Quick Guide to Web Development

Topics include:

Jay Wang
Summer 2012

1. REST

Designed around few key principles:

1. Use HTTP methods
2. Stateless and Cacheable
3. Use Addressable Resources
4. Support the transfer of Representations

(1.) HTTP Methods are a map to CRUD operations

CRUD = Create Read Update Delete (which is used for Actions in REST)
HTTP = Post Get Put Delete
SQL = Insert Select Update Delete

** note how all are using variations of the same idea of CRUD*

(2.) Stateless and Cacheable

- stateless means that no state is stored on the sever
- every HTTP request executes in complete isolation on the server
- easier to scale because of GET method

(3.) Resources

- Any THING - person, concept, artifact - is a resource
- Anything you can point to
- Explicit request and response - no state

(4.) Representations

- A serializable description of a Resource
  (in Rails, a Resource would be an Object)
- Resources are also modifiable through Representations
- ex. XML, JSON, binary formats (jpg, gij, mpeg)

<u>Pros of REST</u>:

- Cacheable
- Scalable
- Different Representations
- Human Readable Results
- Lightweight

<u>Cons of REST</u>:

- Must be HTTP
- No atomic transactions
- No standards for security other than HTTPs
- No standardized discovery

Ultimately, think of the REST API as another data source:

- Serialize the representation
- Format the URI into parameters, variables

- Encode authentication into the header
- Make calls
- Deserialize the responses

2. Cascading Style Sheets (CSS)

HTML is intended to define the content of a document - not to format it
CSS is used for formatting

<u>CSS Rule: contains two main parts → a selector and one or more declarations</u>

Ex: h1 {color: blue; font-size: 12px;}
- h1 is the selector and it is the HTML element you want to style
- Declarations consist of a property and a value
    - "color:blue" and "font-size:12px" are declarations

**note color is a property, blue is a value*

<u>CSS Rule: commenting in CSS is /*   */</u>

Just as how you can set styles for HTML elements, CSS allows you to specify your own selectors called id and class
- The id selector is used to specify style for a single, unique element
- The id selector uses the id attribute of the HTML element, and is defined with a "#"

Ex: The style below will be applied to the element with id = "para1"
```
#para1 {
        text-align:center;
        color:red;
}
```

The class selector is used to specify a style for a group of elements. Unlike an id selector, class selectors are used for several elements.
- The class selector is defined with a "."

Ex: .center {text-align:center;}

You can also specify that only specific HTML elements should be affected by a class. Here, there can be many elements with the class = "center", but only the p elements will be affected.

Ex: p.center {text-align:center;}

There are three ways to insert CSS:
1. External style sheet
2. Internal style sheet
3. Inline style

IMPT especially when referencing external style sheets such as bootstrap:
<u>MULTIPLE styles will cascade into one</u>

From most priority to least priority:

Inline style (inside an HTML element)
Internal style sheet (in the head section)
External style sheet
Browser Default

Background color actually refers to the background color of an element.

To set the background color of a page, you need to use the body selector.

Ex: body {background-color:#b0c4de;}

You can also use an image as a background for an element.

By default, the image will be repeated until it covers the entire element.

Ex: body {background-image:url('paper.gif');}

When dealing with images, oftentimes you need to repeat it horizontally or vertically.

Ex: background-image:url('gradient2.png');
Ex: background-repeat:repeat-x;   //x refers to horizontally, y refers to vertically

Text Color is specified by HEX, RGB, or color name.

Ex: body {color:blue;}
Ex: h1 {color:#00ff00;}
Ex: h2 {color:rgb(255,0,0);}

Text Alignment in CSS:

Text-align property is used to set the horizontal alignment of a text.
Text can be centered, or aligned to the left or right, or justified.

**note - justify means that each line is stretched so that every line has equal width and the left and right margins are straight*

Ex: h1 {text-align:center;}
Ex: p.date {text-align:right;}
Ex: p.main {text-align:justify;}

Text Decoration in CSS:

This is used to remove or set decorations from text.
It is mostly used to remove underlines from links for design purposes.

Ex: a {text-decoration:none;}
Ex: h1 {text-decoration:overline;}
Ex: h2 {text-decoration:line-through;}
Ex: h3 {text-decoration:underline;}
Ex: h4 {text-decoration:blink;}

Text Transformation in CSS:

This is used to specify uppercase and lowercase letters in a text.

