JavaScript, jQuery, and Ajax
JavaScript

- "Language of the web browser"
  - means of interfacing with DOM
- Loosely typed
- Powerful object literal notation (JSON)
- Prototypical inheritance
- Real-time DOM manipulation without HTTP requests
- Can initiate HTTP requests, too...
Just a normal programming language...

- Dynamically typed
  - `var answer = "42";`
  - `var y = "The answer is " + answer;`
- JS tries to be smart, sometimes fails
  - `console.log(answer - 1) returns "41"`
  - `console.log(answer + 1) returns "421"`
- Remedy with guidance on typing
  - `var answer = 42;`
  - `console.log(answer - 1) returns 41`
  - `console.log(answer + 1) returns 43`
Syntax

- `for(var i=0; i<10; i++){...}
- `for(var myProp in myObject){...}
- `while(condition){...}
- `do {...} while(condition);`
More syntax

- If, Else If, and Else

```java
if(condition){
    ...
    ...
} else if(condition) {
    ...
    ...
} else {
    ...
    ...
}
```
Error handling

try {
    doSomething();
} catch (e) {
    // Show errors here
} finally {
    // Always do this regardless of above
}
Error handling another [global] way

window.onerror = function ( message )
{
    document.getElementById("error").innerHTML = "New error: " + message;
}

Functions

function doSomething(theThing, theOtherThing){
    ...
    return "some value!"
}

Calling Functions

doSomething("something","another");

doSomething("something");

doSomething("something","another","dance!");

function myOtherFunction(){
    console.log(arguments);
}

myOtherFunction("a","b"); produces ["a","b"];  
var foo = function(){console.log("bar");};

foo(); //produces "bar"
Asynchronous

```javascript
foo();

function foo(){
    console.log("bar");
}

foo(); //this will not work

var function(){
    console.log("bar");
}
```
Objects

- var m = {"a": 3, "b": 42, foo: function() {
    console.log(this.a);
}};
m.foo(); //produces 3

- m.b = "hello";
m['a'] = "world";
Namespaces

• Global object
  – window in browsers

• Sub-namespaces
  – window.document to access DOM in browser
DOM Available onload

```html
<html>
<head>
<script>console.log(document.getElementById('foo'));</script>
<!-- Yields null -->
</head>
<body>
<p id="foo">test</p>
<script>console.log(document.getElementById('foo'));</script>
<!-- Yields <p id="foo">test</p> -->
</body>
</html>
```
Triggering with event listeners

```html
<html>
<head>
<script>
    document.addEventListener("DOMContentLoaded", function(event) {
        console.log(document.getElementById('foo'));
    });
</script>
</head>
<body>
    <p id="foo">test</p>
</body>
</html>
```
Triggering with event listeners

```html
<html>
<head>
<script>
function doit () {
    console.log(document.getElementById('foo'));
}
</script>
</head>
<body onload="foo()">
<p id="foo">test</p>
</body>
</html>
```
Interacting with the DOM

- `document.getElementById(id)`
  - recall id is unique
- `document.getElementsByClassName(className)`
  - returns array of all elements that match class
- `document.getElementById(id).getElementsByClassName(className)`
  - returns only children of element with id matching className
- `var myLink = document.getElementsByTagName("a")`
- `myLink.href` or `myLink["href"]`
- `myImage.src = "newImage.png"`
Gotcha: Global Variables

- Normally evil, required in JS (global)
- Language provides workarounds
  - `var myVariables = myValue;`
  - `window.myVariable = myValue;`
    - `window` is the global variable in the browser
  - `myvariable = myValue`
    - use variable without declaration
Gotcha: Scope

• Block syntax but no block scope
  – variable declared in a block is visible everywhere in the containing function
• best to declare variables at top of function
• In most languages, variables should be declared on first use
Gotcha: Auto semi-colon

• JS tries to work with malformed code
• return
  "foo";  //returns undefined
• return "foo";  //returns “foo”
• return
  ["2","3","4"];  
• return ["2","3","4"]
• return ["2",
  ["3","4"];
Reserved Words

abstract boolean break byte case catch char class const continue debugger default delete do double else enum export extends false final finally float for function goto if implements import in instanceof int interface long native new null package private protected public return short static super switch synchronized this throw throws transient true try typeof var volatile void while with
Reserved Words

