Javascript, part 1

CS147L Lecture 4
Mike Krieger

Intro

Welcome back!

By the end of today...

- Understand Javascript's syntax
- Get to know jQuery
- How to get DOM element using Javascript
- How to create HTML elements and add them to the DOM
- How to attach events to objects

Our first JS script

```
<html>
<head>
   <script>
      alert("Hello, world.");
   </script>
</head>
<body>
</body>
```

A brief history of JS

Origins

- Created in 1995 by Brendan Eich for Netscape
- Named “**Java**Script” to capture Java's momentum at the time
- Microsoft implements JScript soon after

Since then...

- Incredibly popular
- Has gotten a bit of a bad rap
- Most problems are with browser implementations, rather than language itself

Javascript in 2009

- Toolkits: jQuery, Dojo, Prototype, Yahoo! User Interface library
- Plugins for these toolkits, like jQuery Touch

JS Basics

Variable assignment

```
//assigning a variable
var a = "Hello";
var b = 5;

// comment one line with two slashes
/* comment
   blocks
   with slashes and
   stars
/*
```

For loops

```
/* for loop */
for (var i = 0; i < 10; i++) {
   // do something
}

/* for each loop, useful for arrays*/
for (var i in obj) {
   alert(obj[i]);
   // i is the name of the property,
   // *not* the object represented
   // by that property
}
```

Defining functions

```
function doStuff(argument1, argument2...) {
   alert(argument1);
}

or in line:
function doStuff(argument1) {
   var f = function() {
      // function inside a function
   }
   f(); // calling a function
}
```

Javascript is Loosely Typed

- Everything is assigned to **var**s
- Be careful: ("5" + 10) = 510 (the 10 is coerced to a string)

Coercing types

```
var a = 5;
var b = "10"; // came from somewhere else
a + b;
... 510
a + parseInt(b);
... 15
b = "10.5";
parseInt(b) -> 10
parseFloat(b) -> 10.5
a.toString() -> "5"
```

Booleans

```
true and false

var a = 5;
var b = 5;
a == b;
>>> true

var a = 1;
var b = true;
a == b;
>>> true

a === b;
>>> false
```

== is a “loose” comparison

=== is strict

Objects

- Objects are the building blocks for everything else in Javascript
- Reference properties inside objects with a period (obj.propertyname) or brackets: obj['propertyname']

Objects

```
var a = {
   property1: "value",
   property2: "othervalue"
}
alert(a.property1); -> "value"
a.property2 = "what?";
alert(a.property2); -> "what?"
alert(a['property2']); -> "what?"
```

Arrays

- Ordered lists
- Grows dynamically as things are added to it
- Can have any number of different things in it (numbers, strings, objects...)

Arrays Syntax

```
var a = []; //creates an empty Array
a.push("hi");
console.log(a); -> ["hi"];
var b = a.pop();
console.log(b); -> "hi";
console.log(a); -> [];
a.push("okay");
console.log(a[0]); -> "okay"
var c = {
    propname: "value";
}
a.push(c);
console.log(a); -> ["okay", {propname:"value"}];
```

Going through elements

```
var arr = [1,2,5,7,10];
for(var i = 0; i < arr.length; i++) {
   // do something with arr[i]
}

// common pattern!
```

Variable scope

- In browser, global scope is the **window**
- functions define new scope
- **var** declares variable in that scope
- Scope means: *what variables can I access at this point in the app?*

The scope chain

```
// define a new variable a, in the current scope
// which, since we're not in a function, is window
var a = 5;

// get a variable a, in the current scope or a parent
//scope
console.log(a);
>>> 5

console.log(window.a);
>>> 5

function newFunction(){
    alert(a); // refers to global a
    var b = 3; // creates b in newFunctions' scope
}
console.log(b); // undefined, because b is out of scope
```

Scope exercises

```
var a = 5;
var b = 10;
var x = function() {
    var a = 10;
    b = 5;
}
x();
```

what does *a* equal after x runs?

Scope exercises

```
var a = 5;
var b = 10;
var x = function() {
    var a = 10;
    b = 5;
}
x();
```

what does *a* equal after x runs?

Answer: 5

Scope exercises

```
var a = 5;
var b = 10;
var x = function() {
    var a = 10;
    b = 5;
}
x();
```

what does *b* equal after x runs?

Scope exercises

```
var a = 5;
var b = 10;
var x = function() {
    var a = 10;
    b = 5;
}
x();
```

what does *b* equal after x runs?

Answer: 5

What the browser does

- Checks the currently running function for a declaration of the requested variable
- If not found, go up one in the scope chain, until we hit the **window**

Object-oriented?

- Javascript is Object-oriented, but not class-oriented
- Instead, uses **prototypical inheritance**

Creating instances

```
function Building(location) {
   this.location = location;
   this.getLocation = function(){
      return this.location;
   }
}
// calling a function with new treats
// the called function as a prototype
// and makes a new instance
var Gates = new Building("353 Serra");
```

Changing prototype on the fly

```
var Gates = new Building("353 Serra");
Building.prototype.getStreet = function() {
   return this.location.split(" ")[1];
}

Gates.getStreet();
>> "Serra"
// prototypes can be changed on the fly,
// and anyone based on that prototype will
// pick up the changes
```

What is 'this'?

- Defines the **context** that a function is being executed in

Examples

```
this;
>> DOMWindow
Building.prototype.getThis = function(){
   return this;
}

Gates.getThis();
>> Gates
```

Functions are just objects

```
window.getThis = Gates.getThis;
// now there's a "getThis" function defined
// on the window
getThis();
```

Functions are just objects

```
window.getThis = Gates.getThis;
// now there's a "getThis" function defined
// on the window
getThis();
>> DOMWindow
```

This means...

- Be very aware when using "this" inside your functions

More on 'this'

- We'll see more of this later today when we do events

Timing events

```
// execute one function, delayed:
window.setTimeout( function, delayInMilliseconds);

// execute a function at regular intervals:
var timer = window.setInterval(function, delayBetweenInMillis);

// stop the timer
window.clearInterval(timer);
```

Example

```
function onLoad() {
    window.setTimeout(function(){
        var dv = document.createElement("div");
        dv.innerHTML = "created by setTimeout";
        document.body.appendChild(dv);
    }, 5000); //add something after 5 seconds

    window.setInterval(function(){
        var dv = document.createElement("div");
        dv.innerHTML = "created by setInterval";
        document.body.appendChild(dv);
    }, 500) // add something every half a second

}

window.onload = onLoad;
```

Demo

`timers.html`

Javascript in the Browser

the <script> tag

```
<head>
   <script>
   // your in-file JS
   </script>
   <script src="jsfile.js"></script>
</head>
```

When is JS executed?