Ex: p.uppercase {text-transform:uppercase;}
Ex: p.lowercase {text-transform:lowercase;}
Ex: p.capitalize {text-transform:capitalize;}

Text Indentation in CSS:

This property is used to specify the indentation of the first line of a text.
The cool thing is that you can specify how much indentation you want
Ex: p {text-indent:50px;}

Fonts in CSS:

There are two types of font family names in CSS.
Generic family - a group of font families with a similar look
Font family - a specific font family

Ex: Serif is a generic family and the font family is Times New Roman. Sans-serif is a generic family and Arial is the font family.

The font-family property should hold several font names as a fallback system so that if the browser does not support the first font, it tries the next font.
Ex: p{font-family:"Times New Roman", Times, serif;}

The font style property is mostly used to specify italic text.
There are only three types of font styles:

- Normal
  - Ex: p.normal {font-style:normal;}
- Italic
  - Ex: p.italic {font-style:italic;}
- Oblique
  - Ex: p.oblique {font-style:oblique;}

The font-size property sets the size of a text. The font size value can be absolute or relative.

Here is the normal way to set font-size with pixels.
Ex: h1 {font-size:40px;}
Ex: h2 {font-size:30px;}
Ex: p {font-size:14px;}

Sometimes you will see people setting font size using "em," which is what the Bootstrap does. "em" size unit is equal to the current font size. Note that the default text size in browsers is 16px. Thus, default size of 1em is 16px.
Ex: h1 {font-size:2.5em;} /* 40px/16=2.5em */
Ex: h2 {font-size:1.875em;} /* 30px/16=1.875em */
Ex: p {font-size:0.875em;} /* 14px/16=0.875em */

What the real pros do is set the proportions they want between different h1, h2, and p tags and then they set the overall font-size using a percentage in the body.

```
Ex: body {font-size:100%;}
Ex: h1 {font-size:2.5em;}
Ex: h2 {font-size:1.875em;}
Ex: p {font-size:0.875em;}
```

Styling links in CSS:

Links can be styled with any CSS property (color, font-family, background)

There are 4 link states:

a:link - a normal, unvisited link
a:visited - a link the user has visited
a:hover - a link where the user mouses over it
a:active - a link the moment is clicked

```
Ex: a:link {color:#FF0000;}      /* unvisited link */
Ex: a:visited {color:#00FF00;}  /* visited link */
Ex: a:hover {color:#FF00FF;}  /* mouse over link */
Ex: a:active {color:#0000FF;}
```

**note, it is more common to style links by using text-decoration: none*

CSS Rule: There are ordering rules when it comes to styling links:

a:hover must come after a:link and a:visited
a:active must come after a:hover

Styling Tables in CSS:

Table Border Property: This example shows a black border for table, th, and td elements

Ex:

```
table, th, td {
        border: 1px solid black;
}
```

For further reference, please visit http://w3schools.com/css/default.asp

3. HTML Document Object Model (DOM)

DOM stands for Document Object Model and it presents an HTML document in a tree-structure. DOM is a W3C standard, meaning it defines a standard for accessing documents such as HTML and XML.

HTML DOM defines the objects and properties of all HTML elements and the methods (interface) to access them.

- It is the standard for how to get, change, add, or delete HTML elements.
- It is used heavily in JavaScript/jQuery in calling the elements to impose the actions on.

In the DOM, everything in an HTML document is a node:

- The entire document is a document node
- Every HTML element is an element node
- The text in the HTML elements are text nodes
- Every HTML attribute is an attribute node
- Comments are comment nodes

DOM example:

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

The root node in the above HTML is <html>

All the other nodes in the document are contained in <html>
The <html> node has two child nodes → <head> and <body>
<head> node holds a child <title> node
<body> node holds two children nodes <h1> and <p>

**note – text is always stored in text nodes*

A common error in DOM processing is to expect an element node to contain text
However, the text of an element node is stored in a text node
<title>DOM Tutorial</title>

The element node <title> holds a text node with the value “DOM tutorial”
The <title> node itself doesn’t actually hold the value of “DOM tutorial”

DOM Rule: the value of the text node can be accessed by the innerHTML property

The HTML DOM views an HTML document as a node-tree.

