

Human-Computer Interaction Design

The background of the slide is a dark blue gradient. A lighter blue world map is centered behind the title. In the top right corner, there is a cluster of five lightbulb icons, some of which are inside a wireframe cube. Another single lightbulb icon is located over the South American continent of the world map.

COGS120/CSE170 - "Intro. HCI"

Instructor: Philip Guo

Lab 3 - Interacting with webpage elements (2016-10-13)

by Michael Bernstein, Scott Klemmer, and Philip Guo

Format for labs:

1. I'll give a mini-lecture on the slides for 20-30 minutes. You can follow along or jump ahead at your own pace. [room will be mostly silent so that everyone can hear clearly]

2. Then we will open it up for free-form lab work with TAs walking around to help. You can also come up to the podium to ask me for help too. [room will be louder]

When the web began (early 1990s), interaction meant clicking on **blue links**.

To become interactive, web pages rendered changes by loading an entirely new page.

This is pretty klunky.

Today, modern web
browsers and JavaScript
enable far richer interaction.

Two main ways to write JavaScript ...

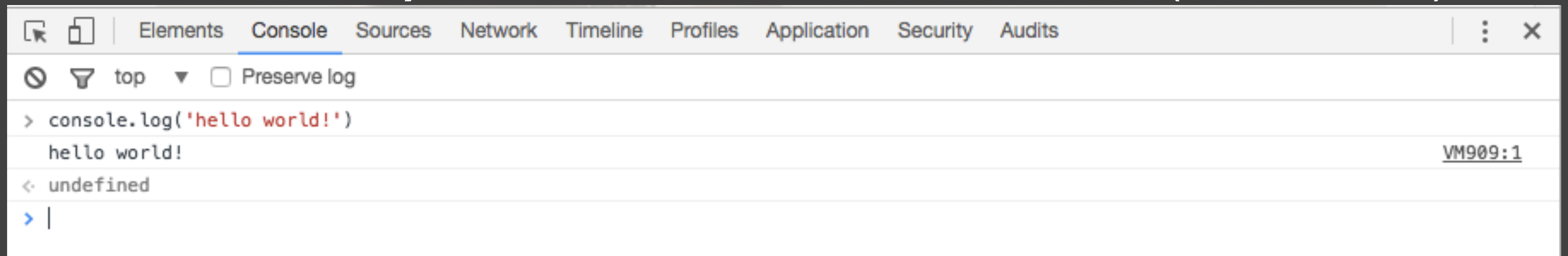
- Write it in a file and add a reference to it in the HTML file.

e.g.,: `<script src="js/introHCI.js"></script>`

```
<div class="project" id="project">
  <a href="#" class="thumbnail">
    
    <p>Design tools</p>
  </a>
</div>
</div>

<script src="https://code.jquery.com/jquery.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

- Write it directly in the web browser console (not saved!)



I) Reacting to user input

Javascript event handling
Debugging in Javascript

2) Updating the page

Add content: adding new elements

Change how things look: CSS updates

Effects

concepts

Reacting to user input

To react to user input, use the jQuery library to listen for events on elements

- Ask to be notified of clicks on element `'#myelement'` using its `click` method

```
$(' #myelement' ).click(functionName)
```



Select the HTML element
with id “myelement”



Call this function when
the user clicks the element

Example

