JavaScript
Lecture 3

Waterford Institute of Technology

June 9, 2016

John Fitzgerald

JavaScript Introduction

Topics discussed this presentation

- Scripts
- Chrome Developer Tools
- Functions
- jQuery
- Document Object Model (DOM)

Script Tags

Inserts program in html document

```
<script></script>
```

- Allows program in html
- Bad idea to place JavaScript in html
- Instead use source tags

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    ...
    ...
    <script src="js/jquery-2.0.0.js"></script>
    <script src="js/reportMap.js"></script>
  </body>
```

Script Tags

Where to locate

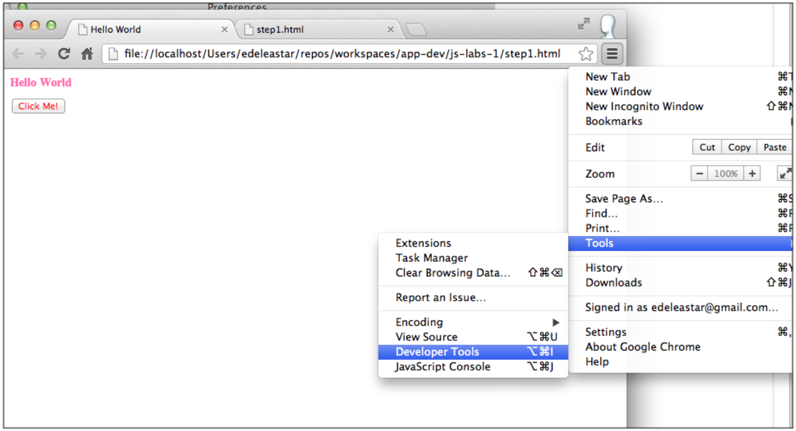
`<script></script>`

- Script files may have big impact on page load
- Place tags close as possible to bottom of body
- Place css <link> high as possible in head
- Reduce number of script files as much as possible
- Minify script files in release versions
  - Has big impact on load time

Chrome Dev Tools

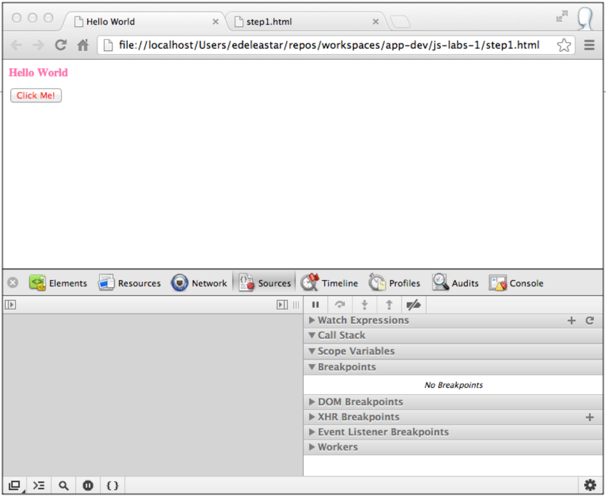
Web authoring & debugging tools

View->Developer Tools



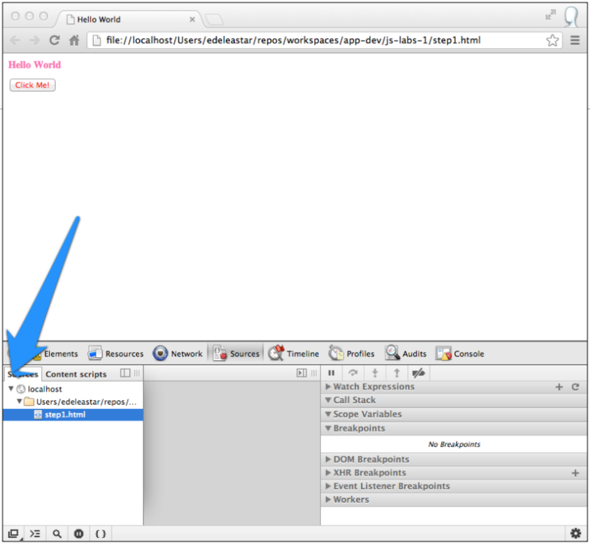
Chrome Dev Tools

Web authoring & debugging tools



Chrome Dev Tools

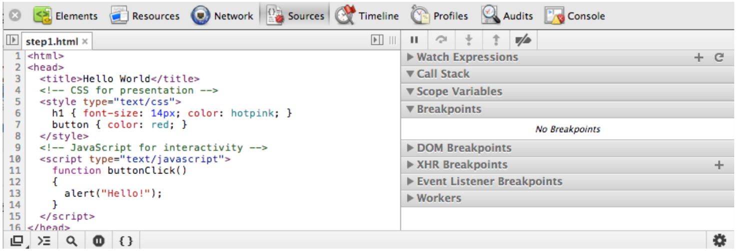
Web authoring & debugging tools



Chrome Dev Tools

Web authoring & debugging tools

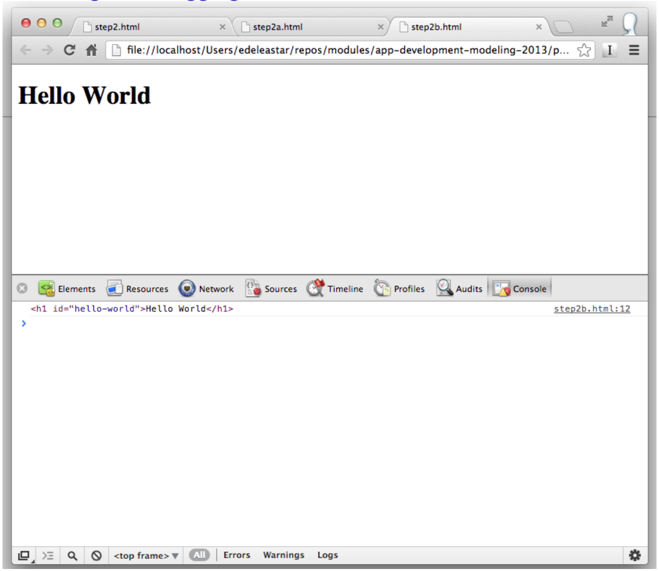
Source view in Developer Tools