All the nodes in the tree have relationships to each other.
All nodes are to be accessed through the tree.
Their contents can be modified or deleted and new elements can be created.

The tree starts at the root node and branches out to the text nodes at the lowest level of the tree.
Root node = <html>
Text node = "My link"

DOM Rule: Nodes in the node tree have a hierarchical relationship to each other

There are some properties and methods that define the programming interface of the HTML DOM.
In DOM, HTML docs consists of sets of node objects and the nodes can be accessed with JavaScript or other programming languages.

Properties - referred to as something that is (i.e. the name of a node)
Methods - referred to as something that is done (i.e. remove a node)

Here are some DOM properties:
x.innerHTML - the text value of x
x.nodeName - the name of x
x.nodeValue - the value of x
x.parentNode - the parent node of x
x.childNodes - the child nodes of x
x.attributes - the attributes nodes of x
**note: In the list above, x is a node object (HTML element).*

Some DOM methods:
x.getElementById(id) - get the element with a specified id
x.getElementsByTagName(name) - get all elements with a specified tag name
x.appendChild(node) - insert a child node to x
x.removeChild(node) - remove a child node from x
**note: In the list above, x is a node object (HTML element).*

The easiest way to get or modify the content of an element is by using the innerHTML property. This property is useful for returning or replacing the content of HTML elements (including <html> and <body>)

Ex:
```
<p id="intro">Hello World!</p>
<script type="text/javascript">
txt=document.getElementById("intro").innerHTML;
document.write("<p>The text from the intro paragraph: " + txt + "</p>");
</script>
```

The DOM provides the infrastructure to do a lot of the JS, jQuery stuff when you want to select certain parts of the document.

Even using AJAX, you trigger events based on certain parts of the DOM.

Using the DOM, you can access every node in an HTML document in 3 ways:

1. By using the getElementsById() method
2. By using the getElementsByTagName() method
3. By navigating the node tree, using the node relationships

Ex: document.getElementById("intro"); //this gets the element with id = intro
Ex: document.getElementsByTagName("p"); //returns a nodeList of all <p> elements in the document

DOM Rule: Every node is an object.
Objects have methods and properties that can be accessed and manipulated by JavaScript.

Three important node properties:

1. nodeName
2. nodeValue
3. nodeType

The nodeName property specifies the name of a node.
nodeName is read-only
nodeName of an element node is the same as the tag name
nodeName of an attribute node is the attribute name
nodeName of a text node is always #text
nodeName of the document node is always #document
*note: nodeName always contains the uppercase tag name of an HTML element.

The nodeValue property specifies the value of a node.
nodeValue for element nodes is undefined
nodeValue for text nodes is the text itself
nodeValue for attribute nodes is the attribute value

Here is an example of retrieving the text node value:

```
<html>
  <body>
    <p id="intro">Hello World!</p>
    <script type="text/javascript">
      x=document.getElementById("intro");
      document.write(x.firstChild.nodeValue); //will write “Hello World!”
    </script>
  </body>
</html>
```

The nodeType property returns the type of node.
nodeType is read-only

Most important node types:
1. element
2. attribute
3. text
4. comment
5. document

HTML elements can be changed using JS, the HTML DOM, and events.

Change an HTML element:
Here is an example of using HTML DOM and JS to change the background color of a <body> element.

```
<html>
  <body>
    <script type="text/javascript">
      document.body.bgColor="lavender";
    </script>
  </body>
</html>
```

The easiest way to modify the content of an element is by using the innerHTML property.
This example changes the text of a <p> element.

```
<html>
  <body>
    <p id="p1">Hello World!</p>
    <script type="text/javascript">
      document.getElementById("p1").innerHTML="New text!";
    </script>
  </body>
</html>
```

An event handler allows you to execute code when an event occurs.
Events are generated by the browser when the user clicks an element, when the page loads, when a form is submitted, etc..

Ex: this changes the background color of the <body> element when a button is clicked

```
<html>
  <body>
    <input type="button" onclick="document.body.bgColor='lavender';"
value="Change           background color" />
  </body>
</html>
```

Events are actions that can be detected by JavaScript. Every element on a web page has certain events, which can trigger JS functions

Ex: Using the onClick event of a button element to indicate that a function will run when a user clicks on the button.

