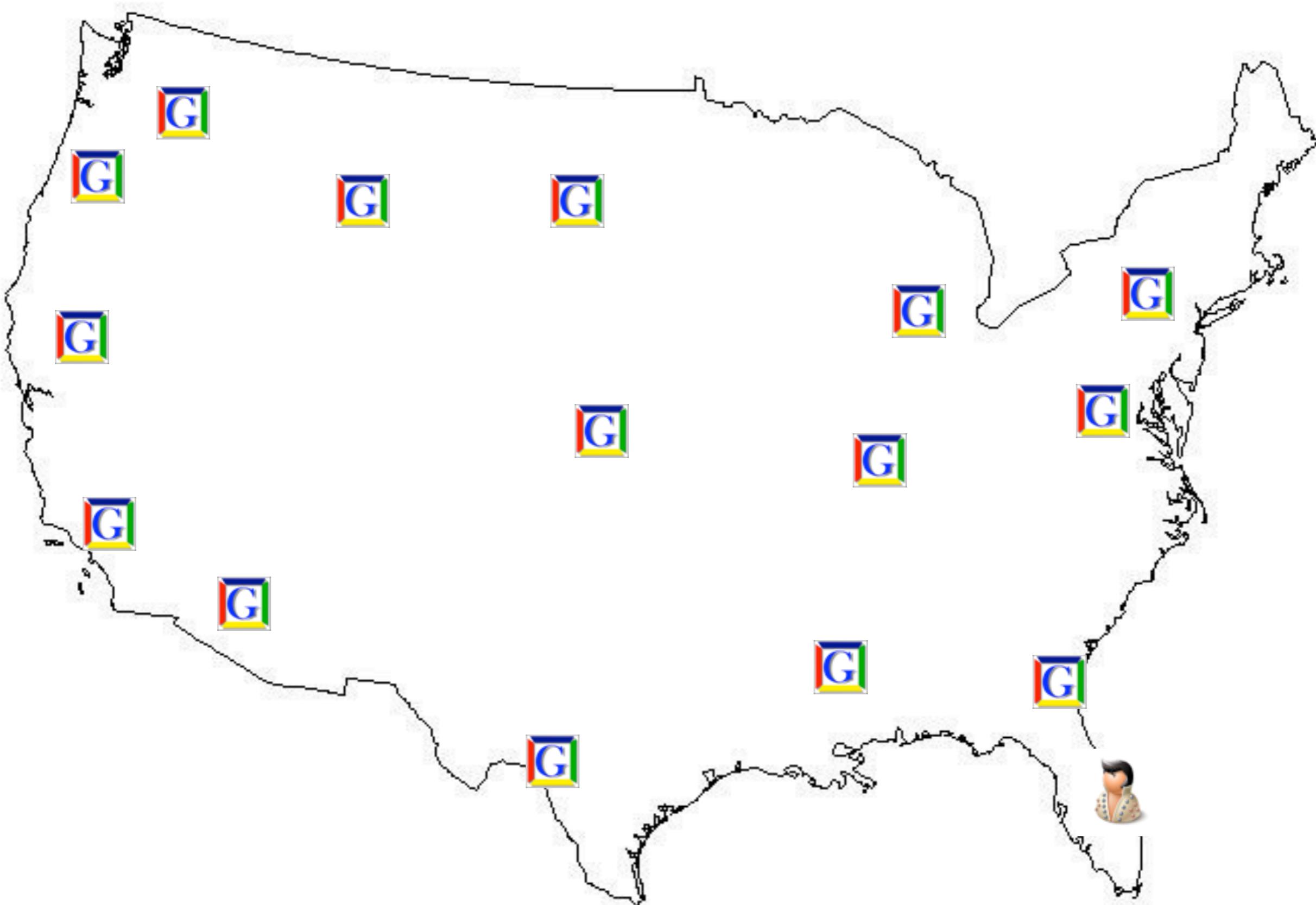
# Google CDN



# Google CDN



# Google CDN



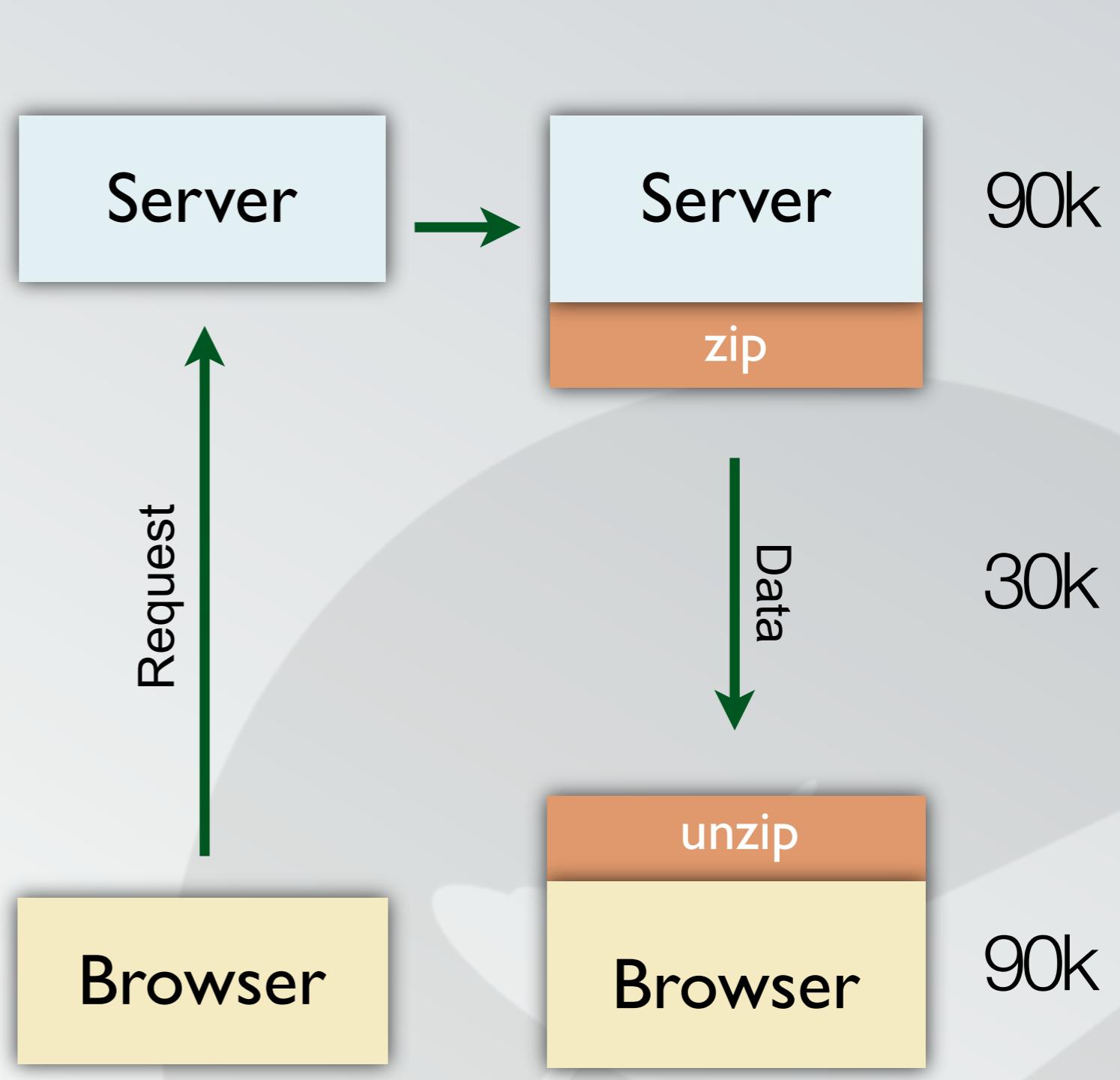
# GZip Compression

*GZip is a compression format similar to zipping, but for single files.*

*The server can be instructed to gzip any files it passes to the browser, making transfers faster.*

*The browser has the engine to uncompress the file and then read it as normal.*

- example...