```
<p id="myelement">Example text</p>
```

```
$( '#myelement' ).click( changeText );  
function changeText( event ) {  
    $( this ).text( "Changed text" );  
}
```

Example text

Other events you might care about

- `$( '#myelement' ).click(functionName)`

when the user clicks

- `$( '#myelement' ).keypress(functionName)`

when the user types

- `$( '#myelement' ).mouseenter(functionName)`

when the mouse enters the element

- `$( '#myelement' ).mousemove(functionName)`

every time the mouse moves within the element

- `$( '#myelement' ).submit(functionName)`

when a form is submitted

- <http://learn.jquery.com/events/event-basics/>

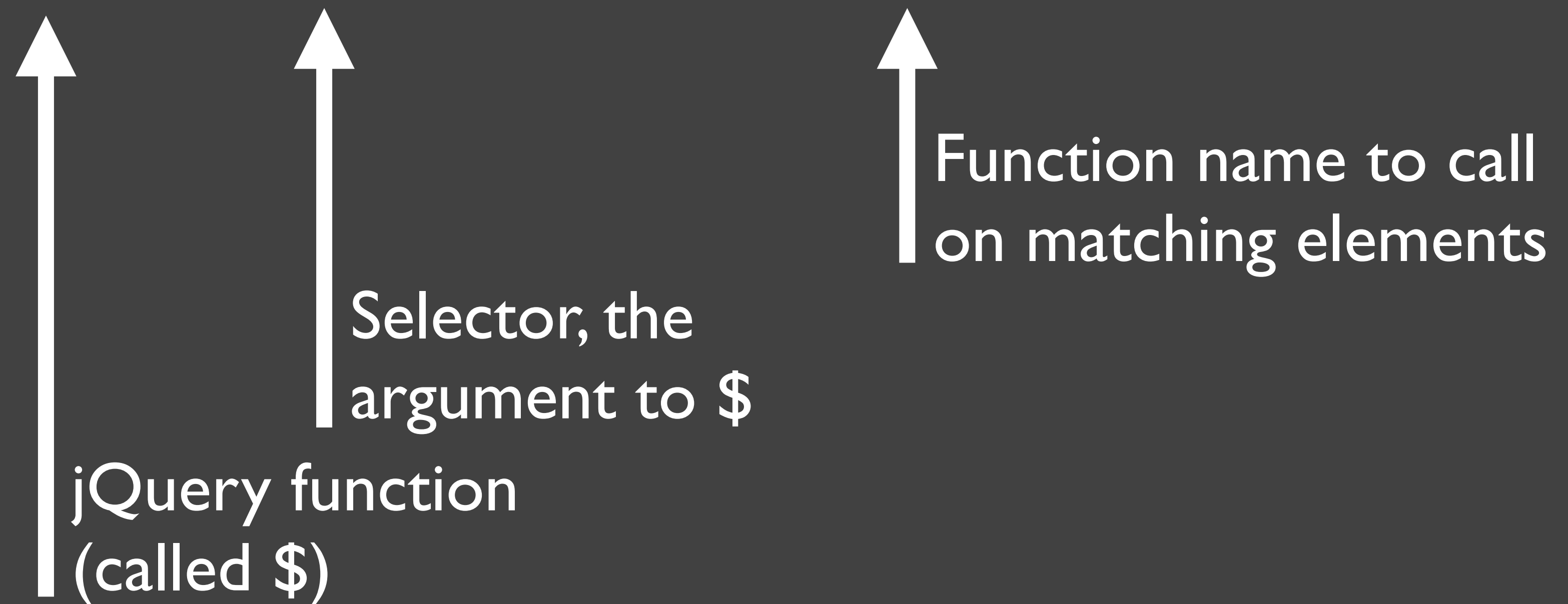
concepts

Selectors



- `$()` is the shortcut for `jQuery()`
- jQuery is a popular library that allows you to **select** elements of the web page, then ask for info about or change anything that matches your selection.

```
$(".project").click(listenerFunction);
```





- `$()` is the shortcut for `jQuery()`
- jQuery is a popular library that allows you to **select** elements of the web page, then ask for info about or change anything that matches your selection.

```
$(".project").click(listenerFunction);  
$(".project").append("<h1>jQuery</h1>");  
$(".project").hide();
```

Selectors: dot for CSS class name

All elements of class “project”:

`$(".project")`

`<p>Projects are below.</p>`

`<div class="project" id="project1">`

`<p>Waiting in line</p>`

`</div>`

`<div class="project" id="project2">`

`<p>Needfinding</p>`

`</div>`

Selectors: no dot means HTML element type

All `<p>` elements:

```
$(“p”)
```

```
<p>Projects are below.</p>
```

```
<div class=“project” id=“project1”>
```

```
    <p>Waiting in line</p>
```

```
</div>
```

```
<div class=“project” id=“project2”>
```

```
    <p>Needfinding</p>
```

```
</div>
```

Selectors: spaces indicate hierarchy

All `<p>` elements inside an element of class “project”:
`$(".project p")`