Other examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input box in an HTML form
- Submitting an HTML form
- A keystroke

**note - events are usually used in combination with functions, and the function will not be executed before the event occurs*

unload and onUnload are two events that are triggered when a user enters or leaves a page.

The onload event is usually used to check the visitor's browser type and version and load the proper version of the web page based on that information

onFocus, onBlur, and onChange

These events are often used in combination with validation of form fields

Below is an example of how to use an onChange event. The checkEmail() function will be called whenever the user changes the content of the e-mail field:

Ex: E-mail: <input type="text" id="email" onchange="checkEmail()" />

**Note - Scripts in an HTML file are executed on the client (in the browser), scripts in PHP/ASP file are executed on the server*

For further reference, please visit http://w3schools.com/htmldom/default.asp

4. JavaScript (JS)

JavaScript was designed to add interactivity to HTML pages. It is a scripting language, which means lightweight programming language.

JavaScript is largely used to give HTML designers a tool. JS can react to events. It can be set to execute when something happens, such as when a page has finished loading or when a user clicks an HTML element. This is what allows for the dynamic creation of content.

One common use of JS is to validate form input data. Other uses include detecting a visitor's browser data and creating cookies.

HTML <script> tag is used to insert a JavaScript into an HTML document.
HTML id attribute is used to identify HTML elements (read up on DOM).

To access an HTML element from a JS, use the document.getElementById() method
    That method will access the HTML element with the specified id

JS Rule: how to comment out the JS → <!--   -->

```
<h1>My Web Page</h1>
  <p id="demo">A Paragraph</p>
  <script type="text/javascript">
    document.getElementById("demo").innerHTML="My First JavaScript";
  </script>
  </body>
</html>
```

In this example, the JS is executed when the page loads but that is not always what we want. Sometimes we want to execute a JS when an event occurs such as when a user clicks a button. When we want this, we need to put the JS inside a function so we can reference it. You generally put these JS in the head of the html doc.

Ultimately though what you really want to do is use an external javascript, as it is better coding practice.
    External js files can contain code to be used on several different web pages
    They must have the extension of .js

Example of how to reference external JS:
```
<!DOCTYPE html>
<html>
  <body>
   <script type="text/javascript" src="myScript.js"></script>
  </body>
</html>
```

Another way to think of JavaScript is basically as a sequence of statements to be executed by the browser. A JS statement is basically a command to a browser, where the purpose of the command is to tell the browser what to do.

Commenting in JS → (commenting in JS is the same as commenting in Java)
single line comments start with //
multi-line comments are /* */

**note - JS is case sensitive (unlike HTML)*

Variables in JavaScript:
Variables are containers for storing information
Variable names must begin with a letter, $, or _

To declare JS variables: var carname;
To assign value: carname = "volvo";

**note - JavaScript operators are basically the same as Java / Ruby*

JavaScript switch statement:
Use switch statement to select one of many blocks of code to be executed
If there is no match, you want to use the default keyword to specify what to do

Ex:

```
var day = new Date().getDay();
switch (day) {
case 6:
  x="Today it's Saturday";
  break;
case 0:
  x="Today it's Sunday";
  break;
default:
  x="Looking forward to the Weekend";
}
```

JavaScript popup boxes:
There are 3 kinds: alert, confirm, and prompt

Alert box is used to make sure information comes through to the user. When alert box pops up, user will have to click OK to proceed.
Ex: alert("sometext");

Confirm box is used if you want the user to verify or accept something. When confirm pops up, user will have to click either OK or Cancel to proceed.

This confirm box returns true if user clicked ok, false if clicked cancel.

Ex:

```
var r=confirm("Press a button"); //notice how you set a variable to the confirm
box to get the value
if (r==true)
  {
  x="You pressed OK!";
  }
else
  {
  x="You pressed Cancel!";
  }
```

Prompt box is used if you want the user to input a value before entering a page. When a prompt box pops up, the user will click OK or Cancel to proceed before entering an input value.

If the user clicks OK, the box will return the value they inputted.
If the user clicks Cancel, the box will return null.