# GZip Compression

*Since most media files are already compressed, usually the only things needed zipped are html, css, and javascript.*

*The easiest way with an apache server is to add an htaccess file*

```
<FilesMatch "\.(php|html|css|js)$">  
SetOutputFilter DEFLATE  
</FilesMatch>
```

.htaccess

# Recommended jQuery Resources:

Website	Description
wddbs.com/javascript/sfw/campus	JS / jQuery Lectures
api.jquery.com	Official Documentation
yayjquery.com	Best jQuery Podcast Ever...
podcast.jquery.com	Official Podcast

Twitter	Person
jeresig	John Resig <i>founder of jQuery, duh.</i>
cowboy	Ben Alman ( <a href="http://benalman.com">benalman.com</a> ) <i>amazing plugin developer</i>
bennadel	Ben Nadel ( <a href="http://bennadel.com">bennadel.com</a> ) <i>jquery / coldfusion developer</i>
paul_irish	Paul Irish ( <a href="http://paulirish.com">paulirish.com</a> ) <i>google / jquery developer</i>
rem	Remy Sharp ( <a href="http://remysharp.com">remysharp.com</a> ) <i>js / jquery / mobile developer</i>
codylindley	Cody Lindley ( <a href="http://codylindley.com">codylindley.com</a> ) <i>core js / prototype developer</i>

IRC	Description
#jQuery on irc.freenode.net	Official jQuery Chat ( <i>free software: colloquy</i> )

# Course Review: jQuery Namespace

## › jQuery Namespace

- › ALL javascript libraries has a namespace to contain their functionality. This is a forward-thinking developer initiative so that multiple frameworks can occupy the same global space without causing browser conflicts.
- › Keep in mind that in JavaScript, **a namespace is simply GLOBAL object, with methods and properties.**

## › jQuery Namespace

- › jQuery namespace **GLOBAL object is represented using: jQuery**
- › jQuery also provides a shortcut to itself, as **\$**

```
$("#superman").addClass("sun");  
jQuery("#superman").addClass("sun");
```

- › **\$** is a more efficient identifier, so you'll notice it is used in nearly all documentation and tutorials, and among most developers.
- › The only conflict that arises is if either *Mootools* or *Prototype* are also used in the same page, since all 3 libraries utilize a global **\$** in some way.

## › jQuery Namespace

```
$ (“#superman”).addClass(“sun”);
```

- › So.. if **jQuery** and **\$** are the global namespace objects, why are they being executed as functions? ie.. `$()`
- › Remember that objects in JavaScript are very loose, they can be arrays, objects, or functions at the same time.
- › The genius is that jQuery is all of the above! (*an object, an array, and a function*)

## ‣ jQuery Namespace

- As an object, \$ has methods...

**\$ .each()**

- As a function, we passed arguments that result in a *selector target*:

**\$ (“#superman”) .addClass(“sun”);**



## Utility methods

```
$ .support()  
$ .each()  
$ .extend()  
$ .merge()  
$ .trim()  
...  
...
```

## Factory methods

```
$ () .addClass()  
$ () .css()  
$ () .animate()  
$ () .html()  
$ () .slideUp()  
...  
...
```



# Course Review: How jQuery Works

## › How jQuery Works

- › Almost any line of code we write will begin with a *target*, using jQuery's **Factory**

```
$ (“#mydiv”).animate();
```

← ***jQuery’s Object Factory***

- › A *factory* is an object-oriented design pattern, that can **most closely be described as an object creator** (*like a Class*), although more dynamic.

## ▶ How jQuery Works

- ▶ jQuery factory arguments: #ID, .class, or HTML Tag
- ▶ Most jQuery methods will return the factory object for chaining, unless a different data type is specified
- ▶ CSS *methods*: .css() .attr() .height() .offset()

```
$function(){
  $("#target").css({border:"2px"});
});
```



factory target      jQuery method

## › How jQuery Works

```
$(".superman").addClass("sun").fadeIn().slideUp();
```

- › What's actually happening is that the `$` factory is **first finding the element based on the *selector argument*, and then creating and returning a new internal object.** This **new internal object inherits all of the methods of the jQuery library.**
- › We can see the `.addClass` method being used. It's important to note that this method of creation is **NOT attaching methods to the DOM element, it is creating an internal object as a reference.**

## ▶ How jQuery Works

```
$(".superman").addClass("sun").fadeIn().slideUp();
```

- ▶ The next important thing to note is that almost all methods in jQuery return the jQuery object to itself.
- ▶ So, when **.addClass** is called, it uses the `$("superman")` factory object as a target, and returns the factory object (*remember that functions can **return** any data type*).
- ▶ This is why **fadeIn** and **slideUp** are able to be called, because each new method call is re-returning the factory object from the beginning of the code line.

## ▶ How jQuery Works

- ▶ You'll notice that jQuery's documentation at <http://api.jquery.com> will always indicate what the return value of any method is.

