# FE101

Intro to jQuery

http://www.teaching-materials.org/jquery/

# jQuerify

A bookmarklet to play with jQuery in your browser

http://www.learningjquery.com/2006/12/jquerify-bookmarklet/

# Exercise: Disrupting Techcrunch

Using jQuerify, make your very own Techcrunch article by replacing the content of a real article with your own!  Be sure to replace the:

- article title
- image
- article text
- byline (author name)

# Including the jQuery library

- At the end of the `<body>` section of your file, include a copy of the jQuery library:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>
```

- In this case, our copy of the library is from the Google Content Delivery Network, meaning it should be loaded pretty quickly no matter where our user is located

# The Document Object Model, or DOM

- The DOM is a standard for interacting with and representing objects in HTML, XHTML, and XML
- It has been around since the “browser wars” of the 90s
- Web browsers implement a standard version of it to normalize how HTML elements are interacted with
- jQuery makes it incredibly easy to select elements in the DOM, which include practically any HTML tag or element

# Selecting elements in the DOM with jQuery

- jQuery can be used to select elements with a similar syntax to CSS. Try this first line in the JavaScript console of a page that has jQuery included:

```
$("body")
>[><body>...</body>]
```

- The format for selecting elements is:

```
$("<element name, class, id, or XPath>")
```

- Here are some examples:

# Selecting elements dissected

- By calling `$`, we're just using a shorthand way of accessing the jQuery library's main function

- Try typing just a `$` in the JavaScript console and pressing enter to see the function itself

- The `("<element identifier")` after `$` is simply an argument being passed to the library's

# The `$(document).ready()` snippet

- This snippet ensures that your code doesn’t run until your document fires a ready event
- By selecting the document, you’re able to listen to events fired at the topmost level of the page
- The ready event fires when all of the page’s DOM elements are loaded, even if multimedia elements aren’t fully loaded
- By encapsulating all jQuery to only run when the document is ready, the elements you're selecting will definitely be on the page

# Let’s learn some cool JavaScript things now

…that will help us build an image slider!

# Working with jQuery results arrays

- Assume we have x images on the page
- If we select `$("img")`, we'll get back an array of image elements
- If we want to operate on a specific one, we use eq()

# Working with jQuery results arrays

- Remember that the results returned from a jQuery selector are an array!

- We can count the amount of results returned from a jQuery selector by using the `.length` method

```
$(“img”).length
```

# Exercise

- Using what you've just learned, put together a primitive image slider that can go from one image to another when you click on it

- Building upon this, see if you can make the slider "loop"

- Building upon that, see if you can have it slide

# `setInterval`

- The `setInterval` method is a method on window, meaning you can call it practically anywhere inside of your JavaScript code

- It allows you to have a function run every x milliseconds

# setInterval

Here's a code sample:

```
setInterval(function(){
  alert("YOU'VE WON!")
}, 1000)
```

# `fadeIn()`, `fadeOut()`

Remember that jQuery fadeIn() and fadeOut() bring in and out elements on the screen by slowly changing the `opacity` CSS attribute

# fadeIn(), fadeOut()

You can also have behavior triggered when the fadeIn or fadeOut is complete:

```
$(“.img”).first().fadeIn(1000,
function(){
   console.log(“something to do when
fadeIn is done”)
})
```

**`window.clearInterval(intervalId)`**

`clearInterval` will clear any interval function that you have running on the page

It takes an `intervalId` as an argument

`setInterval` returns an `intervalId`

## **window.clearInterval(**intervalId**)**

```
//every 3 seconds, set a ‘hi’ alert.
//setInterval returns an intervalId
var intervalId = setInterval(function(){
  alert(‘hi’)
}, 3000)

//use stored intervalId to stop the alert
//when the button with the class //.some-
button is pressed
```

# Exercise

- Now set up your slider to play automatically - try to be organized and use functions!

- Using a Font Awesome `<div>` or just a button element or two, have the slider play or pause when one of these elements is clicked using the `clearInterval` function

# Working with element arrays

**.get()**

- When we call `$("img")`, we get back an array of image elements on the page

- Goal: find out the following:
  - what image is being displayed on the page currently
  - that image's position in the list of `<img>` tags

- Any ideas?

# Working with element arrays

**.get()**

- In order to use the native `.indexOf()` JavaScript array method, which shows us the position of an element in an array:
  - we need to have an actual JavaScript array, not the object that jQuery returns!

- To make sure we have a normal JavaScript array, use the `.get()` method on `$("img")`

# …one more thing

How do we filter to find the exact image element being displayed? The :visible pseudo-class!

```
$(“img:visible”)
```

should only return the visible image, AKA the image the slider is currently on

# Working with element arrays

**.get()**

- Goal: find out the following:
  - what image is being displayed on the page currently

```
$("img:visible")
```

  - that image's position in the list of `<img>` tags on the page

# Exercise

- Using this information, we can accurately figure out which image is currently being displayed on the slider

- This opens up access to the following functionality:
  - Accurate play/pause
  - Next/Previous buttons

# Using a pre-built solution

If you'd rather *not* build your own image slider…

http://wowslider.com/

http://dev7studios.com/plugins/nivo-slider/