```
<html>
<head>
  <title>Hello World</title>
  <!-- CSS for presentation -->
  <style type="text/css">
    h1 { font-size: 14px; color: hotpink; }
    button { color: red; }
  </style>
  <!-- JavaScript for interactivity -->
  <script type="text/javascript">
    function buttonClick()
    {
      alert("Hello!");
    }
  </script>
</head>
```

Chrome Dev Tools

Web authoring & debugging tools



Hello World

```
<h1 id="hello-world">Hello World</h1>                                     step2b.html:12
>
```

JavaScript
Functions

function

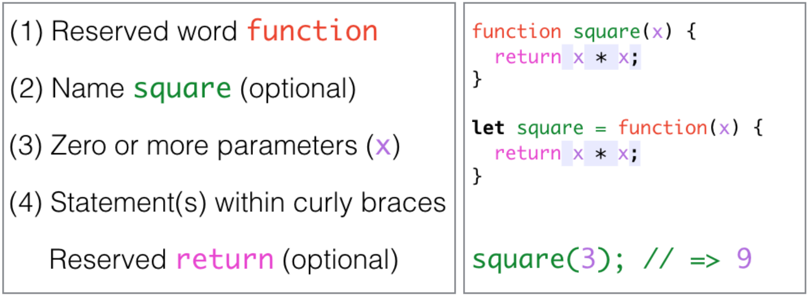
- Block of code defined once
- Invokable many times
- May include parameters
- Observe differences Java
- Functions attached to objects referred to as *methods*
- Functions are objects
  - Assignable to variable
  - Allowable as parameter

```
function square(x) {
  return x * x;
}

console.log(square(10)); // => 100
```

```
function add() {
  let counter = 0;
  function plus() {counter += 1;}

  plus();
  return counter;
}

console.log(add()); // => 1
```

JavaScript

Function has four parts



(1) Reserved word `function`

(2) Name `square` (optional)

(3) Zero or more parameters (`x`)

(4) Statement(s) within curly braces

Reserved `return` (optional)

```
function square(x) {
  return x * x;
}

let square = function(x) {
  return x * x;
}


square(3); // => 9
```

Function

Hidden parameters

Every function has 2 hidden parameters

- **this**
  - Reference determined by which of four available function invocation patterns used.
- **arguments**
  - Array type object containing all parameters.
  - Treat as obsolete, instead use rest arguments.
  - Rest arguments a real Array, not Array-like like arguments