```
<p>Projects are below.</p>
```

```
<div class="project" id="project1">
```

```
    <p>Waiting in line</p>
```

```
</div>
```

```
<div class="project" id="project2">
```

```
    <p>Needfinding</p>
```

```
</div>
```

Selectors: no spaces for logical “and”

All `<p>` elements with class “project”:

```
$(“p.project”)
```

```
<p>Projects are below.</p>
```

```
<div class=“project” id=“project1”>
```

```
    <p>Waiting in line</p>
```

```
</div>
```

```
<div class=“project” id=“project2”>
```

```
    <p>Needfinding</p>
```

```
</div>
```

Selectors: # for CSS element id

The element with id “project2”:

`$("#project2")`

```
<p>Projects are below.</p>
```

```
<div class="project" id="project1">
```

```
    <p>Waiting in line</p>
```

```
</div>
```

```
<div class="project" id="project2">
```

```
    <p>Needfinding</p>
```

```
</div>
```

To see how many items matched
your selector...

`$(selector).length`

For more selectors...

- See the jQuery documentation:
<http://learn.jquery.com/using-jquery-core/selecting-elements/>

concepts

Manipulating content

Adding new content with Javascript: why?

- Add new emails to your Gmail inbox without reloading
- Enable infinite scrolling on Facebook and Twitter
- Insert more information when the user clicks on 'help'
- Show warnings if the user's input isn't properly formatted
- ALL OF THIS HAPPENS WITHOUT THE WEB PAGE RELOADING

Append to an element

- `$(“some_selector”).append(some_html)`
adds *some_html* to the end of the element

Example:

```
var description = $(".description");  
description.append("<p>New text</p>");
```

Replace instead of append

- `$(“some_selector”).append(some_html):` adds content
- `$(“some_selector”).html(some_html):` replaces content
- `$(“some_selector”).text(some_text):` replaces text
- `$(“some_selector”).remove():` gets rid of the element

Example:

```
var description = $(".description");  
description.text("Changed text");
```

Change CSS using Javascript: why?



Prerequisites

Bring a laptop to class, lab, and studio

Lab

Studio

Assignments and Grading

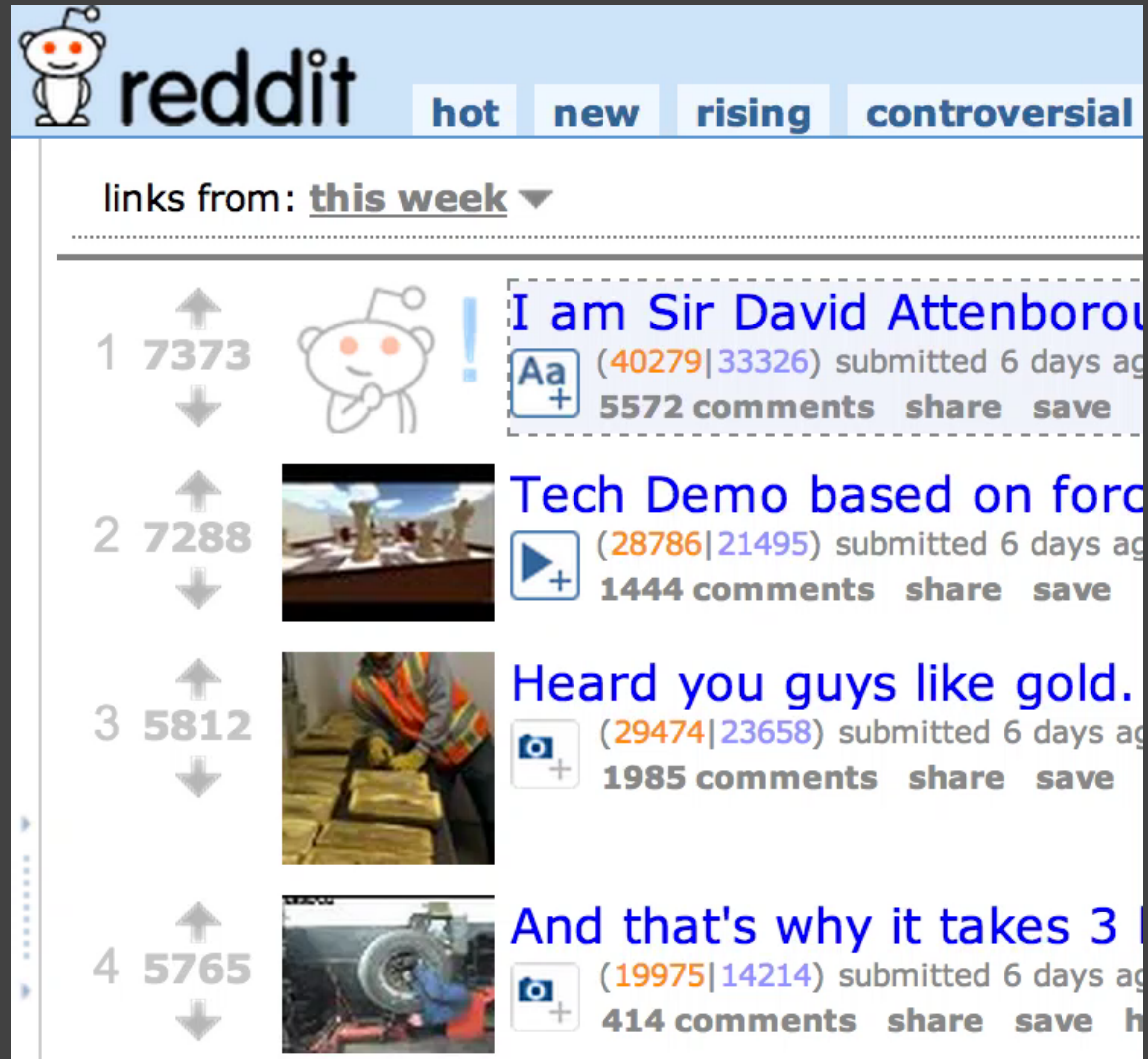
Deliverables

Grade breakdown

Attendance

CS77

Frequently asked questions



Adding a CSS class

- Select the element, then call `addClass()`

Example:

```
var description = $(".description");  
description.addClass("active");
```

```
.description.active {  
    border: 1px solid #ee3223;  
}
```

Remove, and toggle CSS classes

- You can remove classes with `removeClass()`
- Or, toggle them on and off with `toggleClass()`:

```
$(".description").click(function(e) {  
    $(this).toggleClass("active");  
});
```

concepts

Effects

Now you see it...

- Immediate

- `$("#myelement").hide()`: hide an element on the page
- `$("#myelement").show()`: bring it back

- Animated

- `$("#myelement").fadeIn()`
- `$("#myelement").fadeOut()`

Animate any CSS property

- `(“#myelement”).animate({
 css-property: value
}, durationInMilliseconds)`

- For example:

```
<img id=“myImg” src=“...” />
```

```
(“#myImg”).animate( {  
    width: 500  
}, 1000);
```

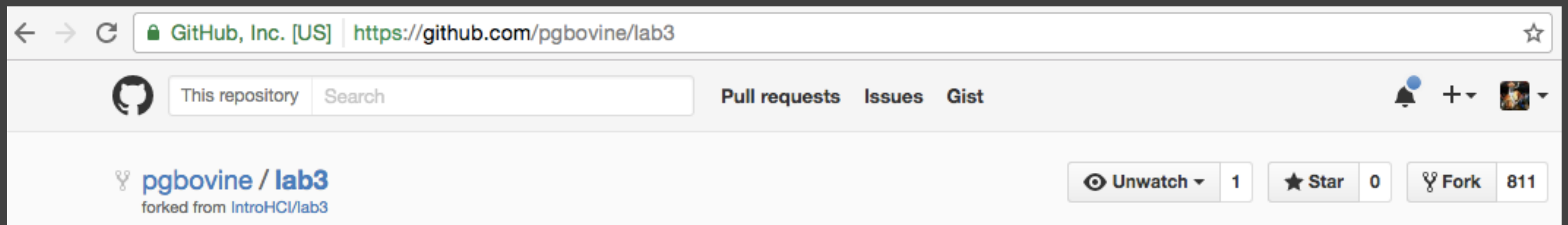


practice - OK now it's your turn to try ...

Reacting to the user

Fork and clone (again!)

- Fork lab 3 at: <https://github.com/pgbovine/lab3>
 - Make sure you fork first, don't just clone!
- Open your command line to your introHCI directory
- `git clone` your forked repository into introHCI
 - (Clone the git repo from the terminal with the https URL; don't download the zip file)



Link js/introHCI.js to our HTML file

- Open lab3/static/index.html in Sublime Text
- Add our Javascript reference near the bottom, below the Bootstrap includes: `<script src="js/introHCI.js"></script>`

```
<div class="project" id="project">
  <a href="#" class="thumbnail">
    
    <p>Design tools</p>
  </a>
</div>
</div>

<script src="https://code.jquery.com/jquery.js"></script>
<script src="js/bootstrap.min.js"></script>
</body>
</html>
```

Open lab3/static/index.html in your browser

- Whenever you reload this page, it will get the latest saved version of your javascript file

Is the Javascript connected correctly?

- Click on the “Test Javascript” button



Michael Bernstein

human-computer interaction · social computing · crowdsourcing

Test Javascript

Listen to clicks on project `<a>` elements

- Open `static/js/introHCl.js` in Sublime Text

- Inside of `initializePage`, register the click listener

```
function initializePage() {  
    $("#testjs").click(function(e) {  
        $('.jumbotron h1').text("Javascript is connected");  
    });  
  
    // Add any additional listeners here  
    // example: $("#div-id").click(functionToCall);  
    $("a.thumbnail").click(projectClick);  
}
```

Define the listener function

- Nothing will happen unless we define `projectClick`
- It should prevent the default behavior (following the link), then change the color of the project node:

```
function projectClick(e) {  
    // prevent the page from reloading  
    e.preventDefault();  
  
    // In an event handler, $(this) refers to  
    // the object that triggered the event  
    $(this).css("background-color", "#7fff00");  
}
```

Did it work?



Waiting in Line

practice

Debugging

The Kōan of Web Debugging

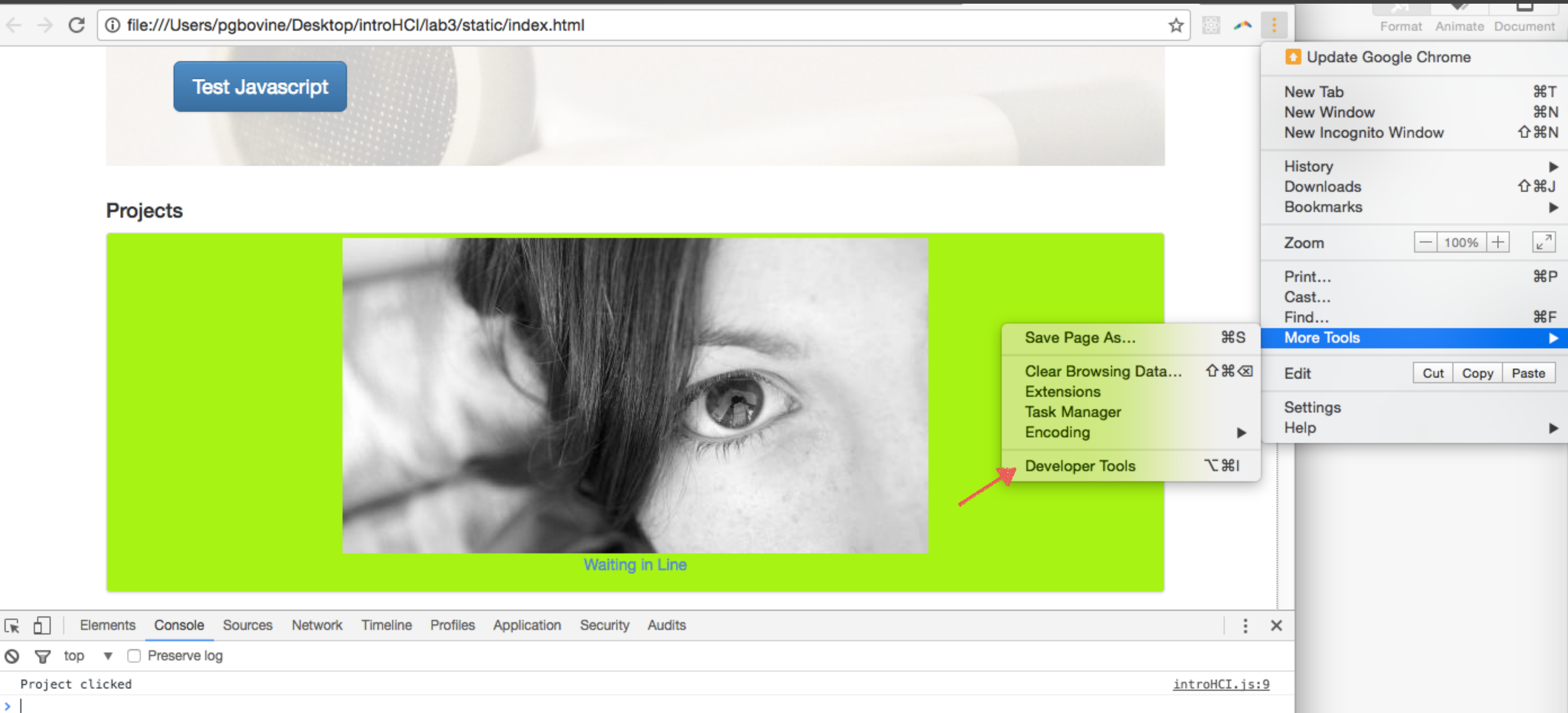
- To be an engineer, you must be a scientist.
- Start with the simplest test case you can find.
- Run experiments: find two conditions that vary in only one way, and which have two different outcomes.

Add a debugging statement

- `console.log(some_string)` will print *some_string* to the browser's console

At the top of `projectClick`:
`console.log("Project clicked");`

```
function projectClick(e) {  
    console.log("Project clicked");  
    e.preventDefault();  
    $(this).css("background-color", "#7fff00");  
}
```



file:///Users/pgbovine/Desktop/introHCI/lab3/static/index.html

Test Javascript

Projects



Waiting in Line

ElementsConsoleSourcesNetworkTimelineProfilesApplicationSecurityAudits

top

Preserve log

Project clicked

introHCI.js:9

Find the bug, using `console.log()`

This code should change the title of the jumbotron header when you click on a project thumbnail. Why doesn't it?

Replace `projectClick` with code from the lab web page:

```
function projectClick(e) {  
    // Cancel the default action, which prevents the page from reloading  
    e.preventDefault();  
  
    // In an event listener, $(this) is the element that fired the event  
    var projectTitle = $(this).find("p").text();  
    var jumbotronHeader = $("#jumbotron h1");  
    jumbotronHeader.text(projectTitle);  
}
```

Warning: solution ahead ...
try it yourself first!



index.html



introHCl.js



untitled



untitled



```
1  'use strict';
2
3  // Call this function when the page loads (the "ready" event)
4  $(document).ready(function() {
5      initializePage();
6  })
7
8  /*
9   * Function that is called when the document is ready.
10  */
11  function initializePage() {
12      $("#testjs").click(function(e) {
13          $('#jumbotron h1').text("Javascript is connected");
14      });
15
16      // Add any additional listeners here
17      // example: $("#div-id").click(functionToCall);
18      $("a.thumbnail").click(projectClick);
19  }
20
21  function projectClick(e) {
22      // Cancel the default action, which prevents the page from reloading
23      e.preventDefault();
24      // In an event listener, $(this) is the element that fired the event
25      var projectTitle = $(this).find("p").text();
26      var jumbotronHeader = $("#jumbotron h1");
27      jumbotronHeader.text(projectTitle);
28  }
```

