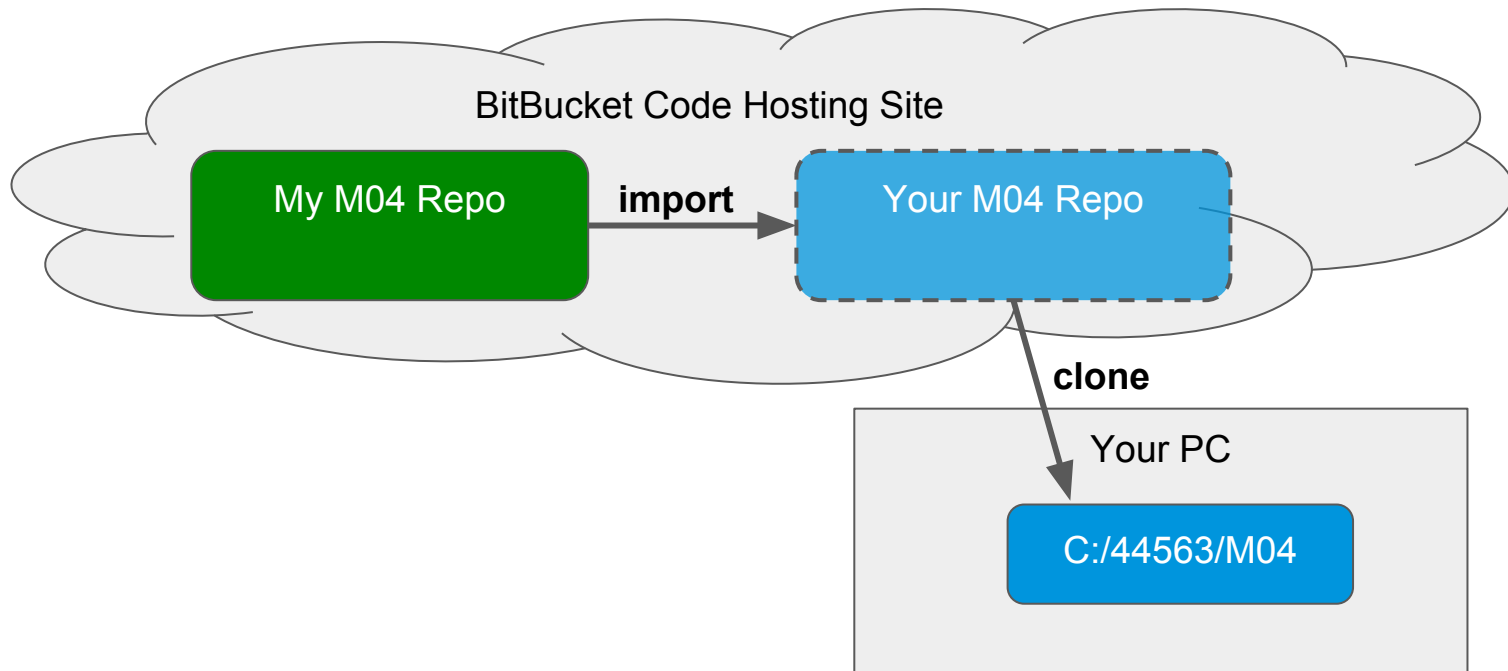


44-563: Unit 04

Developing Web Applications and Services

Today

If you arrive early, please **import** to your own cloud-based repository from <https://bitbucket.org/professorcase/m04>
Then clone your repo down to your laptop and open your m04 folder in VS Code.



Wk	Topics
1	Intro, static pages, HTML5, CSS3
2	Holiday (no class M) Responsive design
3	JavaScript
4	DOM, JQuery, Workshop 1, M04, W04, W05, A02
5	Node.js, Exam 1

Exam 1 covers first 4 weeks:
HTML / CSS / Responsive design /
BootStrap / JS / DOM / JQuery

End of this Week

1. 5-minute Reinforcement Quiz
2. **Some sections will have the first 10-minute Whirlwind Workshop**
 - a. Typically, only one Workshop Team from each section will present (if that section has a workshop that day - see your instructor for section schedules)
3. Lab time to work on A02, ask questions about practice exercises

Includes

- Manipulating HTML & CSS
- Document Object Model (DOM)
- JQuery JavaScript Library - manipulating the DOM
- Debugging
- AJAX
- **Workshop 1**

Manipulating HTML and CSS with code

Goal: Greet users based on geolocation

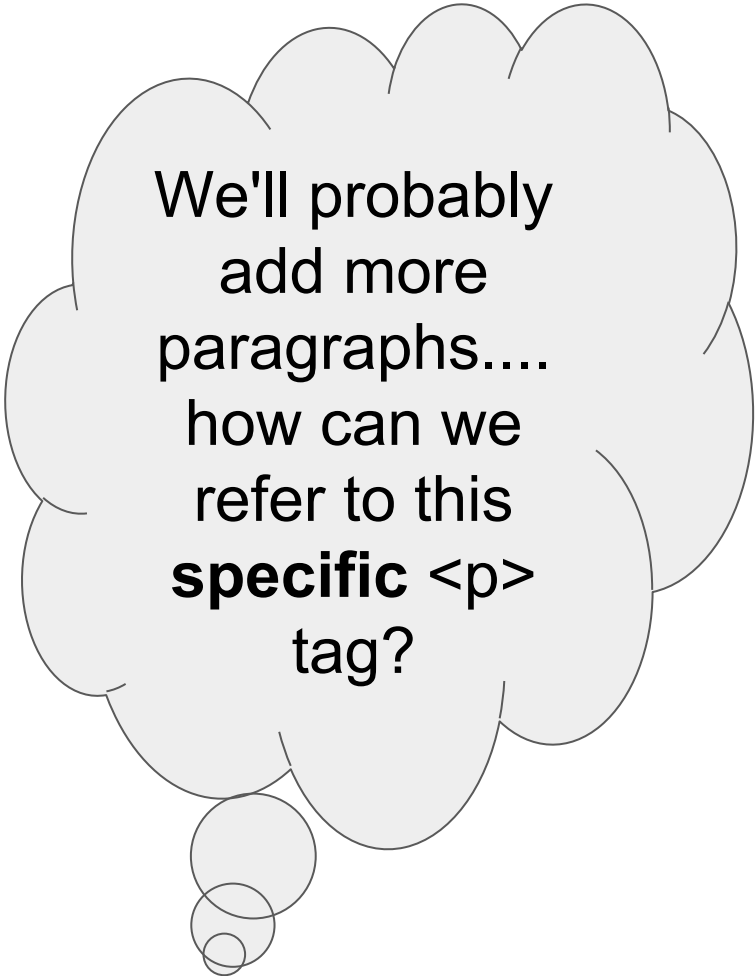
```
<html>
```

```
<body>
```

```
<p>Hello, World!</p>
```

```
</body>
```

```
</html>
```




