WEEK 8 | PROGRAMMING
Introduction to JQuery

Shawn Park & Jeff Zhan

{ } Web Design DeCal

DESIGN MEETS PROGRAMMING

Review Functions & Loops

Today's Outline

1. Review JavaScript
2. Introduction to jQuery
3. jQuery selectors
4. Accessing Content
5. Events
6. Effects

Goal Today:

Learning jQuery

Intro to jQuery

jQuery

- Functions and variables are cool… but can we do more with JavaScript?
- **jQuery,** a JavaScript library, comes in handy!
- jQuery allows you to access HTML elements and do all sort of things!
  - Click events
  - Keyboard presses
  - Scrolling animations
  - Modal popups
  - Toggling menus
  - Smoother hovers and animations
  - Etc.

jQuery

- How do we add jQuery to our page?
- Either insert the first element below, or download the jQuery .js file and link to it
  - **Important!** Add this before any links to your own .js files!

```
<script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
```

Above: Link to an online copy of jQuery

```
<script src="assets/js/jquery-1.11.0.min.js"></script>
```

Above: Link to a local copy of jQuery

jQuery
Selectors & Accessing Content

jQuery - Selectors

- To use jQuery, you have to access HTML elements first
- Syntax
  - **$(selector)** – The $ tells the browser this is jQuery
  - Replace *selector* with the CSS selector in quotes (single or double)

```
$('body')
$('.box')
$('#container .box')
$('a')
```

Like in CSS, we must select elements before we can apply actions to them

- Once you select an element, you can access different values, properties, or functions to the element

jQuery – Accessing Content

- 3 common ways to access element content
  - **.text()** – Grabs all text inside an element
  - **.html()** – Grabs all HTML inside an element
  - **.val()** – Grabs the value attribute or an element (useful for inputs)

```
<div id="box">
    <div class="title">
        Hello World!
    </div>
</div>
```

$('#box").text() → Hello World!

$('#box").html() →
Hello World!

- You could test the above using **console.log($('#box').text())** or **alert($('#box').html());**

jQuery – Accessing Content

- .val()
  - Useful to get search results, form input, etc.

<input type="text" id="name">

$('#name").val() → Jeff

jQuery – Setting Content

- Likewise, you can set content

```
$('#box').html('<p>New Title</p>');

$('#email').val('sample@example.com');
```

Demo

Follow along:
jsfiddle.net/MBK4g/1/

jQuery

- Accessing content, or setting content is useful!
- But how so? When **should** we access an element's value? Or HTML? Or text?
- When using **Events (actions user does)**
  - *E.g. Click a button to replace the HTML on a page*
  - *Hit enter to get User's Login information*
  - *Scroll down page and change CSS*

jQuery Events

jQuery – Event Listeners

- **Event Listeners** are always on the page, “listening” for an event to occur (and react to these events)
- Events include:
  - Clicking
  - Key presses
  - Scrolling
  - Hovering (smoother than CSS, more powerful)
  - Tons more

jQuery – Event Listeners

- Event Listeners are *attached* to jQuery selected elements
  - *E.g. $('#box').click(function() { … });*
- Syntax:
  - .click(function(){ … });
  - .hover(function(){ … });
  - .scroll(function(){ … });
- Always pass a **function** to an event listener! (*event handlers*)

```
$('#box').click(function(){
        alert("You are so cool.");
});
```

Browser binds **click** event to **#box**. It continues to "listen" as user is on page, and when user clicks on #box, a popup will appear!

Demo

Follow along:
jsfiddle.net/suX84/3/

jQuery – Event Listeners

- More examples

```
$('#box').hover(function(){
    alert($(this).html());
});
```

Inside your event listener function, you can refer to the current element (say **#box**) with simply **$(this)**.

```
$('.box').hover(function(){
    alert($(this).html());
});
```

This hover now applies to all class **box**. **$(this)** only refers to the element you hovered on, not to all elements with class box!

```
$('.box').click(function(){
    $('.box2').hide();
});
```

Don't have to use **$(this)!**
Here, we click on any element with class **.box**, and you hide all elements with class .**box2**

jQuery Effects

jQuery – Effects

- You have seen **ways to access content**
  - *.html(), .text(), .value()*
- You have seen **events**
  - *.click(), .hover(), .scroll()*
- Let's talk about **effects**
  - *Hiding elements*
  - *Showing elements*
  - *Fade In, Fade Out*
  - *Sliding up, Sliding Down*

jQuery – Effects

- .show(), .hide()
  - Extremely useful and common
  - Essentially just adds/removes the CSS property **display: none** to an element

```
$('#menu-show').click(function(){
        $('#menu').show();
});

$('#menu-hide').click(function(){
        $('#menu').hide();
});
```

jQuery – Effects

- .fadeIn(*time*), .fadeOut(*time*)
  - Like show() and hide(), but gradually fading
  - Time takes in milliseconds. 1000 is a second

```
$('#menu-show').click(function(){
        $('#menu').fadeIn(500);
});

$('#menu-hide').click(function(){
        $('#menu').fadeOut(500);
});
```