- As it occurs in the DOM
- So, in the <head> you don't have access to any of the elements in your body at first
- Solution? hooking up to **onload**

Hook up to onload using JS

```
<head>
<script>
   function onloadActions() {
      // hook up events, etc;
   }
   window.onload = onloadActions;
</script>
</head>
<body>...
```

jQuery

Javascript Frameworks

- Abstract away common functions that are slightly different across browsers
- Simplify common tasks like showing & hiding elements
- Help build features like a tab menu, a “CoverFlow” type menu, drag and drop

Why jQuery?

- Good fit for our class—syntax is very CSS-selector based
- We'll use jQueryTouch plugin next week

If you're interested...

- We can cover Javascript “guts” in the last week of class

jQuery crash course

- Global jQuery() function that selects elements to act on
- Shortcut $( ) function that we'll use instead

Interacting with the DOM

CSS Flashback

- **#***name* (selects by id)
- **.***name* (selects by class)
- *tagname* (selects by tag)

CSS Flashback

- **#***name* -> $("#name");
- **.***name* -> $(".name");
- *tagname* (selects by tag) -> $("tagname");

Onload function

```
$(document); // selects the whole document
$(document).ready( func ); // sets func to be executed onload
```

Selecting By Id

```
<head>
    <script src="../jquery.js" type="text/javascript"
charset="utf-8"></script>
    <script>
    function onloadActions(){
        var a = $("#selectme");
        console.log(a);
    }
    $(document).ready(onloadActions);
    </script>
</head>
<body>
    <div id="selectme"></div>
</body>
```

< jQuery object

By Class

```
<head>
    <script src="../jquery.js" type="text/javascript"
charset="utf-8"></script>
    <script>
    function onloadActions(){
        var a = $(".main");
        console.log(a);
    }
    $(document).ready(onloadActions);
    </script>
</head>
<body>
    <div class="main" id="selectme"></div>
    <div class="main" id="selectmetoo"></div>
</body>
```

```
▼ Object
  ▶ 0: HTMLDivElement
  ▶ 1: HTMLDivElement
  ▶ context: HTMLDocument
    length: 2
  ▶ prevObject: Object
    selector: ".main"
```

By Tag

```
<head>
    <script src="../jquery.js" type="text/javascript"
charset="utf-8"></script>
    <script>
    function onloadActions(){
        var a = $("div");
        console.log(a);
    }
    $(document).ready(onloadActions)
    </script>
</head>
<body>
    <div id="selectme"></div>
</body>
```

```
▼ Object
  ▶ 0: HTMLDivElement
  ▶ context: HTMLDocument
    length: 1
  ▶ prevObject: Object
    selector: "div"
```

Acting on selector results

```
function actOnElement(index, element) {
    console.log(index, element);
}

$(".main").each( actOnElement );
```

```
0 <div class="main" id="selectme">This is the first div.</div>     selecting.html:7
1 <div class="main" id="selectmetoo">This is the second div.</div> selecting.html:7
```

One gotcha

- Sometimes jQuery returns a "jQuery wrapped" object that responds to jQuery commands like **.each**, **.click**, etc
- Other times it's just the raw DOM element
- Use $( ) to convert back to jQ wrapped

Examples

```
var a = $("#selectme"); // a is jQ object array
>>> Object
    0: HTMLDivElement
    context: HTMLDocument
    length: 1
    selector: "#selectme"

a[0];
>>> <div class="main" id="selectme">This is the first div.</div>

$(a[0]);
>>> Object
    0: HTMLDivElement
    context: HTMLDivElement
    length: 1
```

Specifying context

```
<div id="firstcontainer">
    <div class="main" id="selectme">This is the first div.</div>
    <div class="main" id="selectmetoo">This is the second div.</div>
</div>
<div id="secondcontainer">
    <div class="main" id="selectmethree">This is the third div.</div>
    <div class="main" id="selectmefour">This is the fourth div.</div>
</div>

// if we want to select just the .main divs in second container:
$(".main", "#secondcontainer");
>>> Object
    0: HTMLDivElement
    1: HTMLDivElement
    context: HTMLDocument
    length: 2
    prevObject: Object
    selector: "#secondcontainer .main"
```

Traversing the DOM

- Use **parent()**, **children()**, **next(), prevObject**

Traversing the DOM

```
<script>
function onloadActions(){
    var a =
$("#container");
    console.log(a.children());
    console.log(a.children()[0]);
    console.log($(a.children()[0]).parent());
}
window.onload = onloadActions;
</script>
</head>
<body>
    <div id="container">
        <div id="selectme"></div>
        <div id="selectmetoo"></div>
    </div>
```

```
▶ Object
<div id="selectme">
▶ Object
▶ Object
```

Chaining

```
/* $() usually returns the object acted on,
   which lets you do things like:
*/
$("#mydiv").css("background-color",
"red").click(function(){ alert('hi')}).show()
```

Creating & Adding Elements

Task

- Trying to insert objects into the DOM dynamically
- For example, a "Loading..." indicator

$("htmlstring")

```
var el = $("<div></div>");
```

Accepts: a **string** representing the HTML to create

Returns: the created Element (which **hasn't** been added to the DOM)

$.append

```
var a = $("<div>Loading</div>");
var container = $("#container");

container.append(a);
```

append is called **on** an element, **accepts** an element, returns the original element

you can also do:

```
a.appendTo("#container"); // or,
a.appendTo($("#container"));
```

appendTo returns the element being appended

Setting the content

```
/* use .html() */

var el = $("<div></div>");
el.html("Loading...");

$(document).append(el);
```

Changing styles

Styling from JS

- All jQuery elements have a .css() function
- Either call it with .css("property", "value"), or pass in an object like so:

```
.css({
        'prop1': 'value',
        'prop2': 'value'
    })
```