We'll probably
add more
paragraphs....
how can we
refer to this
specific <p>
tag?

How to change HTML / CSS?

<html>

<body>



Now we can
access it - how
can we change
it?

<p id="demo">Hello, World!</p>

</body>

</html>

JS can change HTML/ CSS

```
<html>  
<body>
```

```
<p id="demo"> </p>
```



1. The inner HTML is between the opening and closing <p> tags.

```
<script>
```

```
document.getElementById("demo").innerHTML="Hola!";
```

```
</script>
```

2. Set the innerHTML with code.



```
</body>
```

```
</html>
```

JavaScript can:

- change all HTML elements
- change all HTML attributes
- change all CSS styles
- remove HTML elements & attributes
- add HTML elements & attributes
- react to existing HTML events
- create HTML events
- ~~save the world ... maybe~~

JS uses the DOM API

The **Document Object Model (DOM)** is an **API** that provides a standard way of referring to everything on our page, enabling us to manipulate it as we please.

The DOM specifies both a **data structure** to represent the elements of a document, and **methods/properties** to access/manipulate them.

While the DOM is “**platform and language-neutral**”, we will look at its methods/properties only in JavaScript.

API - Application programming interface

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Other Web APIs

Additional APIs include [1]:

- Device APIs
- Communication APIs
- Data management APIs

A good API is very, very important. The DOM is not considered the best example of an API. (We're sorry.) [2]

[1] https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

[2] [Douglas Crockford: An Inconvenient API - The Theory of the DOM](#)

DOM for DOMMIES :-)

JavaScript provides a **window** object.

We can use the **window.document** object to access the **DOM**.

Here is just a small sampling of what DOM can do:

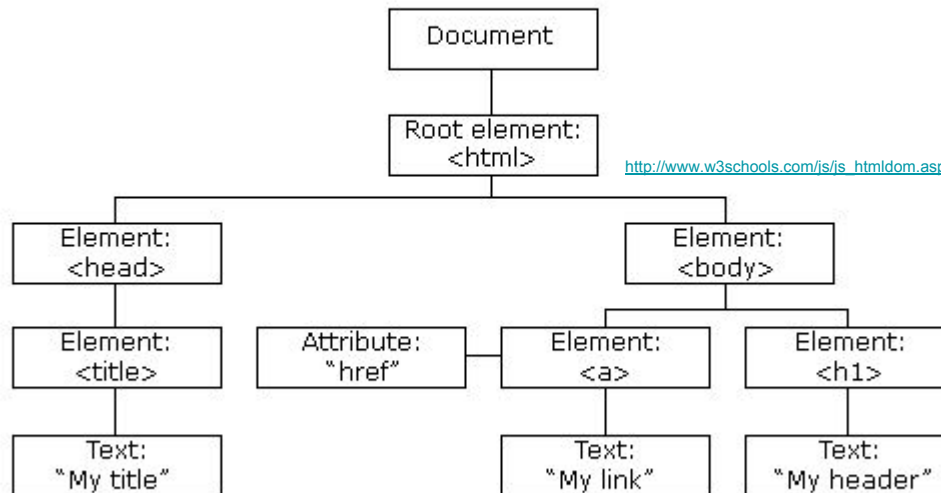
- `document.getElementById(id)` // access an element
- `document.getElementsByTagName(name)` // return an array of elements with given tag (e.g., “p”, “button”)
- `document.getElementsByClassName(name)`
- `document.createElement(name)` // create an element
- `parentNode.appendChild(node)`, `parentNode.removeChild(element)`,
`document.replaceChild(element)`
- `element.innerHTML`
- `element.style.left`
- `element.setAttribute`
- `element.getAttribute`
- `element.addEventListener`
- `window.content`
- `window.onload`
- `window.dump`
- `window.scrollTo`

Another Day, another DOM

The DOM is a tree of nodes.

Each node can be an **element**, an **attribute**, or **text**

The HTML DOM Tree of Objects



Everything's a **node**.

- **Parent** node
- **Child** node(s)
- **Sibling** node(s)

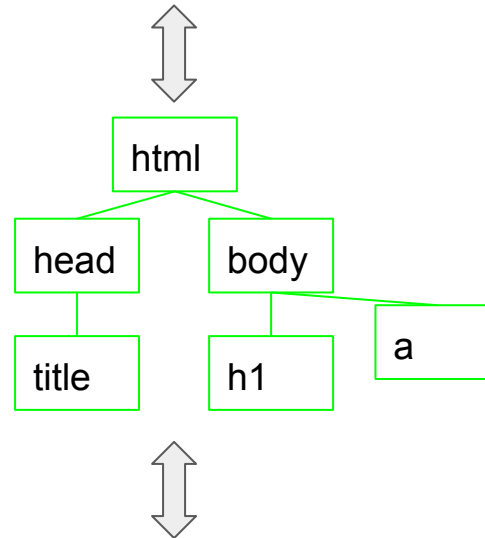
JS and the DOM, Visualized

This is a web page (OK, you knew that ☐):

```
<!DOCTYPE HTML>
<html>
<head>
<title>Hey, there</title>
</head>
<body>
<h1 id="hi">Can you hear me?</h1>
<a href="cbc.ca">If not, try going to the CBC</a>
</body>
</html>
```

This is the DOM's representation* of that page, as a tree: each node is an HTML element.