Most will return the jQuery object, which means they can be chained.

Others may return a specific value (such as an integer or string)

- ▶ *Cheat sheet available on studentvfiler*

Returns: [Integer](#)

of the first matched element.

Returns: [jQuery](#)

ment.

Returns: [Integer](#)

of the first matched element.

Returns: [jQuery](#)

ment.

Returns: [Integer](#)

ler and includes the padding) for



# jQuery Fundamentals

## › The “dom ready” callback

```
\$(function() {  
    // all site code  
});
```

```
\$(document).ready(function() {  
    // all site code  
});
```

- › This argument is a nameless function literal. It should already look familiar, as this is how our **DOMReady** method worked last month.
- › In the same manner, any code inside this function is run when the DOM is loaded.

## ▶ Using \$

- ▶ Your logical next question should be, what arguments does the **\$ factory function** take in?
- ▶ *There are 4 possibilities:*
  - ▶ **CSS selector expressions**
  - ▶ **DOM element objects**
  - ▶ **HTML strings**
  - ▶ **or a callback function**

# jQuery Fundamentals

## ▶ CSS Selector Expressions

```
$("#id > .class")
```

- ▶ The above is an example of a CSS selector method.
- ▶ In this parameter, a string is passed that contains references to css selectors, combined with some special selector characters that jQuery defines. *Note that **selectors must be strings, using quotes.***
- ▶ This returns the factory-created library object, containing a reference to the element you were searching for that is used as a target.

# jQuery Fundamentals

## ▶ CSS Selector Expressions

```
$(".myclass").slideUp();
```

- ▶ The next important thing to note relates to the above example. Notice that this selector string is targeting all elements with a specific css class name. In this, the factory works the same way, only it **creates an array of objects**, just like how **getElementsByTagName** works.
- ▶ The **benefit here is that any jQuery methods used on this wrapper will automatically be applied to each matching element.**
  - ▶ If there were 4 divs with a class of “myclass” in this example, all 4 would animate up at the same time, with this one line of code.



FULL SAIL  
UNIVERSITY

# jQuery Fundamentals

## ▶ DOM Element Objects

- ▶ The factory can also be used in targeting DOM elements directly, by their normal object names... (*notice that strings are not being used, only object names*)
  - ▶ *Keep “window” and “document.body” in mind for later...*

```
$ (window)...
```

```
$ (document.body)...
```

- ▶ *also...*

```
var el = document.getElementsByTagName('div');  
$(el)...
```

# jQuery Fundamentals

## ▶ HTML Strings

```
$('<a href="">Click me</a>')
```

- ▶ This option of the `$` factory will ***create a new block of html code***, and store it by creating a factory object as normal, with all of the methods of the library attached.
- ▶ It's important to note that, while **this creates an object of html code, it does NOT insert into the DOM**, until you tell it to... for which there are jQuery methods, of course.

# DOM Manipulation

## › jQuery vs JavaScript

- › Always remember that the jQuery factory is returning an *object*.
- › Once we enter a ***jQuery chain***, most raw JavaScript will not be accessible.

```
$ (“#mydiv”).innerHTML=“hi”;
```

← ***jQuery’s Object Factory***

- › innerHTML is a property of DOM objects, not a property of the jQuery object.



**does nothing**

jQuery stops most errors



# DOM Manipulation

## ‣ What's To Come

- Most of the functionality of jQuery is well documented in their website, and broken up by categories of toolsets which we'll be covering in-depth:
  - **selectors** (factory targets)
  - **element manipulation, walking, and attributes**
  - **events**
  - **animations and effects**
  - **ajax methods**
  - and for today's finish, **css**.

# DOM Manipulation

## ▶ Testing Elements

- ▶ Since jQuery won't throw errors on bad selectors, sometimes you may need to test.
- ▶ jQuery objects come with a **.length** property