Examples

```
Black background: $("#selectme").css("background-
color", "black")

12px font -> $("#selectme").css("font-size", "12px");

5px rounded corners -> $("#selectme").css("-webkit-
border-radius", "5px")

//All together:
$("#selectme").css({
    "background-color": "black",
    "font-size": "12px",
    "-webkit-border-radius": "5px"
});
```

Showing / hiding

```
var el = $("#loading");

el.hide();
el.show();

el.hide(true); //with animation
el.show(true); //with animation
```

Example

```
<script src="../jquery.js" type="text/javascript" charset="utf-8"></script>
<script>
function onloadActions(){
    var a = $(".main");
    a.each(function(i, el) {
        $(el).css({
            'width': '200px',
            '-webkit-border-radius': '10px',
            'border': "1px solid #333"
        })
    });
    $(".main").click(function(){
        $(this).hide(true);
    })
}
function showAll() {
    $(".main").show(true);
}
$(document).ready(onloadActions);
</script>
<div id="firstcontainer">
    <div class="main" id="selectme">This is the first div.</div>
    <div class="main" id="selectmetoo">This is the second div.</div>
</div>
<input type="button" onclick='showAll()' value="show all"/>
```

Demo

`showhide.html`

Changing classes

```
$("#id").addClass("classname");
$("#id").removeClass("classname");
$("#id").toggleClass("classname");
```

Example

```
<script src="../jquery.js" type="text/javascript"
charset="utf-8"></script>
<style type="text/css" media="screen">
    .highlight { background-color: yellow;}
</style>
<script type="text/javascript" charset="utf-8">
    $(document).ready(function(){
        $("#clickme").click(function(){
            $(this).addClass("highlight");
        })
    })
</script>
<script type="text/javascript" charset="utf-8">
    <div id="clickme">Click to highlight</div>
</script>
```

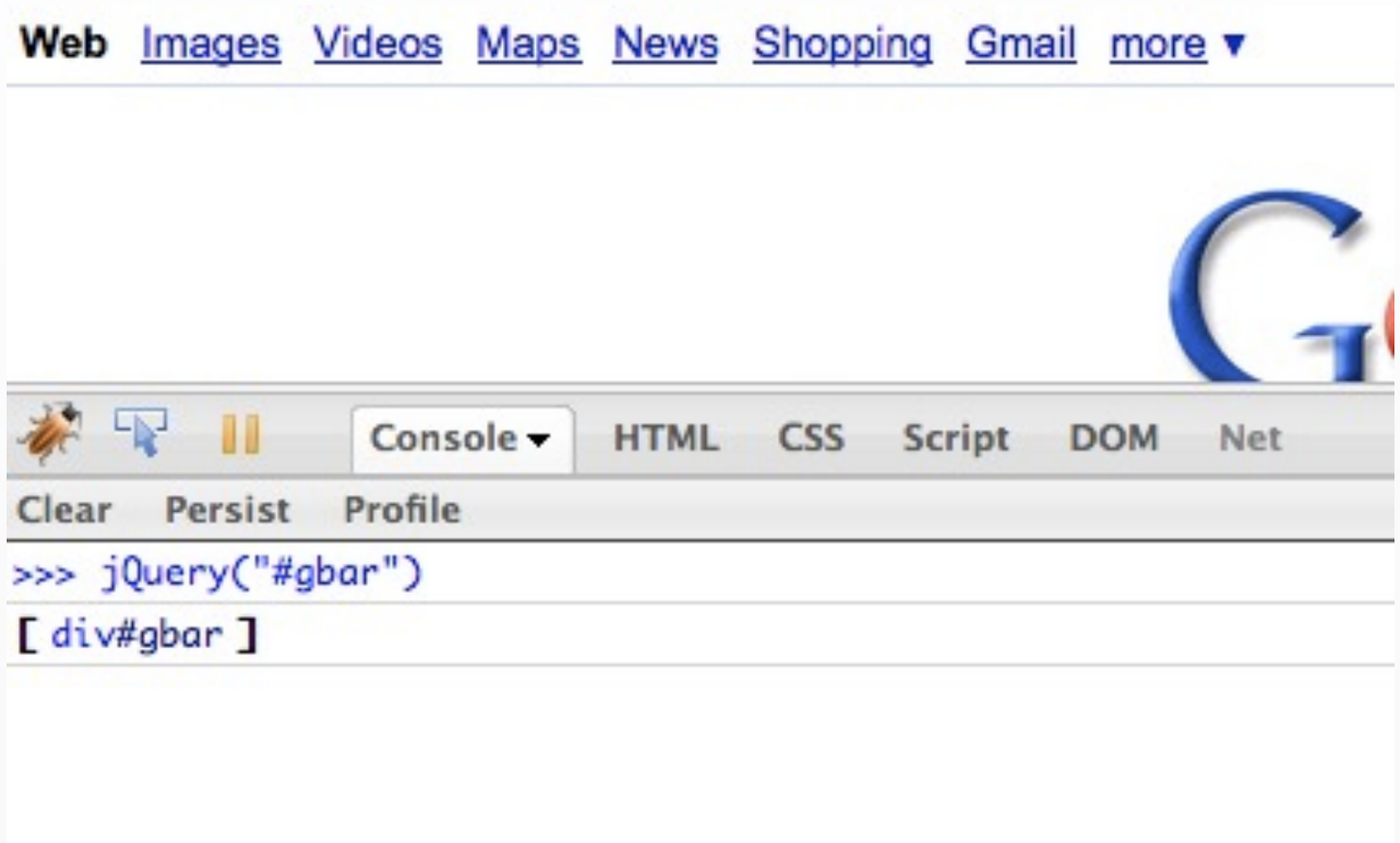
Demo

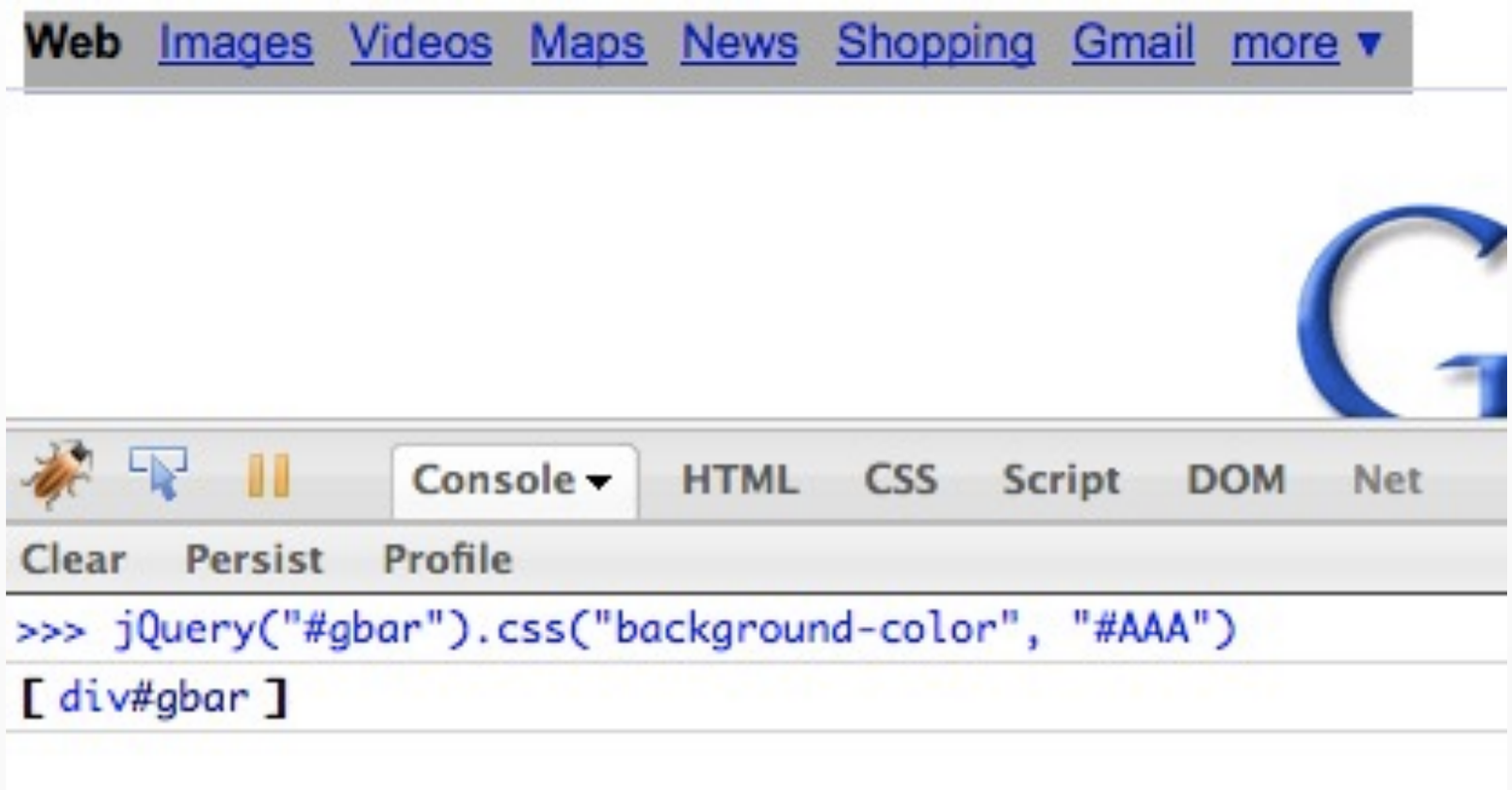
`classnames.html`

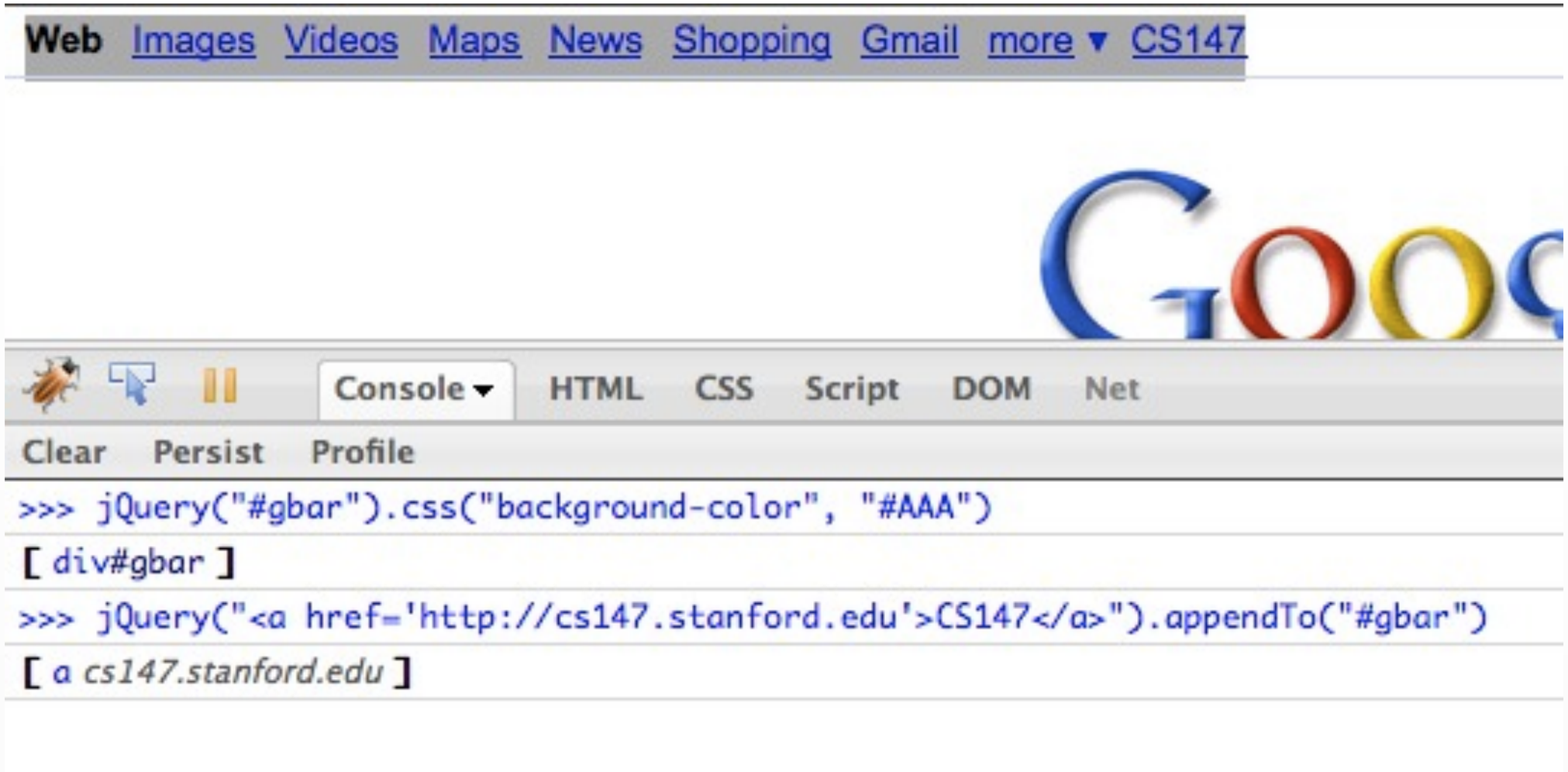
Firebug is your friend

Interactive console

- Use it whenever you want to test something out

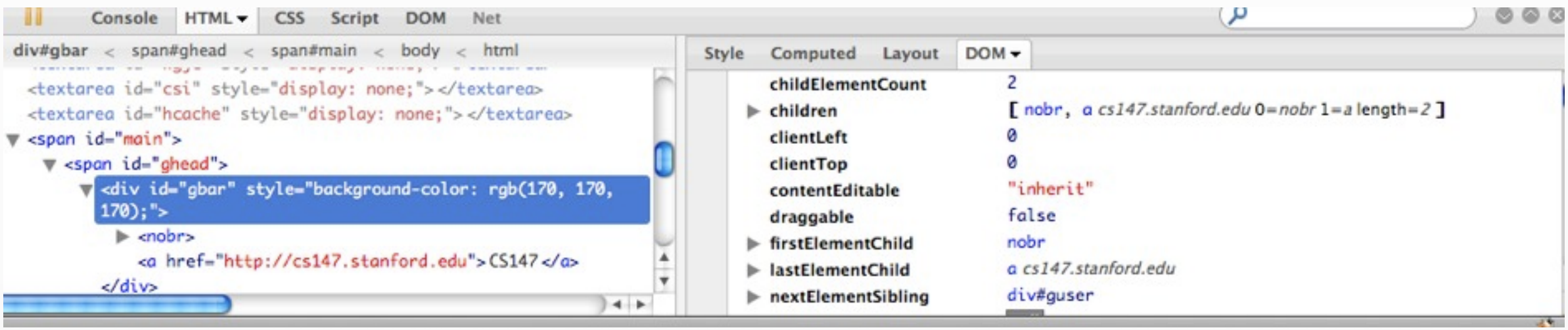

Web Images Videos Maps News Shopping Gmail more ▾
Console HTML CSS Script DOM Net
Clear Persist Profile
>>> jQuery("#gbar")
[div#gbar]


Web Images Videos Maps News Shopping Gmail more ▼
Console HTML CSS Script DOM Net
Clear Persist Profile
>>> jQuery("#gbar").css("background-color", "#AAA")
[div#gbar]



Create element, and appendTo the top bar

Clicking on DOM element inspects it



Event-driven architecture

Event-driven vs polling

- Two different approaches to UI programming: polling & event-driven

Polling