**slightly simplified*



There are *myriad* properties and methods to manipulate the DOM: here we see a couple in action.

```
var msg = document.getElementById("hi")
msg.innerHTML = "hello world"
```

To explore the DOM for yourself, check out the [Live DOM Viewer](#)

Techy Aside: Lang.-Neutrality

The methods and properties that we use to access the DOM tree are all expressed in JavaScript, but the DOM is supposed to be platform/language neutral

To achieve the latter, the W3C docs use Web IDL (Web Interface Definition Language). It specifies the name of the function, the types of each argument, and the return type of the function in a non-language-specific way.

Web IDL

4.2.2 Interface `NonElementParentNode`

Note: The `getElementById()` method is not on `elements` for con

```
IDL
[NoInterfaceObject,
 Exposed=Window]
interface NonElementParentNode {
    Element? getElementById(DOMString elementId);
};
Document implements NonElementParentNode;
DocumentFragment implements NonElementParentNode;
```

JavaScript

Syntax

```
1 | element = document.getElementById(id);
```

Parameters

`id`

is a case-sensitive string representing the unique ID of the element being sought.

Return Value

`element`

is a reference to an `Element` object, or null if an element with the specified ID is not in the document.

Building a DOM from Code

```
<!DOCTYPE html>
```

HTML

```
<html>
```

```
<head>
```

```
  <title>||Working with elements||</title>
```

```
</head>
```

```
<body>
```

```
  <div id="div1">The text above has been  
created dynamically.</div>
```

```
</body>
```

```
</html>
```

```
document.body.onload = addElement;
```

JS

```
function addElement () {
```

```
  // create a new div element
```

```
  // and give it some content
```

```
  var newDiv = document.createElement("div");
```

```
  var newContent = document.createTextNode("Hi there and greetings!");
```

```
  //add the text node to the newly created div.
```

```
  newDiv.appendChild(newContent);
```

```
  // add the newly created element and its content into the DOM
```

```
  var currentDiv = document.getElementById("div1");
```

```
  document.body.insertBefore(newDiv, currentDiv);
```

```
}
```

Result Displayed

Hi there and greetings!

The text above has been created dynamically.

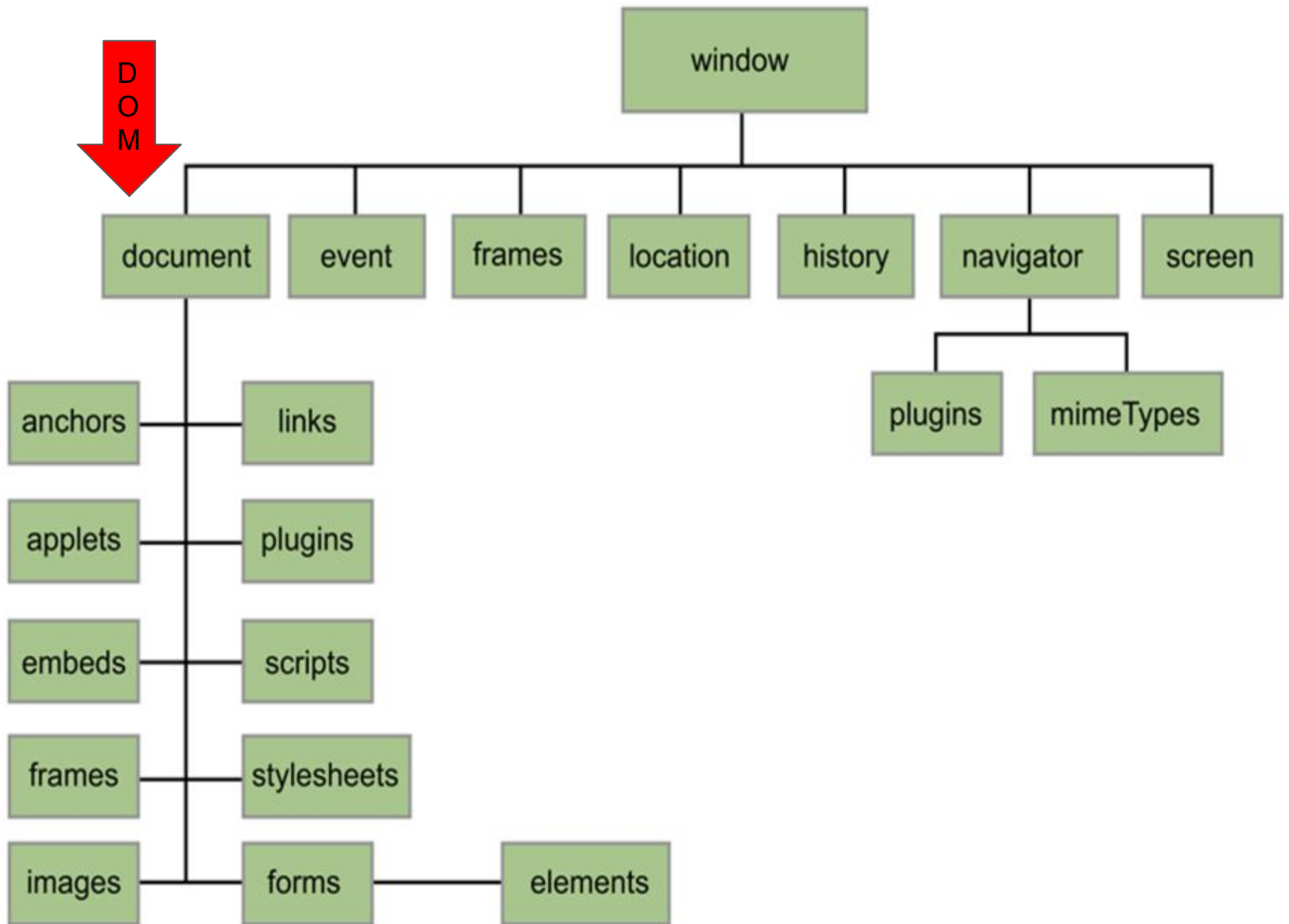
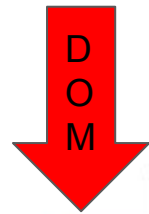
Beyond the DOM API

JavaScript provides a **window** object; we use **window.document** to access the DOM.