```
let anObject = {
  value: 0,
  increment: function () {
    this.value += 1;
  },
};

// Output: 1
anObject.increment();
```

```
function aFunction(...args) {
  return args.length;
}

// Output: 2
console.log(aFunction(3, 4));
```

Functions

Invocation Patterns

Four function invocation patterns:

- 1. Method invocation
  - **this** bound to containing object
  - function is method - a property of containing object
- 2. Function invocation
  - **this** bound to global object
  - function property of global object
- 3. Constructor invocation
  - **this** bound to containing object
  - **new** not used: this bound to global
- 4. Apply invocation
  - Outside course scope

```
let anObject = {
  value: 0,
  increment: function () {
    this.value += 1;
  },
};

// method invocation
anObject.increment();
```

```
value = 0;
function increment() {
  this.value += 1;
};

// function invocation
increment();
```

JavaScript

this binding

Note: behaviour different in strict mode

```
// Function invocation: this bound to global object
function set(x) {
  this.x = x;
  console.log(x); // => 100
};
set(100); // sets global variable x to 100
```

```
// Here, because of strict mode, this is undefined
'use strict';
function set(x) {
  this.x = x; // => TypeError
  console.log(x);
};
set(100); // fails due to TypeError
```

JavaScript

this binding

```
// Method invocation: this bound to containing object
const myObj = {
  x: 100,
  set: function (x) {
    this.x = x;
    return this;
  },
};
myObj.set(100);// sets myObj.x to 100
console.log(myObj); // Object {x: 100}
console.log(myObj.set(100)); // Object {x: 100}
```

JavaScript

this binding

strict mode causes different behaviour:

- 'use strict';
- Prevents access to global variable
- **this** undefined
- TypeError generated when code below run in strict mode

```
// Method invocation: this now bound to global object
myObj = {
  x: 0,
  set: function (x) {
    modify(x);
    function modify(val) { // nested function
      this.x = x; // this bound to global obj: undefined in strict mode
    };
  },
};
```

JavaScript

this binding

Arrow function - introduction

```
// What we're familiar with:
function add(x, y) {
  return x + y;
}

console.log(add(10, 20)); // 30
```

```
/**
 * Alternative approach: arrow function.
 * @see page 46 ES6 and Beyond (referenced)
 * @see MDN (referenced)
 */
const add2 = (x, y) => x + y;
console.log(add2(10, 20)); // 30
```

JavaScript

this binding

Pre-ES6 workaround hack

```
'use strict';
let myObj = {
  x: 0,
  set: function (x) {
    let that = this;
    modify(x);
    function modify(val) { // nested function
      that.x = x; // workaround hack
    };
  },
};

myObj.set(100); // myObj.x set to 100
```

JavaScript

this binding

Use arrow function to bind inner **this** to containing object

```
//this now bound to containing object myObj
let myObj = {
  x: 0,
  set: function (x) {
    let modify = (val) => { // nested function
      this.x = val; // this now bound to myObj
      console.log(this); // Object{x: 0}
    };

    modify(x);
  },
};

console.log(myObj); // Object{x: 0}

myObj.set(100); // myObj.x set to 100
```

JavaScript

this binding

Another JavaScript booby trap

```
// Okay: Method invocation: this bound to containing object
myObj = {
  x: 0,
  set: function (x) {
    this.x = x;
    return this;
  },
};

console.log(myObj); // Object {x: 0}
console.log(myObj.set(0)); // Object {x: 0}
```

JavaScript

this binding

Another JavaScript booby trap

```
/**
 * Not okay: Alternative approach: arrow function.
 * Method invocation: this now bound to global object
 * @see page 50 ES6 and Beyond (referenced)
 */
myObj = {
  x: 0,
  set: x => {
    this.x = x;
    return this;
  },
};

console.log(myObj); // Object {x: 0}
console.log(myObj.set(0)); // Window {...}
```

JavaScript

this binding

Constructor invocation: not recommended