- Scripts that are interested in changes have to go: "did anything change? did anything change? did anything change?" every *n* seconds

Event-driven

- Interested listeners **register** themselves
- When an event occurs, source **notifies** its listeners

In Javascript

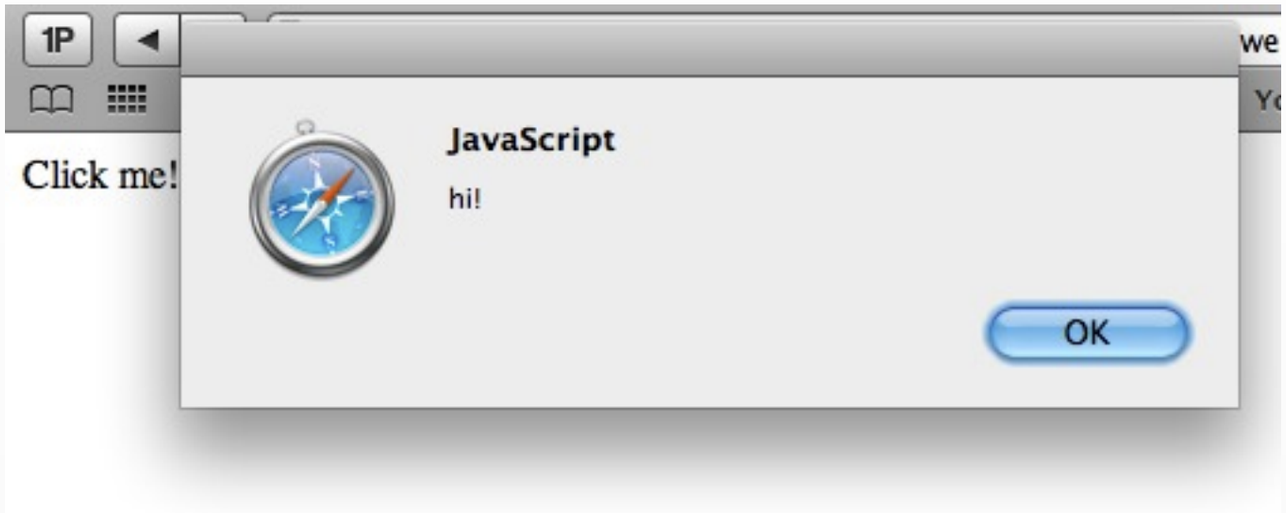
- **addEventListener**('eventName', function, ...);
  - in jQuery, we do **.bind('eventname', callbackFn)** or the actual event, so **.click(callbackFn)**, **.hover**, etc

Example: hooking up to listen to a click

```
<script src="../jquery.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript" charset="utf-8">
    function doSomething(event) {
        alert("hi!");
    }

    function init() {
        var el = $('#clickme');
        el.click(doSomething);
    }
    $(document).ready(init);
</script>
<body>
    <div id="clickme">Click me!</div>
</body>
```

Does...