There are other objects in window as well, e.g.:

- **navigator** (browser)
- **screen**
- **events**
- **history**
- **localStorage**
- and more.....





Using DOM by id or tagname

document.**getElementById**('id');

*Gets one element with this **id** as an object*

document.**getElementsByTagName**('tagname');

*Gets all elements with this **tagname** as an array list*

Using the DOM as tree nodes

`node.getAttribute('attribute')`: Retrieves the value of the attribute with the name `attribute`, or null if the attribute does not exist

`node.setAttribute('attribute', 'value')`: Sets the value of the attribute with the name `attribute` to value

`node.nodeType`: Reads the type of the node (1 = element, 3 = text node)

`node.nodeName`: Reads the name of the node (either element name or `#textNode`)

`node.nodeValue`: Reads or sets the value of the node (the text content in the case of text nodes)

DOM tree, nodes & navigation

node.previousSibling: Retrieves the previous sibling node and stores it as an object.

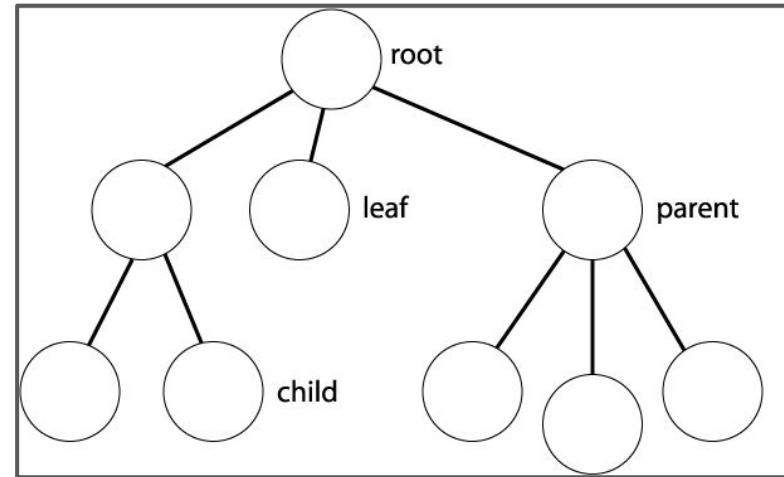
node.nextSibling: Retrieves the next sibling node and stores it as an object.

node.childNodes: Retrieves all child nodes of the object and stores them in an list.

node.firstChild: returns first child as a node object

node.lastChild: returns last child as a node object

node.parentNode: Retrieves the parentNode that contains node.



<https://i.stack.imgur.com/5kJXf.gif>

<https://www.christianheilmann.com/stuff/JavaScript-DOM-Cheatsheet.pdf>

Creating New Nodes

`document.createElement(element)`: Creates new element node named *element*.

`document.createTextNode(string)`: Creates a new text node with value *string*.

`newNode = node.cloneNode(bool)`: Creates *newNode* as a copy of *node*. If *true*, includes clones of all child nodes.

`node.appendChild(newNode)`: Adds *newNode* as a new (last) child node to *node*.

`node.insertBefore(newNode,oldNode)`: Inserts *newNode* as a new child node of *node* before *oldNode*.

`node.removeChild(oldNode)`: Removes the child *oldNode* from *node*.

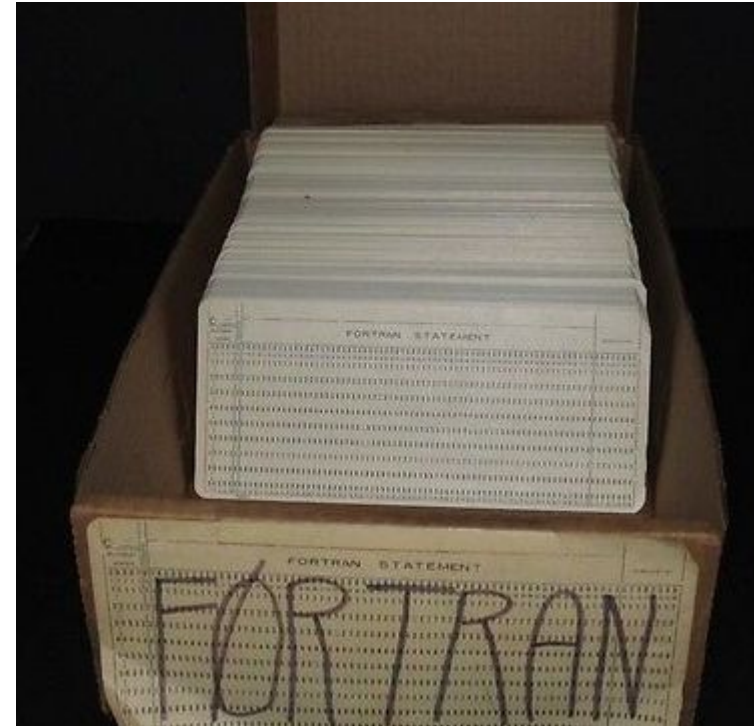
`node.replaceChild(newNode, oldNode)`: Replaces the child node *oldNode* of *node* with *newNode*.

`element.innerHTML`: Reads or writes the HTML content of the given element as a string—including all child nodes with their attributes and text content.

Event-driven programming

Old FORTRAN programs were a type of "**procedural programming**". You loaded your stack of cards, the program started at the beginning and proceeded to the end.