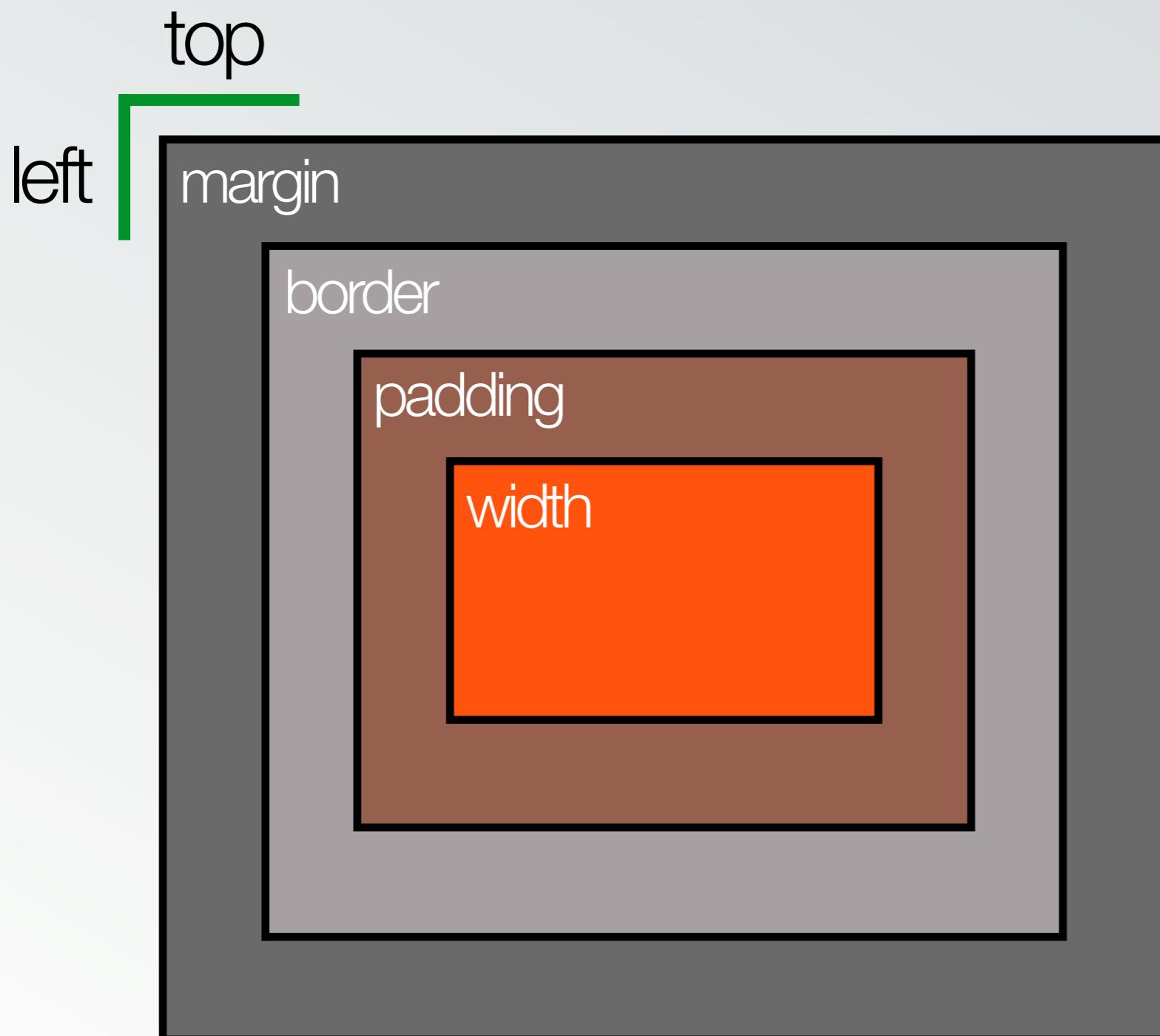
```
var nav = $('#nav');
if( nav.length === 0 ){
    // elements don't exist
};
```

# css, classes, and attributes

# jquery css, classes, and attributes

# The Box Model

size & position



web design and development  
bachelor of science degree program

# Size Methods

Method	Returns	Description
<code>\$().height()</code>	<code>number</code>	Returns the numeric size ( <i>not including padding/border</i> )
<code>\$().width()</code>	<code>number</code>	...
<code>\$().innerHeight()</code>	<code>number</code>	Returns size including padding but not border
<code>\$().innerWidth()</code>	<code>number</code>	...
<code>\$().outerHeight(bool)</code>	<code>number</code>	Returns size including padding AND border. If the <i>boolean</i> is <i>true</i> , the value also includes the margin.
<code>\$().outerWidth(bool)</code>	<code>number</code>	...