Ex:

```
var name=prompt("Please enter your name","Harry Potter");
if (name!=null && name!="")
  {
  x="Hello " + name + "! How are you today?";
  }
```

Here is how you display line breaks in a pop up box

Ex: alert("Hello\nHow are you?");

JavaScript functions:

A function can be executed by an event such as clicking a button
Function is a block of code that executes only when you tell it to execute

Ex:

```
function myFunction()
{
alert("Hello World!");
}
        //when you call a function, you can also pass it parameters
```

Sometimes, you want your function to do more than just an alert statement, you want your function to return a value back to where the call was made.

Ex:

```
function myFunction()
{
var x=5;
return x;
}
```

You can also use return if you want to exit a function. The return value is optional.

JavaScript events **IMPT**
Events are actions that are detected by JavaScript.
Ex: onClick()

By using events in JS, you have the ability to create dynamic web pages.
Every element in JS has certain events, which can trigger JS.

More comprehensive examples of events:
Mouse click
Web page or an image loading
Mousing over a hot spot on the web page
Selecting an input field in an HTML form
Submitting an HTML form
A keystroke

onLoad and onUnload are events that are triggered when a user enters or leaves the page.
onFocus, onBlur, and onChange are events used in combination with validation of form fields.

Ex. of using the onChange event:

```
<input type="text" size="30" id="email" onchange="checkEmail()" />
```

//checkEmail() function will be called whenever the user changes the content of the field

onSubmit event is used to validate all form fields before submitting it.
Ex: `<form method="post" action="xxx.htm" onsubmit="return checkForm()">`

onMouseOver event can be used to trigger a function when the user mouses over an HTML element.
It's pretty tricky though because you have to use coordinates.

Try..Catch Statement
This statement allows you to test a block of code for errors.
Basically, the try block contains the code to be run and the catch block contains the code to be executed if an error occurs.

There is also the Throw statement that is used with the try…catch statement.

Throw statement allows you to create an exception. An exception can be a string, integer, Boolean, or an object.

Ultimately, Try…Catch…Throw are used to control program flow and generate accurate error messages.

Ex:

```
<!DOCTYPE html>
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 5 and 10:","");
try {
  if(x>10) {
    throw "Err1"; }
  else if(x<5) {
    throw "Err2"; }
  else if(isNaN(x)) {
    throw "Err3"; }
  }
catch(err) {
  if(err=="Err1"){
    document.write("Error! The value is too high.") }
  if(err=="Err2"){
    document.write("Error! The value is too low.");}
  if(err=="Err3"){
    document.write("Error! The value is not a number.");}
  }
</script>
</body>
</html>
```

Here, we are throwing/outputting the error name the error name.

JavaScript also has its own section on special characters.

Ex: '\' is used to insert apostrophes, new lines, quotes, and other special characters into a text string.

Final things to know when scripting in JavaScript:

JS is case sensitive.

JS ignores extra spaces, so you can add extra white space to make your text more readable.

You can break up a code line within a text string with a backslash.

For further reference, please visit http://w3schools.com/js/default.asp

5. AJAX

AJAX stands for asynchronous JavaScript and XML.

AJAX is not a programming language; rather, it is a way to use existing standards.
AJAX is the art of exchanging data with a server and updating parts of the page without having to reload the entire page.

You use AJAX to create fast and dynamic web pages

It allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes.
Generally, web pages will reload the entire page if the content should change.

How AJAX works:

An event occurs in the browser so you create an XMLHttpRequest object and send that http request and the server will then process that HTTP request.
Server then creates a response and sends the data back to the browser.
The browser will then process the returned data using JavaScript and update the page content.

Google Suggest is an example of something that uses AJAX.

When you type into the search bar, JavaScript sends the letters off to the server and the server returns a list of suggestions.

So to use AJAX, you do it on an event. For example, the click of a button:

Just include the onclick attribute to a button.
Ex: `<button type="button" onclick="loadXMLDoc()">Change Content</button>`

You also need to add a script tag to the head of the page. Obviously, the script needs to contain the loadXMLDoc() function.

Ex:

```
<head>
  <script type="text/javascript">
    function loadXMLDoc() {
       .... AJAX script goes here ...
    }
  </script>
</head>
```

The keystone to Ajax is the XMLHttpRequest Object.

This is used to exchange data with the server behind the scenes.
To create this object, it is pretty simple:

Ex: `variable=new XMLHttpRequest();`

To send a request to the server you need to use both the open and send methods of the object.