Web apps use "**event-driven programming**". You load your app and... nothing happens. After the load procedures, it does nothing until an event occurs (e.g. a timer ticks, a user clicks). The browser listens for events. When an event is detected, if there is a function specified to handle the event, that event handler function is called.



DOM Events

JS can be executed when an event occurs. Just add an **event attribute** to any HTML element (see [MDN](#)) or use **addEventListener()**

- **Mouse** events: click, dblclick, mouseenter/leave, mouseover/out (includes children), mousemove, mousedown/up, contextmenu
- **Network** events: online, offline
- **Form** events: reset, submit
- **Focus**: focus, blur
- **Drag & Drop**: drag, dragstart/end, dragenter/leave, drop

Be able to recognize these common events and know how they could be used.

DOM: Adding an Event **Attribute**

```
<button onclick="showTime()">The time at the tone is ... </button>
```

```
<p onmouseenter="welcome()">My name is Inigo Montoya!</p>
```

How many HTML elements are shown?

What is the first element type?

How many attributes does the first element have?

What is the attribute? What is its value?

DOM EventListener

JS can attach an event handler to an element.

Use the **document** object to access one or more **elements**, and call **addEventListener** by passing in the **event string** and the **function**.

```
element.addEventListener("mouseover", myFunction);  
element.addEventListener("click", mySecondFunction);  
element.addEventListener("mouseout", myThirdFunction);  
element.addEventListener("click", function(){ add(p1, p2); });  
element.removeEventListener("mousemove", myFunction);
```

*Which example passes an **anonymous** function directly rather than calling a named function?*

AddEventListener

```
<button id="myBtn">Try it</button>
```

```
<p id="demo"></p>
```

```
<script>
```

```
var btn = document.getElementById("myBtn");  
btn.addEventListener("mouseover", showOver);  
btn.addEventListener("click", showClick);  
btn.addEventListener("mouseout", showOut);
```

```
function showOver() {  
    document.getElementById("demo").innerHTML += "Moused over!<br>";  
}
```

```
function showClick() {  
    document.getElementById("demo").innerHTML += "Clicked!<br>";  
}
```

```
function showOut() {  
    document.getElementById("demo").innerHTML += "Moused out!<br>";  
}
```

```
</script>
```

*Arg 1: pass in event as a **string***

*Arg 2: pass in function by its **name** without () or tick marks*

A Parenthetical Observation

Just to underline the fact, **parentheses are used when specifying a callback as HTML attribute**, but not when used with `addEventListener()` in JavaScript:

```
<p id="me" onclick="displayEmoji()">Click me to be reminded of a bad movie</p>
```



string (function name with parens)

```
var cuteBtn = document.getElementById("me")
```

```
btn.addEventListener("click", displayEmoji)
```



string



value (no parens)



JQuery

<https://jquery.com/>

jQuery

jQuery is a fast, small, and feature-rich JavaScript **library**.

It focuses on:

- HTML document traversal and manipulation
- CSS manipulation
- event handling
- Animation
- AJAX

Easy-to-use API that works across a multitude of browsers.

Access dynamic HTML (JS, HTML, CSS, DOM)

All web developers should be familiar with HTML, CSS, JS, and jQuery.

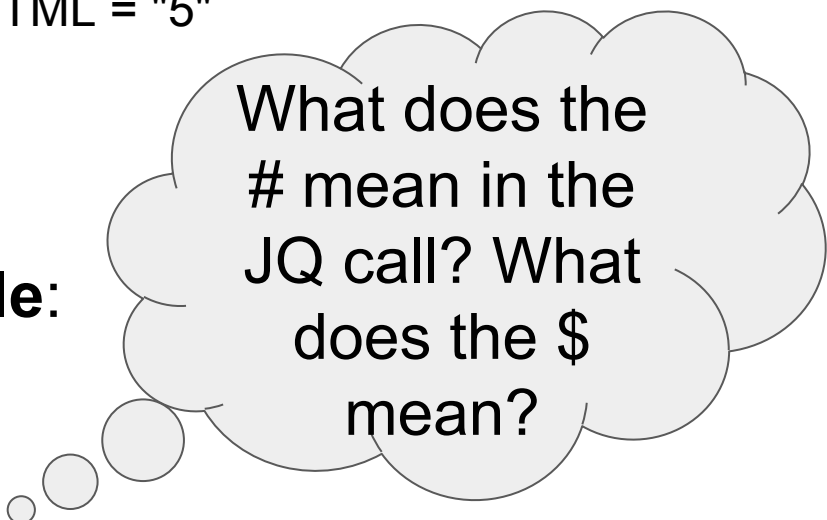
JQuery benefits

With JavaScript we can access/modify elements on a web page with calls such as:

- `document.getElementById("width").innerHTML = "5"`
- `document.getElementsByName("length")`
- `document.getElementsByTagName("p")`

jQuery is more **concise** and **flexible**:

- `$("#width").html(); // gets`
- `$("#width").html(5); // sets`

A light gray thought bubble with a black outline, containing text. It has three smaller circles leading to it from the bottom left.

What does the # mean in the JQ call? What does the \$ mean?

jQuery works well with JS's `parseFloat()` to return a number:

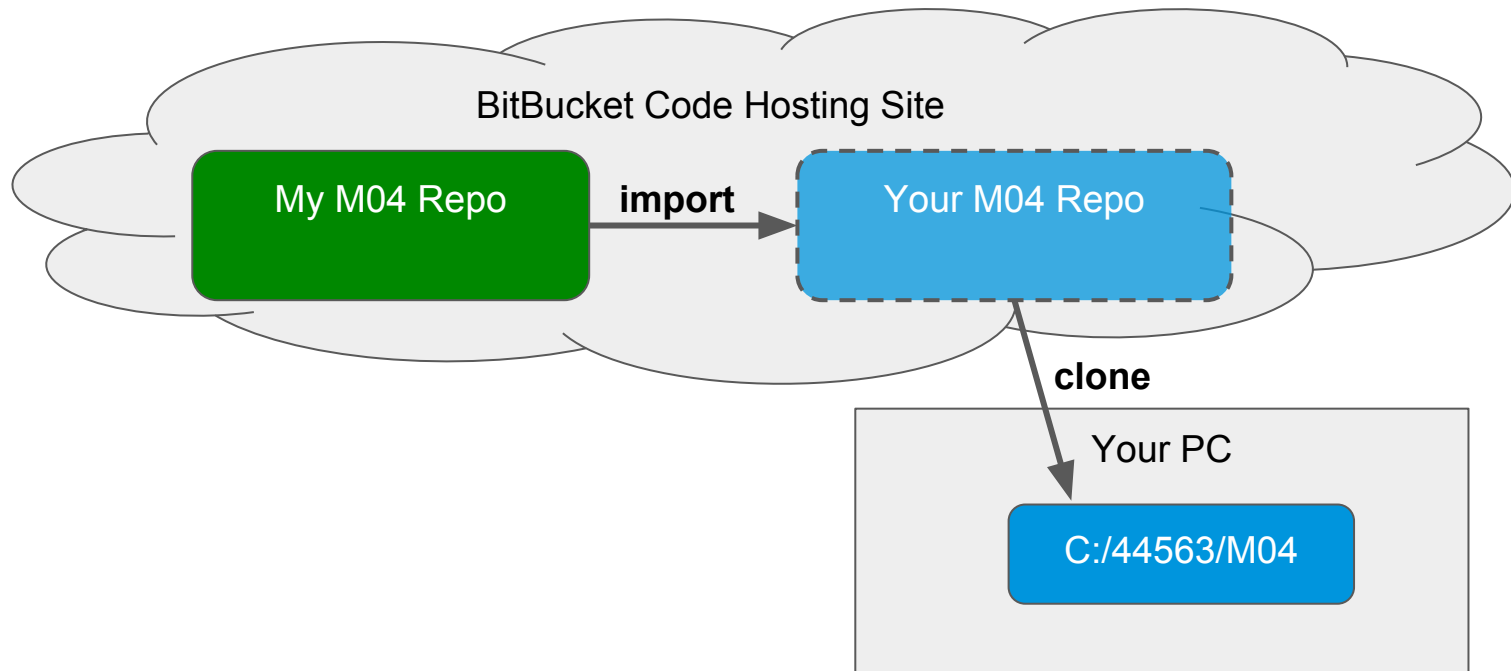
- `let inputWidth = parseFloat($('#width').html());`

Let's Try It (M04)

Go to your BitBucket profile. **Import** the repo from

<https://bitbucket.org/professorcase/m04>

Clone your repo down to your laptop and open your m04 folder in VS Code.



M04

Look it over. What **files** are there? Is there a **unit test**?

Which file would have links to CSS and scripts that point to JavaScript?

Open this **HTML** file. What frameworks and libraries are included? Is Bootstrap? Is JQuery? How can you tell?

What **file** has our JavaScript code? Open that file. Can you see any jQuery calls? How can you tell?

Run the app.

Click the **Explore the Code** button. Learn how to watch values in the built-in debugger.

M04

What do we do for this assignment?

Where will those changes take place?

Can you find three calls to **document.getElementById**?

How do we change those to more concise and flexible jQuery calls?

Wednesday: Continuing with jQuery

jQuery: Basic Syntax

`$(selector).action()`

- A \$ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples

- `$(this).hide()` - hides the current element.
- `$("p").hide()` - hides all `<p>` elements.
- `$(".test").hide()` - hides all elements with `class="test"`.
- `$("#test").hide()` - hides the element with `id="test"`.

jQuery CDN

To use jQuery in a web page, we must include a link to the jQuery library

Latest versions at: <https://code.jquery.com/>

```
<script  
src="https://code.jquery.com/jquery-3.2.1.min.js">  
</script>
```

The .min version, optimized for production/fast download, is hard to read, sort of like this font size 🐛. For students interested in looking at a more legible version, with comments, line spacing -- truly a marvel to behold -- click [here](#) to see the jquery-3.2.1.js version.

Adding JQuery to HTML

When adding libraries to your HTML, order Matters!

- Add jquery **AFTER** CSS
- Add jquery **BEFORE** other <script>s.

jQuery benefits

Animation and visual effects

- `setTimeout()` and `setInterval()` are often used to call functions that work with expandable menus and sliding menus
- jQuery simplifies animation and visual effects with methods such as
 - `animate()` for animation effect on CSS properties,
 - `fadeIn()`, `fadeOut()`, `slideUp()`, `slideDown()`, for VFX
- jQuery simplifies form validation with form-specific methods such as `.select()` or `.change()`

JQuery benefits cont.

Events and Ajax

- jQuery also has pre-written methods for most common types of events
 - click event is handled by click()
 - dblclick event is handled by dblclick()
 - [Etc.](#)
- Each of these methods takes a single argument, a function to be executed when the event occurs.
- jQuery makes Ajax calls easier, too

Run Me First

To execute a function automatically when the DOM is ready, jQuery uses this syntax

`$(handler)`

handler is a 0-parameter, possibly anon. function

E.g.,

`$(function(){alert("Ready to rock and roll")})`

jQuery: ready()

page can't be manipulated safely until it is "ready," i.e., when the DOM is completely loaded.

jQuery detects this state of readiness in two ways:

Code inside `$(document).ready()` will *only run when the page DOM is ready* for JS to execute.