```
'use strict';
function Person(name) {
  this.name = name; // this bound to Person object
}

let x = new Person('Jane');
console.log(x); // Object { name: "Jane" }
```

```
// Omitting 'use strict'
// If strict mode & new omitted then this undefined
function Person(name) {
  this.name = name; // this bound to global object
}

let x = Person('Jane'); // Oops! Forgot new keyword
console.log(x); // undefined
```

JavaScript

Passing function as function argument

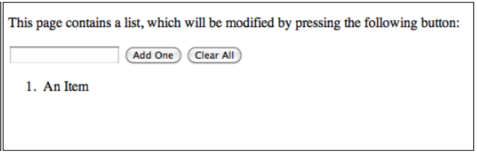
```
// Passing a named function as an argument
function myFn(fn) {
  const result = fn();
  console.log(result);
};

function myOtherFn() {
  return 'hello world';
};

// logs 'hello world'
myFn(myOtherFn);
```

Example

Button press causes invocation JavaScript function

Page contains:

- paragraph <p>
- input elements <input>
    - text field
    - buttons
- list

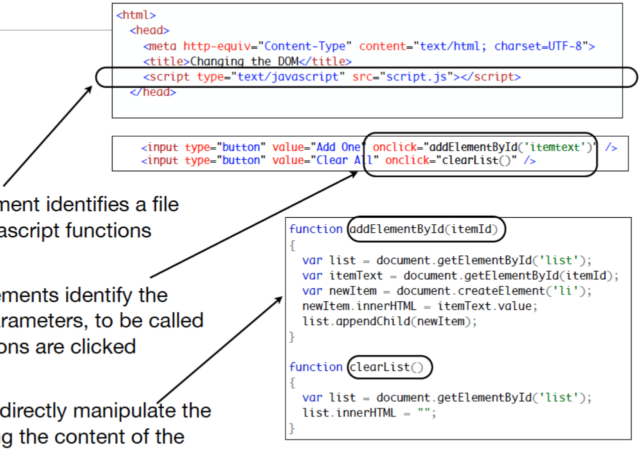

This page contains a list, which will be modified by pressing the following button:

1. An Item

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Changing the DOM</title>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
    <p>
      This page contains a list, which will be modified by pressing the following button:
    </p>
    <input type="text" id="itemtext" />
    <input type="button" value="Add One" onclick="addElementById('itemtext')" />
    <input type="button" value="Clear All" onclick="clearList()" />
    <ol id="list">
      <li> An Item </li>
    </ol>
  </body>
</html>
```

Example

JavaScript functions



```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Changing the DOM</title>
    <script type="text/javascript" src="script.js"></script>
  </head>
```

```
    <input type="button" value="Add One" onclick="addElementById('itemtext')" />
    <input type="button" value="Clear All" onclick="clearList()" />
```

```
function addElementById(itemId)
{
  var list = document.getElementById('list');
  var itemText = document.getElementById(itemId);
  var newItem = document.createElement('li');
  newItem.innerHTML = itemText.value;
  list.appendChild(newItem);
}

function clearList()
{
  var list = document.getElementById('list');
  list.innerHTML = "";
}
```

script.js

- The script element identifies a file containing javascript functions
- The button elements identify the functions + parameters, to be called when the buttons are clicked
- The functions directly manipulate the DOM, changing the content of the current page

Functions

Which to use? Function expression or function statement

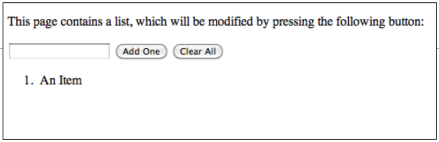
```
// Function statements: Airbnb recommendation (ES6)
function outer1() {
  hoisted(); // => foo
  function hoisted() {
    console.log('foo');
  }
}
```

```
// Function expressions: Crockford recommendation (ES5)
let outer2 = function outer2() {
  notHoisted(); // => TypeError: notHoisted is not a function
  let notHoisted = function() {
     console.log('bar');
  };
};
```

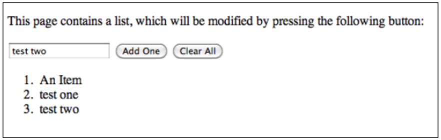
Static v Dynamic

JavaScript enabled page

Example


This page contains a list, which will be modified by pressing the following button:

1. An Item


This page contains a list, which will be modified by pressing the following button:

test two

1. An Item
2. test one
3. test two

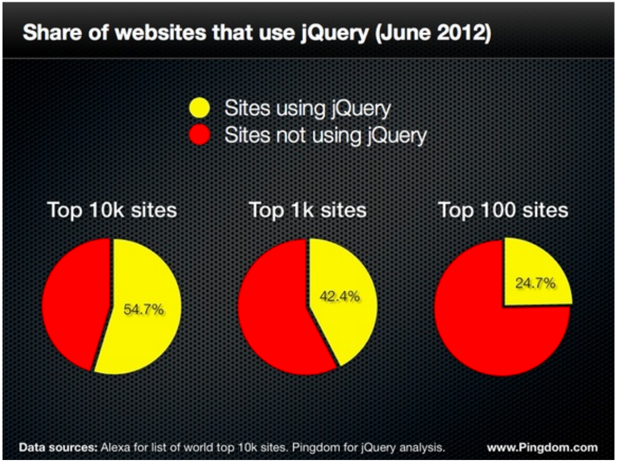
- For a static page, clicking on a link/button takes the browser to a new page (new url)
- With a dynamic page (javascript enabled), clicking on a button may change the *current* pages structure, content or style

jQuery
Introduction

jQuery JavaScript library:

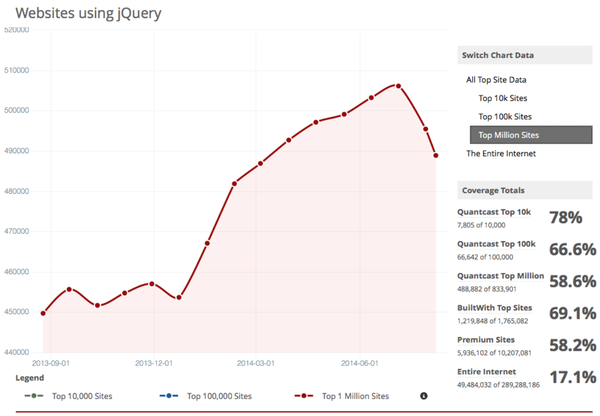
- Credible claims that most widely used
- Competitors exist:
  - Prototype
  - Modernizer
- Hides browser incompatibilities
- Facilitates finding & manipulating elements in document

jQuery
Statistics



jQuery

Statistics



Coverage Totals

Quantcast Top 10k 7,805 of 10,000	78%
Quantcast Top 100k 66,642 of 100,000	66.6%
Quantcast Top Million 488,882 of 833,901	58.6%
BuiltWith Top Sites 1,219,848 of 1,765,082	69.1%
Premium Sites 5,936,102 of 10,207,081	58.2%
Entire Internet 49,484,032 of 289,288,186	17.1%

jQuery
Features

Facilitates modifications to web page:

- Add or change specific content
- Change HTML attributes
- Change CSS properties
- Define event handlers
- Perform animation

```
<!DOCTYPE html>
<html>
<head>
  <title>JQuery</title>
</head>
<body>
  <button class="edit" onclick="
       change()">Change</button>
  <script src="jquery.js"></script>
  <script>
  function change() {
    $("button.edit").html("Next Step...");
  }
  </script>
</body>
</html>
```

jQuery
Focus

jQuery focussed on queries
Typically uses CSS selectors

- Identify set document elements
- Return object representing these
- Object has useful methods to operate on data
- Method chaining provided where possible
- Can operate on elements as group rather than individually

```
// Returns a jQuery object containing all div elements in document.
// Observe jQuery variable naming convention: $div.
let $divs = $('div');
console.log($divs);
```

jQuery

Returned jQuery object

```
<body>
  <button onclick="lotsadivs()">Press</button>
  <div id = 'div-1'>
      <div id = 'div-2'>
      </div>
  </div>
  <script src="jquery-2.2.3.min.js"></script>
  <script src="jquery.js"></script>
</body>
```

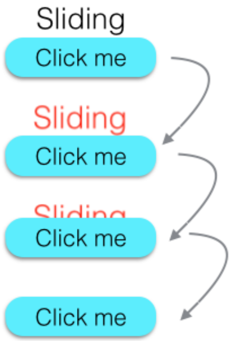
```
//File: jquery.js
function lotsadivs() {
  console.log($('div').length);
}
```

```
 // output
 [div#div-1, div#div-2, ... selector:"div"]
```

jQuery

Method chaining

```
//html
  <body>
    <p id="p1">Sliding</p>
    <button>Click me</button>

    <script src="jquery.js"></script>
    <script>
    $(function()
      {
      $('button').click(function(){
        $('#p1').css('color','red').slideUp(2000);
      });
    });
    </script>
  </body>
```



