GENERAL ASSEMBLY

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

1. Pull changes from the `svodnik/jsd5` repo to your computer
2. Navigate to the `starter-code` folder

DOM/JQUERY CONTINUED & TEMPLATING

LEARNING OBJECTIVES

At the end of this class, you will be able to

- Implement advanced jQuery events
- Use event delegation to manage dynamic content.
- Use implicit iteration to update elements of a jQuery selection, and use chaining to place methods on selectors.
- Add a templating language to your projects for better and more abstracted content manipulation.

AGENDA

- jQuery events
- Event delegation
- Templating

Checkin and questions

- The **most significant thing I've learned** about the DOM and jQuery is ________________.
- My **biggest outstanding question** about the DOM and jQuery is ________________.

Think about events that can trigger a change in a web page.

- Events you’ve experienced (such as `click`)
- Events you imagine would be useful
- Events you’ve heard of (even if you don’t know exactly how they work)

EVENTS

DOM EVENTS WE'VE USED SO FAR

`load`	the page has finished loading
`click`	a mouse button has been clicked and released when the pointer is over an element

A SELECTION OF OTHER DOM EVENTS

Mouse Events

```
click
dblclick
mouseenter
mouseleave
```

Form Events

```
submit
change
focus
blur
```

Keyboard Events

```
keypress
keydown
keyup
```

Document/Window Events

```
load
resize
scroll
unload
```

CHAINING

- jQuery lets us attach one or more methods to a selector, so we can combine multiple actions into a single statement

```
var $mainCaption = $('<p>');
var $captionWithText = $mainCaption.html('The Bagged Sloth');
var $fullCaption = captionWithText.addClass('main-caption');
```

becomes

```
var $fullCaption = $('<p>').html('The Bagged Sloth').addClass('main-caption');
```

EXPLICIT ITERATION

- We can use the jQuery `.each()` method to iterate through a jQuery collection

```
$listItems.each(function() {
  var $qed = $(‘<span>’).html('&there4;');
  $(this).append($qed);
});
```

- This works just like a `for()` loop in vanilla JavaScript

IMPLICIT ITERATION

- On a selector that returns a jQuery collection, chain a method
- This method is applied iteratively to each element in the jQuery collection, *but without needing to explicitly write code that iterates*
- This is known as **implicit iteration**

```
    var $qed = $(‘<span>’).html('&there4;');
    $listItems.append($qed);
```

EVENT DELEGATION

- When the page loads, we can set events on a set of elements
- However, if we add a sibling element later, the event is not set on it

```
var $listItems = $(‘#contents-list li');

$listItems.on('mouseenter', function(event) {
  $(this).siblings().removeClass(‘active’);
  $(this).addClass('active');
});
```

EVENT DELEGATION

- We can ensure that events are attached to elements added to the DOM later by selecting the parent element and specifying the child elements within the `on()` method arguments
- This is known as **event delegation**

```
var $listElement = $('#contents-list');

$listElement.on('mouseenter', 'li', function(event) {
  $(this).siblings().removeClass('active');
  $(this).addClass('active');
});
```

Selector changed from `'#contents-list li'`

New argument `'li'` added to `on()` method

ATTACHING MULTIPLE EVENTS WITH A SINGLE EVENT HANDLER

- We could write a separate event handler for each event on an element:

```
var $listElement = $(‘#contents-list');

$listElement.on(‘mouseenter', ‘li’, function(event) {
  $(this).siblings().removeClass(‘active’);
  $(this).addClass('active');
});

$listElement.on('mouseleave', ‘li’, function(event) {
  $(this).removeClass('active');
});
```

ATTACHING MULTIPLE EVENTS WITH A SINGLE EVENT HANDLER

- Grouping all the events for an element in a single event handler makes our code more organized and is faster

```
var $listElement = $('#contents-list');

$listElement.on('mouseenter mouseleave', 'li', function(event) {
  if (event.type === 'mouseenter') {
    $(this).siblings().removeClass('active');
    $(this).addClass('active');
  } else if (event.type === 'mouseleave') {
    $(this).removeClass('active');
  }
});
```

BREAK (5 MINUTES)

TEMPLATING

SEPARATION OF CONCERNS

- Programming principle of keeping different aspects (or **concerns**) of an application separate
- Many ways to do this
- One common separation is between data (the information we're presenting) and view (the code that determines how data is presented)
- We should be able to change the code for one concern without affecting the code for the other

TEMPLATING

- Lets us reference a snippet of code and populate it with data before adding it to the DOM
- The code snippet includes both HTML elements and JavaScript code
- The data comes from one or more JavaScript objects

TEMPLATING LIBRARIES

- A number of templating libraries are widely used in JavaScript
- We will be using Handlebars
- Documentation at handlebarsjs.com

IMPLEMENTING A HANDLEBARS TEMPLATE

1. Create or reference an object that stores the content
2. Create the template
3. Select the template content
4. Compile the template
5. Pass the object to compile to Handlebars
6. Add the new compiled element to the DOM

1. CREATE/REFERENCE CONTENT

```
var helloStatement = {
  helloTitle: "Hello world",
  helloContent: "GA JS class is just awesome”
};
```

2. CREATE TEMPLATE

```
<script id="hello-world-template" type="text/x-handlebars-template">
  <h1>{{helloTitle}}</h1>
  <p>{{helloContent}}</p>
</script>
```

3. SELECT TEMPLATE CONTENT

```
var source = $('#hello-world-template').html();
```

4. COMPILE TEMPLATE

```
var template = Handlebars.compile(source);
```

5. PASS OBJECT TO COMPILE TO HANDLEBARS

```
var compiledTemplate = template(helloStatement);
```

6. ADD COMPILED TEMPLATE TO DOM

```
$('body').append(compiledTemplate);
```

BREAK (5 MINUTES)

LEARNING OBJECTIVES - REVIEW

- Implement advanced jQuery events.
- Use event delegation to manage dynamic content.
- Use implicit iteration to update elements of a jQuery selection, and use chaining to place methods on selectors.
- Add a templating language to your projects for better and more abstracted content manipulation.

NEXT CLASS PREVIEW

Ajax and APIs

- Identify all the HTTP Verbs & their uses.
- Describe APIs and how to make calls and consume API data.
- Access public APIs and get information back.
- Implement an Ajax request with vanilla JS.
- Implement a jQuery Ajax client for a simple REST service.
- Reiterate the benefits of separation of concerns – API vs. Client.

Exit Tickets!

Q&A