See:

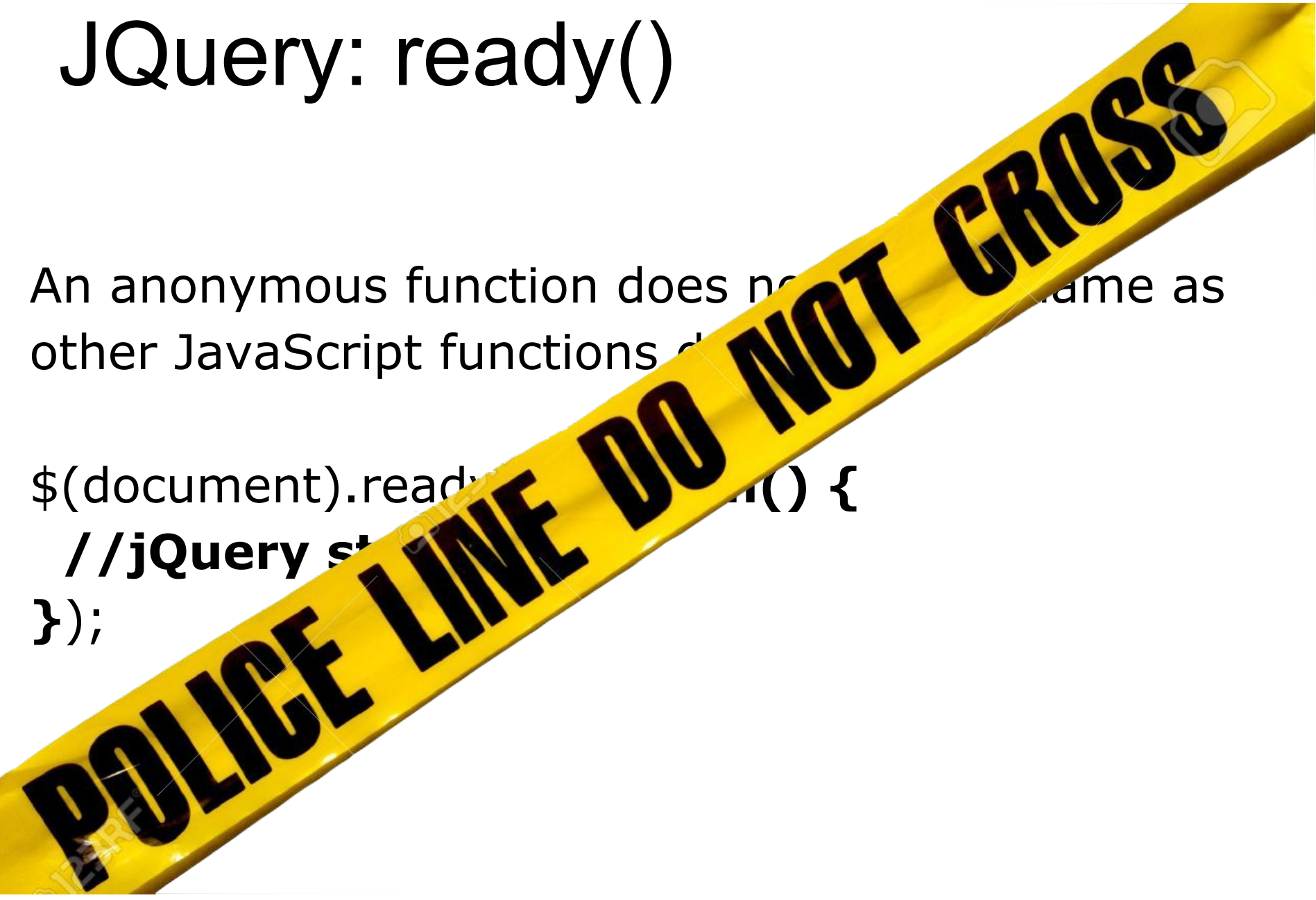
<http://api.jquery.com/using-jquery-core/document-ready>

`document.ready` has been **deprecated** (no longer in use). v3 schools may be a bit behind.

JQuery: ready()

An anonymous function does not have a name as other JavaScript functions do.

```
$(document).ready(function() {  
    //jQuery stuff  
});
```



document.ready() is deprecated

`$(document).ready(handler)` has been deprecated. The preferred way to execute code after the DOM has loaded, is easier:

`$(handler)`

So for example, do not use `ready`, use the `$()` instead.

```
$(document).ready(function(){  
    $('#left').click(  
    })
```

NO

```
$(function(){  
    $('#left').click(ghostIt)  
})
```

YES

JQuery uses CSS Selectors

Selector	Selects
All ("*")	All elements
Class (".class")	All elements that use a specified CSS class
Element ("element")	All elements that match the specified element
ID ("#id")	The first element that matches the specified id attribute
Multiple ("selector1, selector2, ...")	All elements that match the results of one or more specified selectors

JQuery CSS Selectors

- jQuery does not have a selector equivalent to `getElementsByName()` method
- jQuery does have selectors for others

JavaScript Method	jQuery Selector
<code>getElementById()</code>	<code>\$("#id")</code>
<code>getElementsByTagName()</code>	<code>\$("element")</code>

Techy Aside: What's with the \$?

The \$ is an alias for jQuery(), so that when you write:

```
$( function(){  
    $('#left').click(ghostIt)  
} )  
function ghostIt(){  
    $('#middle').animate({left: '+=50', opacity: 0.25},500);  
    $('#middle').animate({left: '-=50', opacity: 1.00},500);  
}
```

It is as if you had written `jquery(function(){ ... })`, and

```
jQuery( '#left' ).click(ghostIt())
```


A Little More jQuery

Download the ALittleMorejQuery repo:

git clone <https://mprogers@bitbucket.org/mprogers/alittlemorejquery.git>

Read the index.html and main.js files (the rest are not critical). This project was made using initializr.com, incidentally

Perform the exercises described in the comments.

jQuery at a Glance

<https://oscarotero.com/jquery>

[illegible]

Very handy reference

- `$(selector).action()`

There is tremendous value in simply reading over & being familiar with available libraries. You can produce much more, faster, better, by just **being aware** of what is available for free.