jQuery

Terminology

the jQuery function

- $ or jQuery: single global function

a jQuery object

- is object returned by $()

the selected elements

- determined by CSS selector parameter in $

a jQuery function

- a function defined within $()

a jQuery method

- bound to jQuery object

```
//these 2 methods exactly the same
function change() {
  $('button.edit').html('Next');
}

function change() {
  jQuery('button.edit').html('Next');
}
//jQuery function: invoke func for each
     element of array
$.each(array, func);

//jQuery method: invoke func2 once for
     each selected element
$('a').each(func2);
```

jQuery
How to Obtain

Method 1

- Download from *jquery.com*

Method 2

- Use a content distribution network (CDN)
    - code.jquery.com/jquery-1.4.2.min.js
    - ajax.microsoft.com/ajax/jquery/jquery-1.4.2.min.js
    - ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js

```
//include jquery script tag before other script calls at end document body
//ensure use latest versions jquery (not shown here)
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js">
```

JavaScript

Scope - Window object

Window object

- The global object
- Represents open window in browser
- Entry point client-side JavaScript
- Defines properties such as:
  - *location*: navigates to new page
  - *document*: returns DOM object

```
<!DOCTYPE html>
<html>
  <head>
    <title>Window</title>
  </head>
  <body>
    <script>
      window.location
          = "http://www.wit.ie";
    </script>
  </body>
</html>
```

JavaScript

HTML5 Elements

Element: extensive HTML5 list:

- `<html>` : root element
- `<head>` : collection metadata
- `<script>` : links to JavaScript
- `<a>` : hyperlink
- `<table>` : tabular data
- `<form>` : input for server

```
<!DOCTYPE html>
<html>
  <head>
    <title>Elements</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <!-- This is a comment -->
    <script src="f.js"></script>
  </body>
</html>
```

jQuery
Invoking $() function

jQuery(), a.k.a **$()**, invokable with 4 different parameters:

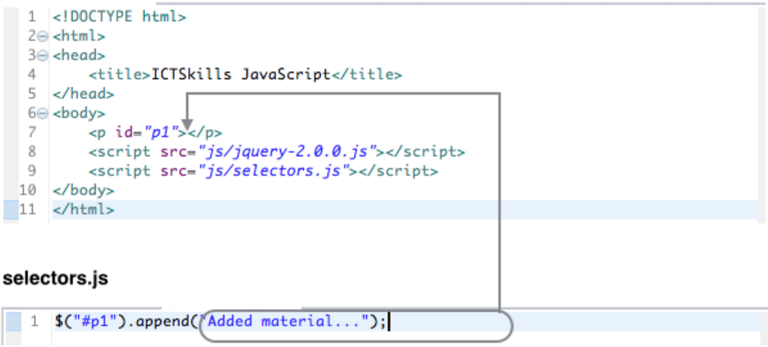
- CSS selector
- Element, Document or Window object
- String of HTML
- Function

```
//CSS selector: <p id="p1"></p>
$('#p1').append('Added material...');

//Element: <p class='p2'></p>
$('p.p2').append('Second para');

//String HTML: dynamically add node
$('<p>Third para</p>').appendTo('body');

//Function: function clickbutton param
$(function clickbutton() {
  alert('Button clicked');
});
```

jQuery

Invoking $() function

example.html

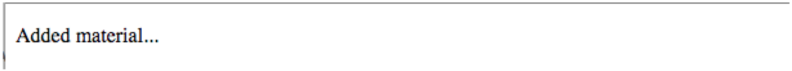


```
<!DOCTYPE html>
<html>
<head>
    <title>ICTSkills JavaScript</title>
</head>
<body>
    <p id="p1"></p>
    <script src="js/jquery-2.0.0.js"></script>
    <script src="js/selectors.js"></script>
</body>
</html>
```

selectors.js

```
$("#p1").append("Added material...");
```

example.html : output in browser



Added material...

jQuery

Immediately-Invoked Function Expression (IIFE)

```
// Click button to trigger alert
<button onclick="clickbutton()">Click</button>
<script src="jquery.js"></script>
<script>
  function clickbutton() {
    alert('Button clicked');
  }
</script>
```

```
// Example of self-invoking function
// Alert triggered on refreshing page
<script src="jquery.js"></script>
<script>
  (function clickbutton() {
    alert('Button clicked');
  }());

</script>
```

jQuery

Execute function when document loaded

onload event

- Ensure page load complete before accessing script code.