Hover

- element.hover(onMouseOverCallback, onMouseOutCallback);

  (first function called on entering the element, other called on leaving)

Hover

```
<script src="../jquery.js" type="text/javascript" charset="utf-8"></
script>
<script type="text/javascript" charset="utf-8">
    function doOnMouseOver(event) {
        $(event.target).css("background-color", "blue");
    }
    function doOnMouseOut(event) {
        $(event.target).css("background-color", "white")
    }

    function init() {
        var el = $('#hoverme');
        console.log(el);
        el.hover(doOnMouseOver, doOnMouseOut);
        el.css("-webkit-transition", "background-color 1s ease");
    }
    $(document).ready(init);
</script>
<body>
```

Demo

hover.html

Events worth watching for

```
click
hover
load
mousemove (useful for drag & drop)
```

Inline Functions

Overview

- Functions can be defined **anonymously** in line
- This is most helpful for event handlers

Example

```
function init() {
   $("#clickme").click(function(event){
      // do something on click
   })
}

// we've defined an anonymous function
// that will execute on click
```

Closures

- Functions defined anonymously inside other functions will have that their parent function's context

Example

```
function init() {
   var saying = "Hello";
   $("#clickme").click(function(){
      alert(saying);
   })
}

// Even though that function executes way
// after init() is done running, it can
// access init's variables
```

Closures gone wrong

```
function init() {
   for(var i = 0; i < 3; i++){
      $("<div>" + i + "</div>").appendTo("#container");
   }
}
```

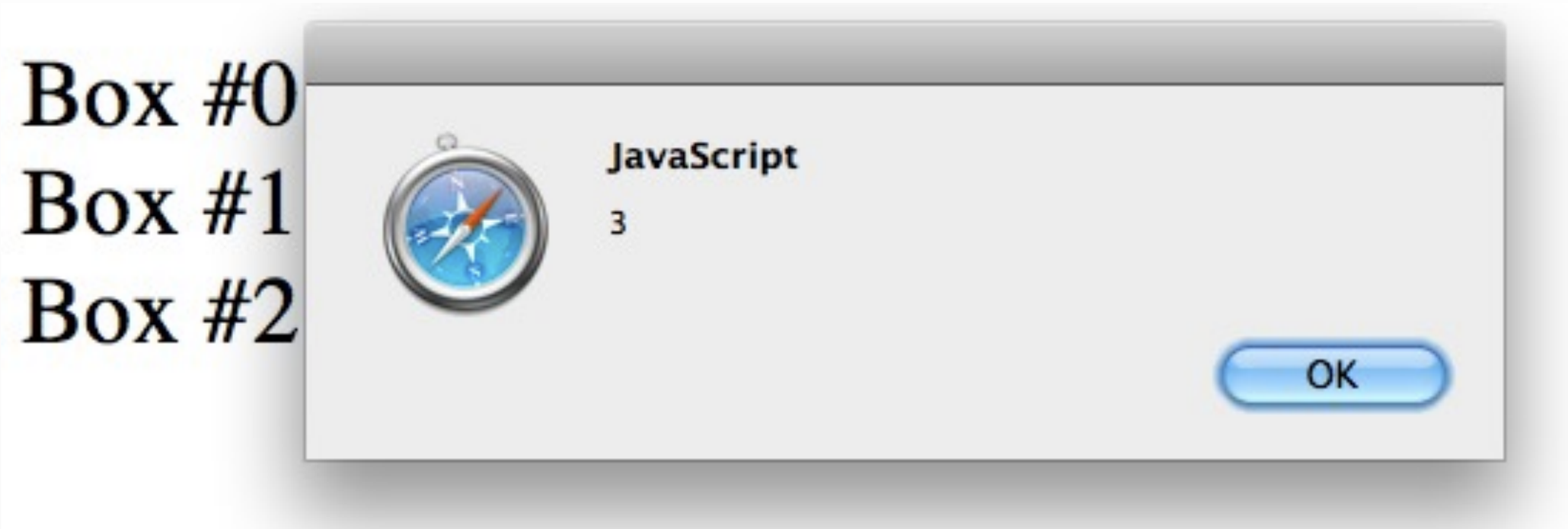
Box #0
Box #1
Box #2

Closures gone wrong

```
for(var i = 0; i < 3; i++){
   var newDiv = $("<div>Box #" + i + "</div>");
   newDiv.appendTo("#container");
   newDiv.click(function(){
      alert(i);
   })
}
```

Box #0
Box #1
Box #2

What happens when I click on Box #0?


Box #0
Box #1
Box #2
JavaScript
3
OK

Uh oh...

```
function init() {
   for(var i = 0; i < 3; i++){
      var newDiv = $("<div>Box #" + i + "</div>");
      newDiv.appendTo("#container");
      newDiv.click(function(){
         alert(i);
      })
   }
}
```

Our click closure points back to init(), but in init (), the i variable equals 3 because the for loop kept going after the event handler was attached to box #0

Workarounds

- There are ways to do this, but they're complicated
- We can cover in last week if interested
- For now, don't rely on values you expect to change in original function, use **event** or **this** instead (example next)

You are Goldilocks. What would you like to do next?

- Try the first bed.
- Try the middle bed.
- Try the last bed.

The code

```
<script src="../jquery.js" type="text/javascript"
charset="utf-8"></script>
<script type="text/javascript" charset="utf-8">

</script>
<body>
You are Goldilocks. What would you like to do next?
<div class="actionlink" id="firstbed">Try the first bed.</div>
<div class="actionlink" id="secondbed">Try the middle bed.</div>
<div class="actionlink" id="thirdbed">Try the last bed.</div>
```