Button, Button, Who's Got the Button

```
$( function(){
    alert("Welcome to Priyanka and Poppy's Pizza Place")

    document.getElementById("addToppingsBTN").addEventListener("
click",addToppings);
    $("#cancelBTN").click(cancelPizza)

} )

function orderPizza(){
    alert("The pizza is on its way!")
}

function addToppings(){
    alert("Alright, we'll add some green peppers & onions")
}

function cancelPizza(){
    alert("Never mind ...")
}
```

```
<!doctype html>
<html class="no-js" lang="">
  <head>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" >
  </head>
  <body>
    <div class="container">
      <h1>Priyanka and Poppy's Pizza Hut</h1>
      <p>This snippet shows 3 related ways to associate a button with an event</p>
      <ol>
        <li>Use onclick= directly in HTML - not recommended</li>
        <li>Use document.getElementById().addEventListener</li>
        <li>Use $("#whatever").click</li>
      </ol>
      <button onclick="orderPizza()">Order Pizza</button>
      <button id="addToppingsBTN">Add Toppings</button>
      <button id="cancelBTN">Cancel Order</button>

      <script
src="https://code.jquery.com/jquery-3.1.1.min.js"
integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGMiJLv2b8="
crossorigin="anonymous"></script>
      <script src="main.js"></script>
      <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" </script>
    </div>
  </body>
</html>
```

Priyanka and Poppy's Pizza Hut

This snippet shows 3 related ways to associate a button with an event

1. Use onclick= directly in HTML - not recommended
2. Use document.getElementById().addEventListener
3. Use \$("#whatever").click()

Order Pizza

Add Toppings

Cancel Order

AJAX

(just a preview)

AJAX

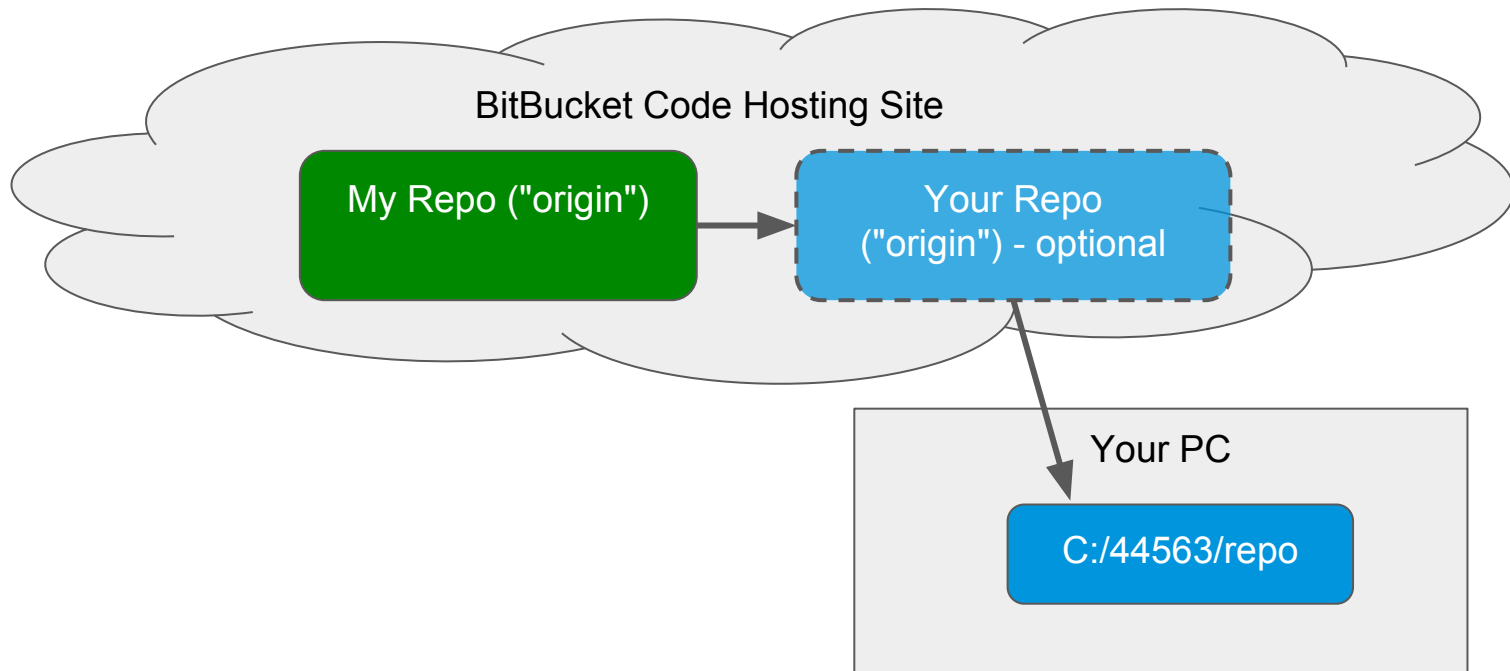
- Originally: **A**synchronous **J**avaScript **A**nd **X**ML but NOT ANYMORE
 - Now - it might not be Async
 - Now - it might not be XML
- **Combination of technologies** that allow pages on a client to exchange data with a server
- Uses HTTP requests

AJAX

- jQuery simplifies AJAX development with a client-side scripting language that acts as a proxy (intermediate server that handles requests from clients) for JavaScript

Let's Try It (W04)

Go to your BitBucket profile. Import the repo from <https://bitbucket.org/professorcase/w04> (lower case w)
Clone your repo down to your laptop and open your w04 folder in VS Code.



W04

Look it over. What do we have to do?

Notice we're doing a bit more with the Bootstrap navbar menu. There's a link to two new pages: About and Contact.

Vary the size of your browser window. What happens to the menu?

To implement the assignment, which functions should change? Which file will we be working in?

W04

Should we still have HTML elements for length and width?
What should they change to? Which file will we be working in?

Will you make any CSS changes?

W05

Preparing for the Exam in Week 5.

Due Sunday evening.

End of Week

1. 5-minute Reinforcement **Quiz**
2. **Our first 10-minute Whirlwind Workshop**
 - a. Only Workshop Team 1 from each section will present
 - b. Check the link below for Workshop (WS) Teams by Section
 - c. Check the link below for WS topics (this week is Node.js)
3. Lab time to work on A02, ask questions about practice exercises