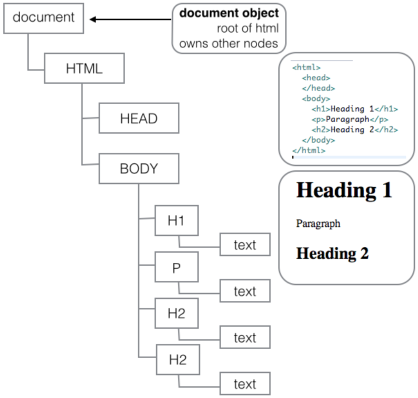
```
$(function () {
  alert('Page has loaded');
});
```

```
//Older, verbose equivalent
$(document).ready(function () {
  alert('Page has loaded');
});
```

JavaScript

Document Object Model

The **DOM**



```
<html>
  <head>
  </head>
  <body>
    <h1>Heading 1</h1>
    <p>Paragraph</p>
    <h2>Heading 2</h2>
  </body>
</html>
```

DOM

Document Object Model

The **DOM**

- A World Wide Web Consortium (W3C) Standard
- Defines standard for accessing web documents
- Represents the displayed web page
- Each element represented in the DOM by its own object
  - Access and modify individual elements
  - Add and delete elements

```
<script>
  document.getElementById('demo').innerHTML = 'Hello JavaScript!';
</script>
```

HTML DOM Document Object
HTML DOM Nodes

In the HTML DOM (Document Object Model), everything is a node:

- The document itself is a document node
- All HTML elements are element nodes
- All HTML attributes are attribute nodes
- Text inside HTML elements are text nodes
- Comments are comment nodes

DOM

The Web Browser

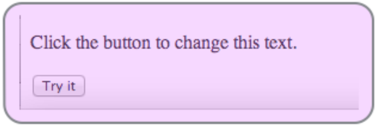
On opening HTML document in browser:

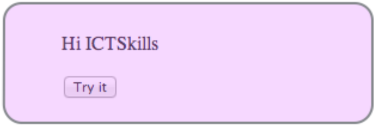
- It becomes a **document object**
    - The **document object** is root node of HTML document
    - **document object** provides properties and methods to access node objects from within JavaScript.

```
<script>
  let x = document.getElementsByName('map');
  alert(map.length);

</script>
```

DOM Access

Demo change text


Click the button to change this text.


Hi ICTSkills

DOM Access

Using DOM method

```
<!DOCTYPE html>
<html>
  <body>
    <p id="demo">Click the button to change this text.</p>
    <button onclick="domAccess()">Try it</button>
    <script src="dom.js"></script>
  </body>
</html>
```

```
//file: dom.js
function domAccess() {
    document.getElementById('demo').innerHTML = 'Hi ICTSkills';
}
```

DOM Access

Using jQuery

```
<!DOCTYPE html>
<html>
  <body>
    <p id="demo">Click the button to change this text.</p>
    <button onclick="jQueryAccess()">Try it</button>
    <script src="jquery-2.0.0.js"></script>
    <script src="jq.js"></script>
  </body>
</html>
```

```
//file: jq.js
function jQueryAccess() {
  $('#demo').html('Hi ICTSkills');
}
```

HTML Tags

Attributes **name** and **id** are not interchangeable

name: Identifies value in form data

id: Uniquely identifies an element so you can access it

```
//View (Semantic UI)
<input id="paypal" name="methodDonated" value="paypal" type="radio">
<label for="paypal">PayPal</label>
<input id="direct" name="methodDonated" value="direct" type="radio">
<label for="direct">Direct</label>
```

```
//Controller (Play): attribute name is methodDonated; content is value
public static void donate(..., String methodDonated)
```

HTML Nodes

Methods to retrieve nodes

- document.getElementById(id)
    - **id** unique on a page hence *getElementById*
- document.getElementsByName(name)
    - returns array of elements with **name** attribute = *name*
    - **name** need not be unique hence *getElementsByName*
- node.getElementsByTagName(tagName)
    - returns array of elements with **tagName** attribute = *tagName*