Attaching listeners

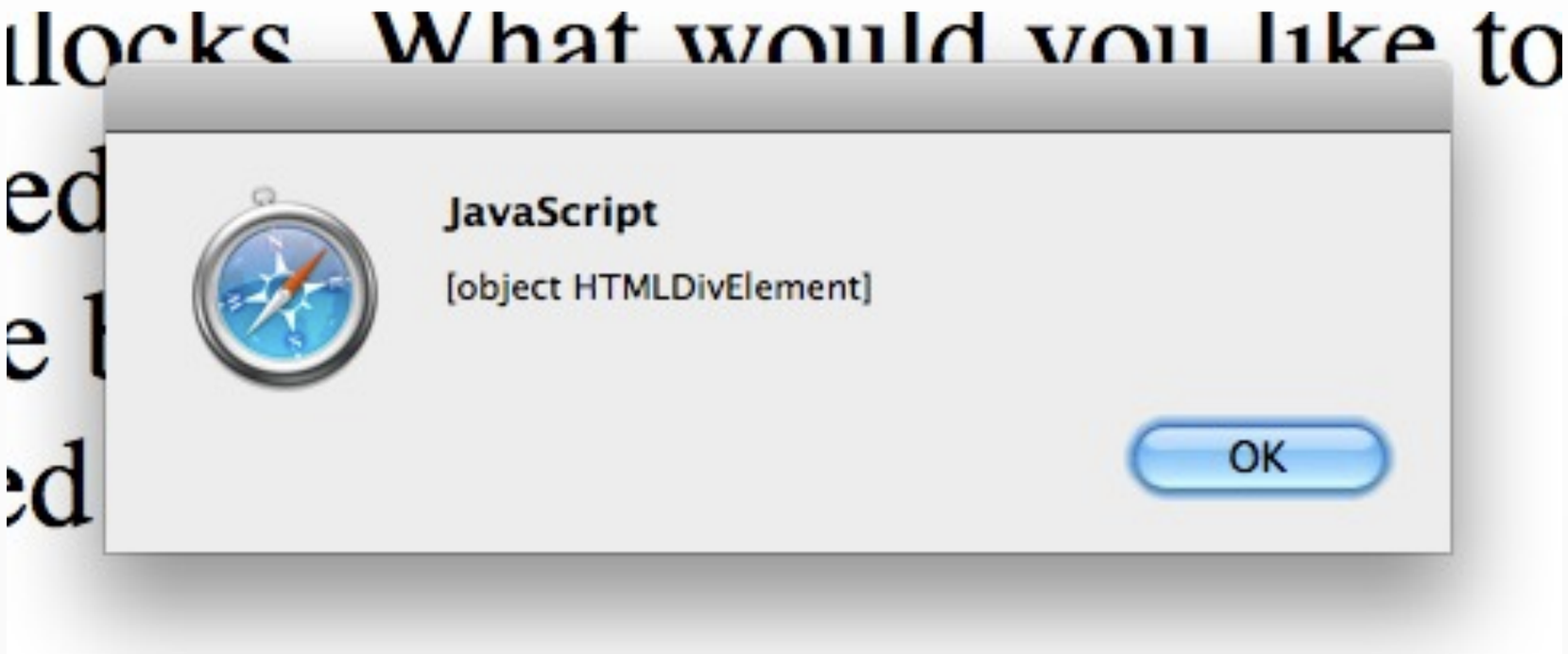
```
$(document).ready(function(){
    var message = "this bed is ";
    $(".actionlink").click(function(event){
        var whichBed = event.target.id;
        var result;
        if(whichBed == 'firstbed') {
            result = "too small!";
        } else if (whichBed == 'secondbed') {
            result = "too big!";
        } else {
            result = "just right!";
        }
        alert(message + result);
    });

})
```

Demo

What context is that callback being executed in?

```
$(".actionlink").click(function(event){
   alert(this);
});
```

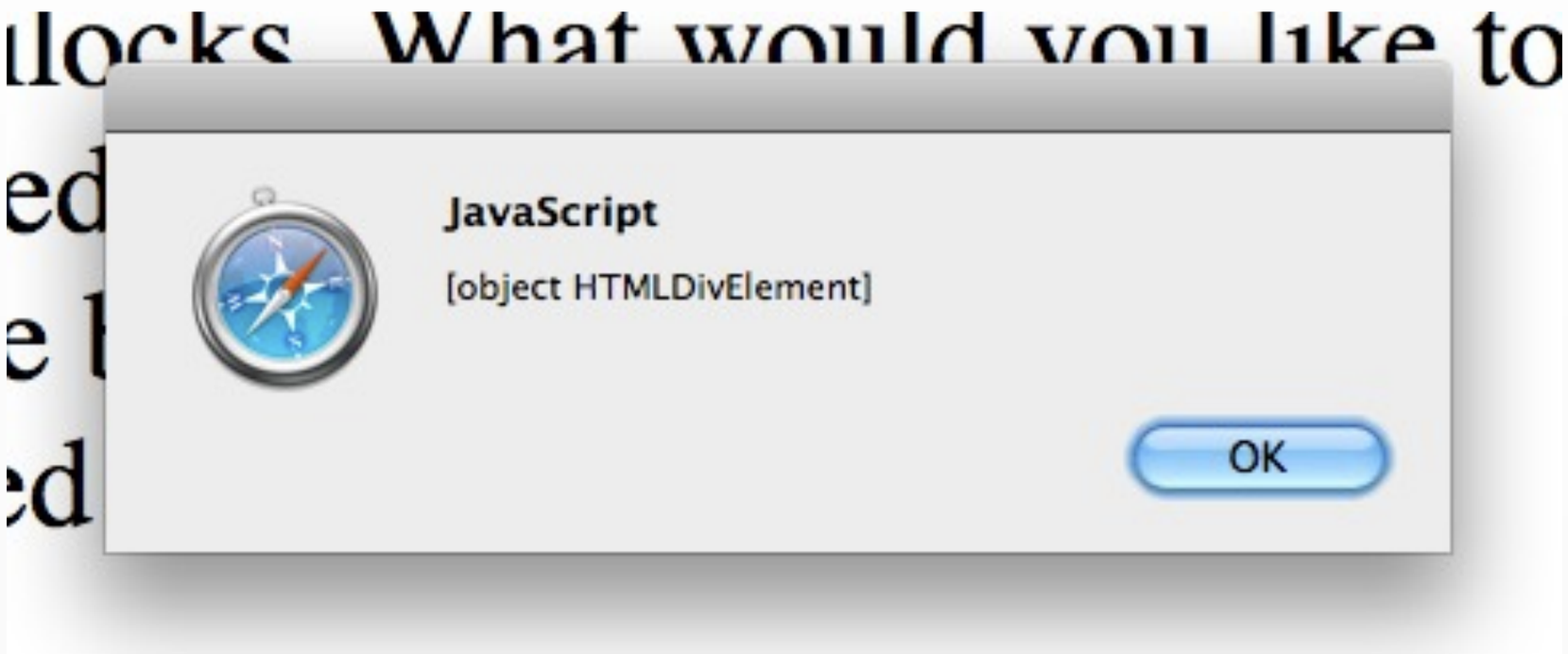

JavaScript
[object HTMLDivElement]
OK

What's going on?