jQuery – Effects

- .slideUp(*time)*, .slideDown(*time)*
  - Like fadeIn() and fadeOut(), but height changes over time
  - Again, time in milliseconds

```
$('#menu-show').click(function(){
        $('#menu').slideDown(500);
});

$('#menu-hide').click(function(){
        $('#menu').slideUp(500);
});
```

jQuery – Effects

- .css*(style, value)*
  - Change the CSS
  - *Ex: $('body').css('color', 'red');*

```
$('#menu').click(function(){
        $(this).css('background', '#333');
});
```

jQuery Initialization

jQuery - Initializing

- Initializing jQuery
  - So you added the **<script … ></script>** for jQuery
  - Now, before we can do anything, you must let the browser know you are using jQuery in your JavaScript files
  - Start your .js files with:

```
$(document).ready(function(){

        … your javascript here …

});
```

$ - Calls the jQuery library
(document) – Selects the entire HTML page
.ready – When *all* the HTML is loaded…
function(){ … } – …do everything inside this function

jQuery

Your HTML

```
1  <html>
2  <head>
3      <title>Web Design Decal</title>
4      <script src="http://code.jquery.com/jquery-1.11.0.min.js"></script>
5      <script src="assets/js/site.js"></script>
6  </head>
7  <body>
8
9  </body>
10 </html>
```

Your Linked JavaScript File (assets/js/site.js)

```
1  $(document).ready(function(){
2      /* Your jQuery/JavaScript here */
3
4      $('body').click(function(){
5          alert('I am the body.');
6      });
7
8  });
```

jQuery

- Sample

```
$(document).ready(function(){

        $('#menu-button').click(function() {
                $('#menu').show();
        });

});
```

Summary

- Syntax to Select
  - $(selector)
- Accessing Content
  - .html(), .text(), .value
- Events
  - .click(), .hover(), .toggle()
- Effects
  - .show(), .hide(), .fadeIn(), .slideDown(), etc.

All lecture material, handouts, and homework can be found at:
http://www.thewebdesignworkshop.co

Bonus Slides

jQuery – Multiple CSS

- .css(style,value)
  - Good for 1 style. For multiple styles, use:
- .css({ style: value, style: value, etc. })
  - Style name must be in Camel-case (no hyphens, letter after hyphen capitalized)

Sample:

```
$('#button').click(function() {
        $('#menu').css({ backgroundColor: 'red', fontWeight: 'bold', color:
'#333'
});

// Notice style names don't need quotes, but values do
```

jQuery – Toggling

- Toggling is essentially clicking once to do one action, clicking again to do something different
- jQuery 1.8 and earlier supported this event, and it was super easy to code (just two functions for the event):

```
$('#menu-button').toggle(function(){
        $('#menu').slideDown(500);
}, function() {
        $('#menu').slideUp(500);
});
```

jQuery – Toggling

- For jQuery 1.9+, a work-around is shown below, making use of *.hasClass(), .addClass(), and .removeClass()*

```
$('#menu-button').click(function(){
        if($(this).hasClass('active')) {
                $('#menu').slideUp(500);
                $(this).removeClass('active');
        } else {
                $('#menu').slideDown(500);
                $(this).addClass('active');
        }
});
```

- Notice we take advantage of this class 'active' in simulating a toggle

Bonus Slides (extra extra!)

jQuery – Global Scope

- The Global Scope, in terms of variables, is everything in your .js file **not** within a function

```
var name = 'John';
// The above is using name in the global scope

function names() {
        var name = 'John';
}
// The above is using name within a defined scope, names()
```

- Putting variables inside functions unclutters the global scope
- You can't just do **console.log(name)** anymore since the scope of name is accessible only in **names()**, not outside

jQuery – Global Scope

- Similarly, functions can clutter the global scope as well
- One trick is to keep all functions within an **object**

```
var obj = {
        title: 'Names',
        names: function() {
                alert('John');
        }
}
obj.title // Get 'Names'
obj.names() // alerts 'John'
```

- Notice the use of : to separate vars and their assignments
- Notice that we broke **function names()** into **names: function()**
  - But we still call it with **names()**

jQuery – Global Scope

- To add arguments to a function like this, simply add an argument

```
var obj = {
        title: 'Names',
        names: function(name) {
                alert(name);
        }
}
obj.title // Get 'Names'
obj.names('John') // alerts 'John'
```

- **function names(name)** in this case broke down into **names: function(name)**

jQuery – Global Scope

- This makes your code more structured
- An example of using jQuery to call such a function:

```
var funcs = {
        names: function(name) {
                alert(name);
        }
}

$('#button').click(funcs.names('John'));
```

- Instead creating a separate **function(){ … }** for the #button, you reference it to an existing function we created in the object
- This method is more **modular** too. Functions are more reusable!

jQuery – Global Scope

- Great resource for a sample JavaScript structure:
  - http://css-tricks.com/how-do-you-structure-javascript-the-module-pattern-edition/
- Know that for this class, we won't concern ourselves with global scope and structure
- For that, there are many helpful resources on the web ☺