- If used as keys for objects, they must be quotes
  - `myObject[foo]` is valid
  - `myObject[case]` is invalid
  - `myObject["case"]` is valid
- `myObject = {foo: 3};` is valid
  - `myObject = {case: 3};` is invalid
  - `myObject = {"case": 3};` is valid
- `var foo` is valid
  - `var case` is invalid
- Reserved words cannot be used with dot notation
  - `myObject.foo` is valid
  - `myObject.case` is invalid
  - `myObject."case"` is invalid
- More gotchas (see Appendix A of "JavaScript the Good Parts")
jQuery

- library commonly used to simplify JS operations
- overcomes some cross-browser compatibility issues
- included local library in web page prior to utilizing

```html
<script src="jquery.js"></script>
<script>
...your jquery code...
</script>
```
jQuery

- `$('#myLink')` equivalent to `document.getElementById('myLink');`
- `$('.items')` equivalent to `document.getElementsByClassName('items');`
- Many DOM manipulation functions:
  - `$('#myLink').css("color","red");` //sets inline css,
    `$('#myLink').css("color");` //gets the already set value
  - `$('#myLink').hide();` //removes change CSS of element to not be visible
  - `$('#myLink').remove();` //removes element from DOM
- Large API of DOM manipulation functions
AJAX

- Asynchronous JavaScript And XML
- Fetching content after the page has loaded
```javascript
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
    if (xhr.readyState == XMLHttpRequest.DONE) {
        if (xhr.status == 200) {
            document.getElementById("myText").innerHTML = xhr.responseText;
        }
        else if (xhr.status == 400) { // HTTP status
            alert('Error 400')
        }
        else {
            alert('Something other than HTTP 200 returned')
        }
    }
}
xhr.open("GET", "anotherURI.html", true); // third value specs synchronicity
xhr.send();
```
AJAX via jQuery

$.ajax({
    url: "anotherURI.html",
    done: function(response){
        $("myText").html(response);
    },
    fail: function() {...}
});

$.ajax({
    url: "changeThatValue.php",
    data: {"whichAttribute": thisAttribute, "toWhat": toThis},
    done: function(response){
        $("#myText").html(response);
    },
    fail: function() {...}
});
Code Practice

- JSLint - http://www.jslint.com/
- JSHint - http://jshint.com/
Node.js

- Server-side JavaScript
- System independent of the browser
- global score is not window
- Intended for web services handling many asynchronous requests
- JavaScript code not dependent on the DOM is portable to Node.js
Web Architecture

Dereference a URI, get a representation

URI
http://weather.example.com/oaxaca

Resource
Oaxaca Weather Report

HTTP GET

HTTP Response
(200 OK)

Representation

Metadata:
Content-type: application/xhtml+xml

Data:
<!DOCTYPE html PUBLIC "... http://www.w3.org/..."
<html xmlns="http://www..."
<head>
<title>5 Day Forecast for Oaxaca</title>
...
</html>
JavaScript makes requests for new resources after the initial page load:

- **HTTP GET**
- **Delivered to**

http://maps.google.com

Identifies

Represents

JavaScript
Deferred Representation

http://maps.google.com

Identifies

Represents

HTTP GET

Delivered to

HTTP Response (200 OK)

JavaScript
### JavaScript != Deferred

<table>
<thead>
<tr>
<th>Nondeferred</th>
<th>Deferred</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HTTP GET</strong></td>
<td><strong>HTTP GET</strong></td>
</tr>
<tr>
<td><code>&lt;HTML&gt;</code></td>
<td><code>&lt;HTML&gt;</code></td>
</tr>
</tbody>
</table>
| Contact Information:  
  Office: (765) 808-3456  
  Email: tyle@ic.edu  
  Show Photo Numbers Show E-Mail Address |  
  **CLICK HERE**  
  **AJAX**  
  200 OK |
| *This is HTML!* | *This is HTML!* |

```html
<html>
  
  <head>
    
  </head>

  <body>
    Click here
  
  </body>

</html>
```
JavaScript Standards/quality

• JSHint.com
  – http://jshint.com/
  – JavaScript code quality

• JSLint.com
  – http://www.jslint.com/
  – JavaScript & JSON code quality
JavaScript Code Error Checking

- Check/enforce your own standards:
  - https://github.com/feross/standard
- Has automatic standards checker
- Error Checker
- Standards enforcer
- Standards changer
JavaScript Security

- Frequent topic of debate
- Example: “target=_blank”
  - New page can gain control of the opener
  - e.g., window.opener.location

https://www.jitbit.com/alexblog/256-targetblank---the-most-underestimated-vulnerability-ever/