- Event function is *attached* to the div
- When div fires event, function fires with the div as context
- It's a closure, so still has access to scope it was created in (but **this** has changed)

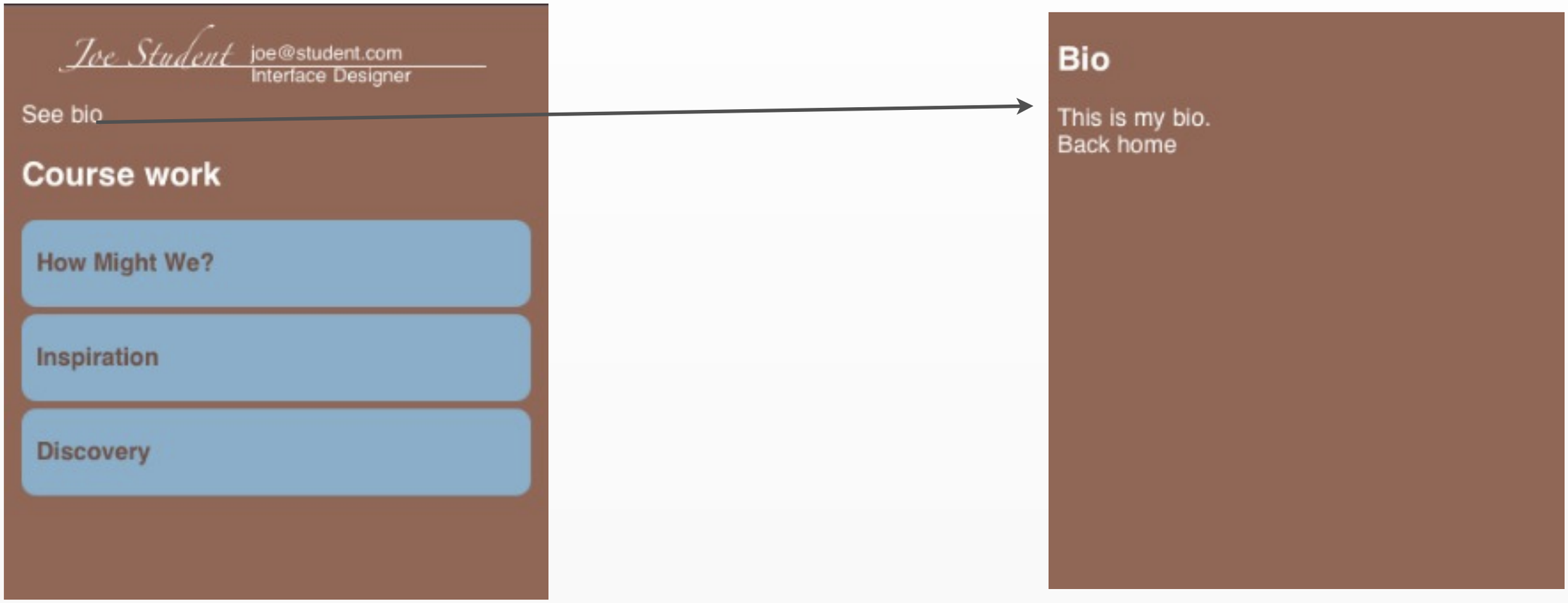
So if we wanted to use 'this'

```
$(".actionlink").click(function(event){
    // this callback is attached to each div,
    // and 'this' is the clicked div
    var whichBed = this.id;
    var result;
    if(whichBed == 'firstbed') {
        result = "too small!";
    } else if (whichBed == 'secondBed') {
        result = "too big!";
    } else {
        result = "just right!";
    }
    alert(result);
});
```


JavaScript
[object HTMLDivElement]
OK

Portfolio 4

Refresher


Joe Student
joe@student.com
Interface Designer
See bio
Course work
How Might We?
Inspiration
Discovery
Bio
This is my bio.
Back home

Goal

- Hook up the subpages, too

Strategy

- Minimize page loads on the iPhone
- All links lead to current page but with an anchor in the hash ("page.html#id")
- Use setInterval to watch for changes in the hash and update page

Step 1: adding content

```
<div id="how-might-we" class='content subpage'>
    This is some great work I did for the How Might We?
Assignment.
</div>
<div id="inspiration" class='content subpage'>
    Wow, that was super inspirational.
</div>
<div id="discovery" class='content subpage'>
    Can you discover?
</div>
```

Step 2: Hooking up links

```
<li><a href="#how-might-we">How Might We?</a></li>
<li><a href="#inspiration">Inspiration</a></li>
<li><a href="#discovery">Discovery</a></li>
```

Step 3: watching for hash changes

```
var loop = setInterval(function(){
   var curid = currentPage.attr('id');
   if (location.hash == '') {
      location.hash = '#' + curid;
   } else if (location.hash != '#' + curid) {
      goPage(location.hash)
   }
}, 100);

// jQTouch will take care of this next week
```

Step 4: Changing Pages

```
function goPage () {
    var pageToLoad = window.location.hash;
    var prevFound = false;
    for(var i = 0; i < pageHistory.length; i++) {
        if (pageHistory[i] == pageToLoad) {
            $(pageToLoad).removeClass("parentpage");
            $(currentPage).addClass("subpage");
            prevFound = true;
            pageHistory.pop();
        }
    }
    if(!prevFound) {
        $(currentPage).addClass("parentpage");
        $(pageToLoad).removeClass("subpage");
        pageHistory.push("#"+currentPage.attr("id"));
    }
    currentPage = $(pageToLoad);
    return false;
}
```

Step 5: CSS Classes

```
.subpage {
   left: 360px !important;
}
.parentpage {
   left: -360px !important;
}
```

Demo

week04.html in portfolio folder

bug when on iPhone, will fix and update