JQUERY METHODS & EVENTS

GOALS FOR THIS UNIT

1. Review
2. Ready method
3. More on Methods
4. Effects
5. Events

REVIEW

READY METHOD

WHY WE NEED IT

Just like plain ol' JavaScript, jQuery needs the browser to construct the DOM before it can select anything.

THE SOLUTION

We can place our script tags at the very end of the body of our HTML. That way, all the HTML has already been loaded before the scripts are run.

THE (PORTABLE) SOLUTION

Wrapping our code in the `.ready()` method means the browser will still wait for the DOM to load before running any scripts, regardless of where they're placed.

```
$(document).ready(function(){
  // JavaScript-y things here
});
```

THE (PORTABLE, SHORTHAND) SOLUTION

We can simplify this method by using a shorthand one! Code inside this very common shorthand method will be run automatically once the page loads.

```
$(function(){
  // JavaScript-y things here
});
```

READY VS. EVERYBODY

Other options include relying on the load event (if your script depends on other assets) or simply placing scripts at the very end of the body of your HTML. In the latter case, developers will often still use the `.ready()` method in case the script tag is moved later.

RETRIEVING CONTENT

The `.html()` method retrieves the HTML inside the first element matched by the selector. This includes any descendants. It can also be used to update that content.

```
< div class="demo-container" >
  < div class="demo-box" >Demonstration Box < /div >
< /div >
```

```
$( "div.demo-container" ).html();
```

The above would return:

```
< div class="demo-box" >Demonstration Box < /div >
```

The `.text()` method retrieves the HTML inside the first element matched by the selector. This also includes any descendants. It can also be used to update that content.

```
$( "div.demo-box" ).text("New Demo Box Text!");
```

The above would change the text from "Demonstration Box" to "New Demo Box Text!".

UPDATING CONTENT

The `.html()` and `.text()` methods can not only retrieve content, but update it as well.

DEMO

The `.replaceWith()` method replaces matched elements with new content and returns the replaced elements.

The `.remove()` method removes any elements in the matched set.

INSERTING CONTENT

The `.before()` method inserts content before the matched element(s).

```
$( ".inner" ).before( "<p>Test</p>" );
```

The `.after()` method inserts content after the matched element(s).

```
$( ".inner" ).after( "<p>Test</p>" );
```

The `.prepend()` method inserts content inside the matched element(s) immediately after the opening tag.

```
$( ".inner" ).prepend( "<p>Test</p>" );
```

The `.append()` method inserts content inside the matched element(s) immediately before the closing tag.

```
$( ".inner" ).append( "<p>Test</p>" );
```

DEMO

JQUERY EFFECTS

EFFECTS

Effects generally enhance the interactive components of our sites. These days, we can accomplish many of the same tasks with pure CSS, but certain older browsers don't play nicely with our shiny CSS3 animations.

MAKING SPACE

When an element is hidden, other elements may move to take up the empty space left in the page. When an element is shown, others will move to make room.

TOGGLING

Methods with `toggle` in their name will look at the current state of the selected element(s) and switch to the opposite.

BASIC EFFECTS

Method	Description
.hide()	Hide the matched elements.
.show()	Display the matched elements.
.toggle()	Display or hide the matched elements.

CUSTOM EFFECTS

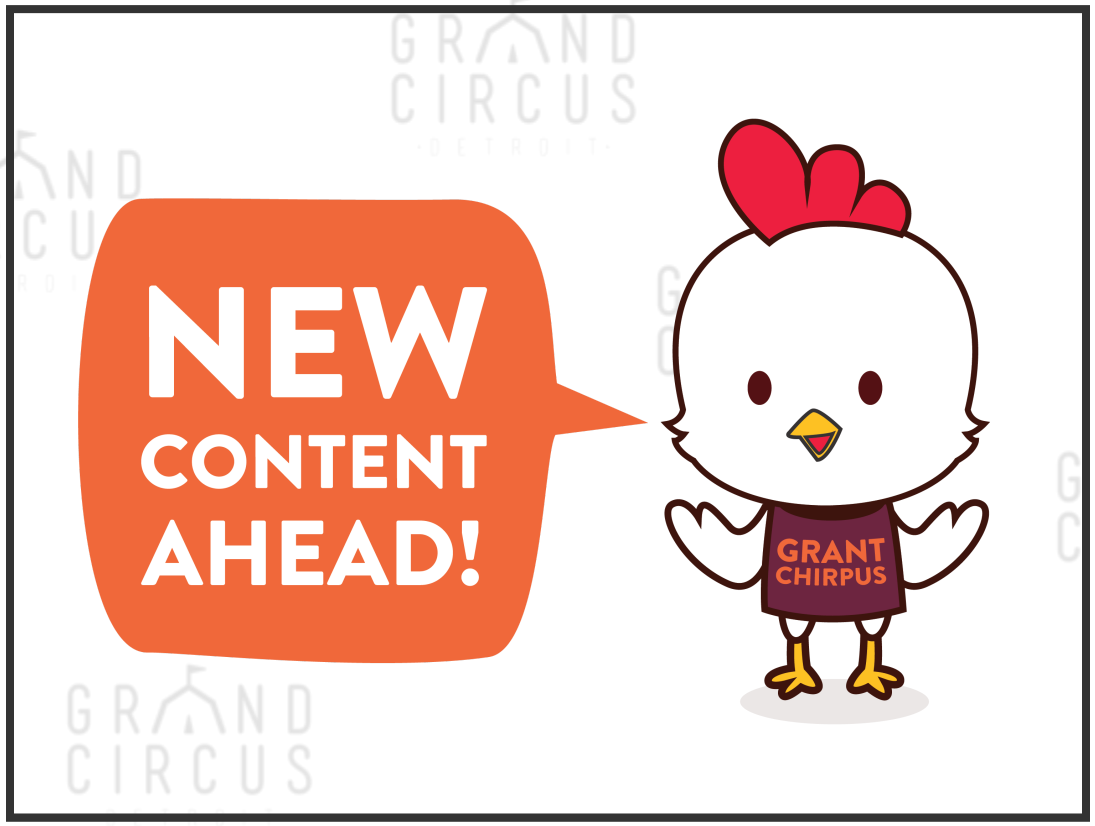
Method	Description
.animate()	Perform a custom animation of a set of CSS properties.
.delay()	Set a timer to delay execution of subsequent items in the queue.
.finish()	Stop the currently-running animation, remove all queued animations, and complete all animations for the matched elements.
.stop()	Stop the currently-running animation on the matched elements.

FADING EFFECTS

Method	Description
.fadeIn()	Display the matched elements by fading them to opaque.
.fadeOut()	Hide the matched elements by fading them to transparent.
.fadeTo()	Adjust the opacity of the matched elements.
.fadeToggle()	Display or hide the matched elements by animating their opacity.

SLIDING EFFECTS

Method	Description
.slideUp()	Hide the matched elements with a sliding motion.
.slideDown()	Display the matched elements with a sliding motion.
.slideToggle()	Display or hide the matched elements with a sliding motion.


NEW
CONTENT
AHEAD!
GRANT
CHIRPUS

JQUERY EVENTS

EVENTS

jQuery's event methods are used to cause certain code to take effect when the user interacts with the browser. This code is called a **handler** because it tells JavaScript how to handle the event.

ONE METHOD TO RULE THEM ALL

As of jQuery 1.7, the `.on()` method is all we need to attach handlers to events. Previously, there were a few different ways to do it, including `bind()`, `live()` and `delegate()`. Those are now deprecated.

THE BREAKDOWN

1. First, we need to select the affected element(s).

```
$('button')
```

THE BREAKDOWN

2. We'll use the `.on()` method to handle the event.

```
$('button').on()
```

THE BREAKDOWN

3. The first argument we pass to the `.on()` method is the event we want to respond to.

```
$('button').on('click')
```

THE BREAKDOWN

4. The second argument we pass to the `.on()` method is the code we want to be triggered by the event in the form of either a named or anonymous function.

```
$('button').on('click', function(){
  console.log('You clicked a button!');
});
```

SHORTCUT METHODS

Most of the events also have shortcut methods, which do the same thing.

```
$('button').click(function(){
  console.log('You clicked a button!');
});
```

is the same as

```
$('button').on('click', function(){
  console.log('You clicked a button!');
});
```

THE EVENT TARGET

Sometimes you need to know what HTML element was clicked or where the mouse moved. jQuery passes an *event* to your function. This event includes a lot of information about what happened. One bit of information is the `event.target` which indicates what was clicked.

```
$('button').on('click', function(event){
  console.log('This element was clicked: ' + event.target);
});
```

THIS

There is another way to identify what was clicked. Just use the `this` keyword.

```
$('button').on('click', function(){
  console.log('This element was clicked: ' + this);
});
```

EVENT.TARGET VS. THIS

Consider the code below. If I click the word "First", am I clicking the "first" div or the "wrapper" div?

```
<div id="wrapper">
  <div id="first">First</div>
  <div id="second">Second</div>
</div>
```

- `event.target` will be the *most specific* element.
- `this` will always be the element for which we *registered* the event. < DEMO >

HOMEWORK

From JavaScript & jQuery:

- Chapter 8: 367-408