```
1  $(document).ready(function() {
2      // Call this function when the page loads (the "ready" event)
3      initializePage();
4  })
5
6  /*
7   * Function that is called when the document is ready.
8   */
9  function initializePage() {
10     $("#testjs").click(function(e) {
11         $('#jumbotron h1').text("Javascript is connected");
12     });
13
14     // Add any additional listeners here
15     // example: $("#div-id").click(functionToCall);
16     $("a.thumbnail").click(projectClick);
17 }
18
19 function projectClick(e) {
20     // Cancel the default action, which prevents the page from reloading
21     e.preventDefault();
22     // In an event listener, $(this) is the element that fired the event
23     var projectTitle = $(this).find("p").text();
24     var jumbotronHeader = $("#jumbotron h1");
25     jumbotronHeader.text(projectTitle);
26 }
```

Check your selector syntax

`#jumbotron` selects the element with `id="jumbotron"`.

`.jumbotron` selects the element with `class="jumbotron"`.

```
var jumbotronHeader = $(".jumbotron h1");
```

practice

Manipulating content

Append to an element

Remember: `$(“some_selector”).append(some_html)`

Add this code to the end of your `projectClick` function:

```
var containingProject = $(this).closest(".project");  
containingProject.append("<div class='project-description'><p>Description  
of the project.</p></div>");
```

Well, it almost works.

- It appends the same thing each time you click on the element.
- You get points for enthusiasm, I suppose.



Waiting in Line



Replace instead of append

- Other options: `$.html(some_html)`, `$.text(some_text)`

Try combining them to dynamically update the description.

Replace the previous version with:

```
var containingProject = $(this).closest(".project");
var description = $(containingProject).find(".project-description");
if (description.length == 0) {
    $(containingProject).append("<div class='project-  
description'><p>Description of the project.</p></div>");
} else {
    description.html("<p>Stop clicking on me! You just did it at " + (new  
Date()) + "</p>");
}
```

Your turn ... try to make this ...

- When the user clicks on the button in the jumbotron, change the text of that button.
 - The button's CSS id is `testjs`
- Make changes in the function that contains:

```
$(".jumbotron h1").text("Javascript is connected");
```

Warning: solution ahead ...
try it yourself first!

Select the element, then call `text()`

```
$("#testjs").click(function(e) {  
    $(".jumbotron h1").text("Javascript has taken  
control");  
    $("#testjs").text("Please wait...");  
});
```

Adding a CSS class

- Select the element, then call `addClass()`
- Let's change the jumbotron subtitle style when the user clicks the button:

```
$("#testjs").click(function(e) {  
    $(".jumbotron h1").text("Javascript has taken  
control");  
    $("#testjs").text("Please wait...");  
    $(".jumbotron p").addClass("active");  
});
```

Nothing happens. Why?

- We haven't defined what that class does yet.
- Open up lab3/static/css/introHCl.css and define a style for paragraph tags with class "active" that are descendants of a node of class jumbotron

```
.jumbotron p.active {  
    color: #1f6ddd;  
}
```

human-computer interaction · social computing · crowdsourcing

Test Javascript

Add, remove, and toggle CSS classes

- You can remove classes with `removeClass()`
- Or, toggle them on and off with `toggleClass()`:

```
$("#testjs").click(function(e) {  
    $('#jumbotron h1').text("Javascript has taken  
control");  
    $("#testjs").text("Please wait...");  
    $(".jumbotron p").toggleClass("active");  
});
```

human-computer interaction · social computing · crowdsourcing

Test Javascript

practice

Effects

Controlling content visibility

- Your task: in `projectClick()`, instead of complaining to the user the second time they click, remove the project description. Animation type is up to you.
- Immediate
 - `$("#myelement").hide()`: hide an element on the page
 - `$("#myelement").show()`: bring it back
 - `$("#myelement").toggle()`: toggle between them
- Animated
 - `$("#myelement").fadeIn()`
 - `$("#myelement").fadeOut()`

Stretch goal

Toggle (show/hide) project descriptions (including media, e.g. images) when you click on a project