<http://jsfiddle.net/user/sfw2/>

# Position Methods

Method	Returns	Description
<code>\$.position()</code>	object	Returns <code>top</code> and <code>left</code> properties relative to the first offset parent of the given element.
<code>\$.offset()</code>	object	Returns <code>top</code> and <code>left</code> properties relative to the entire document for the given element.

```
var ptop = $('div').position().top;           ex: 100  
var pleft = $('div').position().left;
```

```
var otop = $('div').offset().top;  
var oleft = $('div').offset().left;
```

# Location Shortcuts

Method	Returns	Description
<code>\$.scrollTop()</code>	number	Returns vertical scroll position of given element.
<code>\$.scrollLeft()</code>	number	Returns horizontal scroll position of given element.
<code>\$.scrollTop(number)</code>	jQuery	Changes the vertical scroll position of given element.
<code>\$.scrollLeft(number)</code>	jQuery	Changes the horizontal scroll position of given element.

```
var btop = $('div').scrollTop();
$('div').scrollLeft(50);
```

# Attribute Manipulator

Method	Returns	Description
<code>\$.attr(string)</code>	string	Returns the value of requested html attribute.
<code>\$.attr(string, string)</code>	jQuery	Changes a single html attribute on all given elements.
<code>\$.attr(object)</code>	jQuery	Changes multiple html attributes on all given elements.

```
var navalt = $('a').attr("alt");
$('a').attr("href", "http://google.com");

$('a').attr({
  title: "link",
  href: "http://google.com",
  alt: "Google"
});
```

# Attribute Manipulator

Method	Returns	Description
<code>\$( ).removeAttr(string)</code>	jQuery	Removes the given attribute from the element.

```
<a alt="not for long"></a>
```

```
$('a').removeAttr("alt");
```

```
<a></a>
```

# CSS Manipulator

Method	Returns	Description
<code>\$(()).css(string)</code>	string	Returns the value of requested css property.
<code>\$(()).css(string, string)</code>	jQuery	Changes a single css property on all given elements.
<code>\$(()).css(object)</code>	jQuery	Changes multiple css properties on all given elements.

```
var padbot = $('#nav').css("paddingBottom");
$('#nav').css("backgroundColor", "#ff0000");

$('#nav').css({
  backgroundColor: "#ffffff",
  height: 100,
  padding: "5px"
});
```

# Class Manipulators

Method	Returns	Description
<code>\$(()).addClass(string)</code>	jQuery	Adds the class to all given elements.
<code>\$(()).removeClass(string)</code>	jQuery	Removes the class from all given elements.
<code>\$(()).toggleClass(string)</code>	jQuery	Toggles the given css class.
<code>\$(()).hasClass(string)</code>	boolean	If any of the elements have this class, returns true.

```
$('div').addClass("myClass");  
$('div').addClass("myClass anotherClass");  
$('div').removeClass("myClass");  
$('div').toggleClass("myClass");  
var bool = $('div').hasClass("myClass");
```

# Form Value Manipulator

Method	Returns	Description
<code>\$.val( )</code>	string	Returns the value of form input or textarea
<code>\$.val(string)</code>	jQuery	Sets the value of a form input or textarea

```
<input type="text" id="username" value="" />
```

getter      `var username = $("#username").val();`

setter      `$("#username").val('Enter username');`

# javascript vs. jQuery



**FULL SAIL**  
UNIVERSITY.

# js: performance tips



# Performance

- ▶ Always use CSS classes for everything.
- ▶ Only use IDs in HTML for high-level singular targets (header, footer, nav, etc)
- ▶ Try to reduce DOM calls (bad DOM touching) when possible
- ▶ Use addClass and removeClass when possible
- ▶ JavaScript can be hard to debug, use console.log when in doubt



# Lecture Activity

*studentvfile/sfw2/Lecture Activities/day2*

*optional: use the ‘cheat sheet’ for quick reference*

- **Continue to work on your Project Pitch & Creative Brief**
  - *I recommend having your main application design done by the end of lab*
- **Next Milestone: Project Pitch & Draft of your Creative Brief**
  - **must have:**
    1. Your Project Pitch Video COMPLETED
    2. Your Creative Brief COMPLETED
    3. Turn your deliverable into your PWA2 Repo.
      - Name your video file: “**lastname\_firstname\_pitch.mov**”
      - Name your document file: “**lastname\_firstname\_pitch.pdf**”