Get element by id

Simple demo `document.getElementById(id)`

Prints the height of image whose *id="img1"*

Native JavaScript

```
//in html file
<img src="img/01.png" id="img1">
//in javascript file
let image = document.getElementById('img1');
alert('Image height is ' + image.height);
```

jQuery

```
//in html file
<img src="img/01.png" id="img1">
//in javascript file
alert('Image height is ' + $('#img1').height());
```

Get elements by name

Simple demo `document.getElementsByName(name)`

Discovers images with attribute *name="imgs"*

Native JavaScript

```
let images = document.getElementsByName('imgs');
for (let i = 0; i < images.length; i++) {
  alert('Image height is ' + images[i].height);
}
```

jQuery

```
let $images = $('[name="imgs"]');
images.each(function () {
  alert('Image height is ' + $(this).height());
});
```

Get elements by tagName

Simple demo `node.getElementsByTagName(tagName)`

Can be used on a sub-tree, not just entire document

Native JavaScript

```
let imgDiv = document.getElementById('ictskills-images');
let images = imgDiv.getElementsByTagName('img');
for (let i = 0; i < images.length; i++) {
  alert('Image height is ' + images[i].height);
}
```

jQuery

```
//let images: only those contained in node <div id="ictskills-imgs">
//with attribute name="imgs", e.g.: <img src="img/01.png" name="imgs">
let $images = $('#ictskills-imgs [name=\'imgs\']');
images.each(function () {
  alert('Image height is ' + $(this).height());
});
```

Hide | Reveal Elements

Using Native JavaScript

HTML

```
    <p id="text">Watch me appear and disappear</p>
    <button onclick="hide()">Hide</button>
    <button onclick="reveal()">Reveal</button>
```

JavaScript

```
function hide() {
    document.getElementById('text').style.visibility = 'hidden';
}

function reveal() {
  document.getElementById('text').style.visibility = 'visible';
}
```

Hide | Reveal Elements
Using jQuery

HTML

```
<p id="text">Watch me appear and disappear</p>
<button onclick="hide()">Hide</button>
<button onclick="reveal()">Reveal</button>
```

jQuery

```
 function hide() {
  $('#text').hide();
}

function reveal() {
  $('#text').show();
}
```

Semantic UI

Enable Dropdown Box using JQuery

HTML

```
<div class="ui selection dropdown">
  <input name="amountDonated" type="hidden">
  <div class="default text">Amount</div>
  <i class="dropdown icon"></i>
  <div class="menu">
    <div class="item" data-value="100">$100</div>
    <div class="item" data-value="200">$200</div>
    <div class="item" data-value="300">$300</div>
  </div>
</div>
```

jQuery

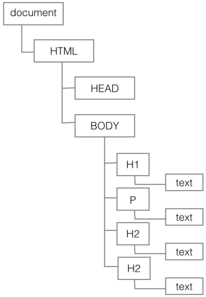
```
//Ensure you use latest version jQuery
//http://code.jquery.com/jquery-latest.min.js
<script>$('.ui.selection.dropdown').dropdown('enable');</script>
```

The DOM

Concluding with one expert's view

DOM (Document Object Model)

- "A vast source of incompatibilites, pain and misery" –Douglas Crockford



Summary

- Scripts
    - How to load - in html or external files
    - Number scripts
    - Minification
- Developer tools
    - Chrome
    - Firefox
- Functions
    - First class objects
    - May be assigned to variable
    - Passed as parameters
    - Values in objects
    - Contain other functions
    - The arrow function (ES6)

Summary (continued)

- jQuery
  - A popular JavaScript library.
  - Abstracts browser inconsistencies.
  - Community supported - continuous growth
  - jquery-2.2.4 contains almost 10,000 lines.
- Document Object Model (DOM)
  - HTML page underlying data structure.
  - Difficult development environment.
  - Better to use jQuery v native JavaScript.

References

1. Simpson Kyle (2015). You Don't Know JS: ES6 & Beyond. O'Reilly Media

http://shop.oreilly.com/product/0636920033769.do?sortby=publicationDate

[Accessed 2016-05-09]

2. MDN: Mozilla Developer Network - Arrow functions

https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Functions/Arrow_functions

[Accessed 2016-06-16]

3. W3Schools JavaScript Tutorial

http://www.w3schools.com/js/

[Accessed 2016-06-16]

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit