IN4MATX 133: User Interface Software

Lecture 6:
jQuery and Advanced JavaScript

Today’s goals

By the end of today, you should be able to…

- Write code which edits the DOM using built-in JavaScript functions and jQuery
- Implement functional programming concepts in JavaScript like forEach, map, and filter

A2 Walkthrough

Socrative Quiz!

Enter your UCI Email when prompted name!!!
e.g.,

xxxxx@uci.edu

https://api.socrative.com/rc/CvereT

Manipulation in pure JavaScript is verbose

- If you’re editing a lot of tags, your code can get very long and difficult to read

jQuery

- Predefines methods for manipulating the DOM
  - Include <u>before</u> your own script
- Remember:
  - Integrity: hashes to ensure the downloaded file matches what's expected
  - Crossorigin: some imports require credentials, anonymous requires none

```
<script
  src="https://code.jquery.com/jquery-3.3.1.min.js"
  integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
  crossorigin="anonymous"></script>
```

jQuery selector

Use the `jQuery()` function to select DOM elements.
The parameter is a CSS selector String (like `querySelector`)

- Compared to JavaScript's document.getElementByID("…"), etc.

```
//selects element with id="foo" (e.g., <p id="foo">)
var fooElem = jQuery('#foo');

//selects all <a> elements (returns an array)
var allLinksArray = jQuery('a');

//selects element with id="foo" (e.g., <p id="foo">)
var fooElem = $('#foo'); ← $() shortcut for jQuery()

//selects all <a> elements (returns an array)
var allLinksArray = $('a');
```

jQuery selector

jQuery handles all CSS selectors, as well as some additional pseudoclasses

```
$('#my-div') // by id
$('div') // by type
$('.my-class') // by class
$('header, footer') // group selector
$('nav a') // descendant selector
$('p.red') // scoped selector

$('section:first') // first <section> element
                   // not a css selector!
```

jQuery and elements

Similar to pure JavaScript, jQuery provides methods to access and modify attributes and CSS

```
//Pure JavaScript
var txt = document.querySelector('#my-div').textContent;
document.querySelector('#my-div').textContent = 'new info!';
document.querySelector('#my-div').innerHTML = '<em>new
html!</em>';

//jQuery
var txt = $('#my-div').text();        //get the textContent
$('#my-div').text('new info!');       //change the textContent
$('#my-div').html('<em>new html!</em>'); //change the HTML
```

jQuery and the DOM tree

```
//create an element (not in DOM)
var newP = $('<p class="new">This is a new element</p>');

//add content to DOM
$('main').append(newP); //add INSIDE, at end
$('main').append('<em>new</em>'); //can create element on the fly
$('main').prepend('<em>new</em>'); //add INSIDE, at beginning
```

Works without closing tag

```
$('main').before('<header>'); //insert BEFORE

$('footer').insertAfter('main'); //insert target AFTER param
```

Selects existing element, so will move!

```
$('main').wrap('<div class="container"></div>'); //surround

$('footer').remove(); //remove element
$('main').empty(); //remove all child elements
```

jQuery event handling

jQuery also provides convenience methods for registering event listeners

Like `addEventListener('click')`

```
$('#button').click(function(event) {
    console.log('clicky clicky!');

    //what was clicked
    var element = $(event.target);
});
```

`event.target` is equivalent to `this`

Get as jQuery-style element to call jQuery methods on it ("wrap it")

Document ready: JavaScript

- Remember earlier: your script should wait until after the page has loaded
- Otherwise, elements you're trying to access might not exist

```
<head>
 <script>
  function pageLoaded()  {
   alert("Page Loaded!");
  }
 </script>
</head>
<body onload="pageLoaded();">

</body>
```

Document ready: jQuery

```
$(document).ready(function() {
  //your program goes here
  //this need not be an anonymous function

});
```

Document ready: jQuery

```
//shortcut: just pass the function to the jQuery method
$(function() {
  //your program goes here
  //this need not be an anonymous function

});

//shortest cut: use the abbreviated syntax
$(() => {
  //your program goes here
  //this need not be an anonymous function

});
```

Importing JavaScript

- When your script uses document ready, convention is to load it in the `<head>` tag
  - Important to think about ordering, particularly for libraries
  - e.g., import JQuery before you use it in a script

```
<head>
 <script src="https://code.jquery.com/jquery-3.6.3.min.js"></script>
 <script src="index.js"></script>
</head>
```

jQuery effects

jQuery supports adding transitions to modify the appearance of effects

```
$('#id').fadeIn(1000);    //fade in over 1 second
$('#id').fadeOut(500);    //fade out over 1/2 sec
$('#id').slideDown(200); //slide down over 200ms
$('#id').slideUp(500);    //slide up over 500ms
$('#id').toggle();        //toggle whether displayed
```

And much, much, more:

https://api.jquery.com/

Let’s discuss some advanced JavaScript concepts…

Functions are objects

```
//assign array to variable
var myArray = ['a','b','c'];


var other = myArray;

//access value in other
console.log( other[1] ); //print 'b'
```

```
//assign function to variable
function sayHello(name) {
   console.log("Hello, "+name);
}

var other = sayHello;

//prints "Hello, everyone"
other('everyone');
```

Functions are objects

```
//assign array to variable
var myArray = ['a','b','c'];


var other = myArray;

//access value in other
console.log( other[1] ); //print 'b'
```

```
//assign function to variable
var sayHello = function(name) {
   console.log("Hello, "+name);
}

//second variable, same object
var greet = sayHello;

//execute object named `greet`
greet('everyone');
    //prints "Hello, everyone"
```

Functions are objects

```
var obj = {};
var myArray = ['a','b','c'];


//assign array to object
obj.array = myArray;

//access with dot notation
obj.array[0]; //gets 'a'


//assign literal (anonymous value)
obj.otherArray = [1,2,3]
```

```
var obj = {}
function sayHello(name) {
   console.log("Hello, "+name);
}

//assign function to object
var obj.sayHi = sayHello;

//access with dot notation
obj.sayHi('all'); //prints "Hello all"


//assign literal (anonymous value)
obj.otherFunc = function() {
    console.log("Hello world!");
}
```

Anonymous variables

```
var array = [1,2,3]; //named variable (not anonymous)
console.log(array); //pass in named var

console.log( [4,5,6] ); //pass in anonymous value,
nameless variable
```

Anonymous variables

```
//named function
function sayHello(person){
   console.log("Hello, "+person);
}

//anonymous function (no name!)
function(person) {
   console.log("Hello, "+person);
}

//anonymous function (value) assigned to variable
var sayHello = function(person) {
   console.log("Hello, "+person);
}
```

Anonymous variables

```
//anonymous functions often follow
an “arrow” (abbreviated) syntax
var sayHello = (person) => {
   console.log("Hello, "+person);
}

sayHello('IN4MATX 133’);
```

`this` keyword

- `this` usually refers to the object that the method was called on
- `this` is only preserved with abbreviated (arrow) syntax

```
var alice = {
  first: 'Alice',
  last: 'Jones',
  sayHello:()=> {
    console.log("Hello, I'm " + this.first);
  }
};

alice.sayHello(); //=> "Hello, I'm Alice"
```

Refers to containing object (alice)

Passing functions

Since functions are objects, they can be passed like variables

```
//anonymous function syntax
var doAtOnce = function(funcA, funcB) {
    funcA();
    console.log(' and ');
    funcB();
    console.log(' at the same time! ');
}

var patHead = function(name) {
    console.log("pat your head");
}

var rubBelly = function(name) {
    console.log("rub your belly");
}
```

No parens … (),
just passing variable

```
doAtOnce(patHead, rubBelly);
```

Callback functions

- A function that is passed to *another* function for it to "call back to" and execute

```
function doLater(callback) {
  console.log("I'm waiting a bit...");
  console.log("Okay, time to work!");
  callback();
}

function doHomework() {
  ...
};

doLater(doHomework);
```

Takes in a callback

Pass in the callback function

Callback function example: `forEach`

- To iterate through each item in a loop, use the `forEach` function and pass it a function to call on each array item

```
//Iterate through an array
var array = ['a','b','c'];
var printItem = function(item) {
   console.log(item);
}

array.forEach(printItem);  ← Callback

//more common to use anonymous function
array.forEach(function(item) {
   console.log(item);
});
```

Callback function example: `map`

`map` applies the function to each element in an array and returns a *new* array of elements returned by the function

```
var array = [1,2,3];
var squared = function(n) {
    return n*n;
};

array.map(squared); //returns [1,4,9]

//more common to do this inline:
array.map(function(n) {
    return n*n;
});
```

Callback function example: `filter`

`filter` applies the function to each element in an array and returns a *new* array of only the elements for which the function returns true.

```
var array = [3,1,4,2,5];

var isACrowd = array.filter(function(n) {
    return n >= 3;
}); //returns [3,4,5]
```

Callback function example: `reduce`

`reduce` applies the function to each element in an array to update an “accumulator” value. The callback function should return the “updated” value for the accumulator.

```
var array = [1,2,3,4];

var sum = array.reduce(function(total, current) {
    var newTotal = total + current;
    return newTotal;
}, 0); //returns 1+2+3+4=10
```

Today’s goals

By the end of today, you should be able to…

- Write code which edits the DOM using built-in JavaScript functions and jQuery
- Implement functional programming concepts in JavaScript like forEach, map, and filter

Utility functions

jQuery utility functions

- jQuery includes many utility functions to simplify syntax

```
//check if an item is in an array
$.inArray(4, [3,4,3] );

//this is like .filter, but works on old browsers
$.grep( [3,4,3], function(item) {
   return item > 3;
});

//iterate over arrays or objects -- works for either!
$.each( [1,3,3], function(key, value) {
   console.log('Give me a '+value);
});

$.each( {dept:'IN4MATX',num:'133'}, function(key, value) {
   console.log(key+' name: '+value);
});
```

http://api.jquery.com/category/utilities/

Even more utilities: Lodash

- A handy library for working with basic data structures

```
_.flatten([1, [2, [3, [4]], 5]]);
// => [1, 2, [3, [4]], 5]

var zipped = _.zip(['a', 'b'], [1, 2], [true, false]);
// => [['a', 1, true], ['b', 2, false]]

_.unzip(zipped);
// => [['a', 'b'], [1, 2], [true, false]]
```

https://lodash.com/