Ex:

```
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
```

open(method,url,async)

Method is always either a GET or POST // the type of request
URL is the location of the file (.txt, .xml, .asp, .php) on the server
async can be TRUE or FALSE

send(string)

Sends the request off to the server
The string is only used for POST requests

Asynchronous requests are a huge time improvement for web developers. After all, many of the tasks performed on the server are very time consuming. Before AJAX, this would cause the application to hang.

The way async works is that the JavaScript does not have to wait for the server response

It can execute other scripts while waiting for the server response
It deals with the response when the response is ready

To get a response from the server, use the responseText or the responseXML property of the object.

responseText gets the response data as a string
responseXML gets the response data as XML data
Ex: document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

Using a Callback Function
Def.: a callback function is a function that is passed as a parameter to another function

Callback function Ex:

```
function myFunction() {
loadXMLDoc("ajax_info.txt",function() {
  if (xmlhttp.readyState==4 && xmlhttp.status==200) {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  });
}
```

To use this bit of AJAX in your HTML, you must trigger functions in your HTML. These triggers are hit by events such as "onkeyup" or "on click."

For further reference, please visit http://w3schools.com/ajax/default.asp

6. jQuery

jQuery can be added to a web page with just a single line of markup. It is incredibly easy to insert and use.

What is jQuery?

It is a library of JavaScript functions.
It is a lightweight; often called the "write less, do more" JavaScript library.

jQuery has the following features:

HTML element selections
HTML element manipulation
CSS manipulation
HTML event functions
JavaScript Effects and animations
HTML DOM traversal and modification
AJAX
Utilities

The jQuery library is stored in a single - huge - file that contains all the jQuery methods.

Ex of how to implement jQuery:

```
<head>
        <script type="text/javascript" src="jquery.js"></script>
</head>
```

When using jQuery, you select (query) HTML elements and perform actions on them.

Ex of jQuery syntax:

```
$(this).hide()
$("p").hide()
```

As can be seen, the basic jQuery syntax is: $(selector).action()

Whenever you see that dollar sign $, that defines jQuery
A (selector) is used to query (find) the HTML elements
The .action() is performed on the elements

Ex: $(this).hide() - hides current element
Ex: $("p.test").hide() - hides all paragraphs with class="test"
Ex: $("#test").hide() - hides the element with id="test"

Document Ready Function:

All jQuery methods are inside a document.ready() function.
This is to prevent jQuery code from running before the document is finished loading.

Ex:

```
$(document).ready(function(){
  // jQuery functions go here...
});
```

Here are some examples of actions that can fail if functions are run before the document is fully loaded:

Trying to hide an element that does not exist.
Trying to get the size of an image that is not loaded.

jQuery Selectors (**IMPT**)

allow you to select/manipulate HTML elements as a group of as a single element
allow you to get the exact element/attribute you want from your HTML document

Ex of jQuery selectors:

$("*") selects all elements.
$("p") selects all <p> elements.
$("p.intro") selects all <p> elements with class="intro".
$("p#intro") selects the first <p> elements with id="intro".
$(":animated") selects all elements that are currently animated.
$(":button") selects all <button> elements and <input> elements of type="button".
$(":even") selects even elements.
$(":odd") selects odd elements.
$(this) Selects the current HTML element

$("p#intro:first")	Selects the first <p> element with id="intro"
$(".intro")	Selects all elements with class="intro"
$("#intro")	Selects the first element with id="intro"
$("ul li:first")	Selects the first <li> element of the first <ul>
$("ul li:first-child")	Selects the first <li> element of every <ul>
$("[href]")	Selects all elements with an href attribute
$("[href$='.jpg']")	Selects all elements with an href attribute that ends with ".jpg"
$("[href='#']")	Selects all elements with an href value equal to "#"
$("[href!='#']")	Selects all elements with an href value NOT equal to "#"
$("div#intro .head")	Selects all elements with class="head" inside a <div> element with id="intro"

jQuery is tailor-made to handle events

Event handlers are methods that are called when something happens in HTML

Similar to JS, it is better to put your jQuery functions in a separate .js file.

Ex:

```
<head>
  <script type="text/javascript" src="jquery.js"></script>
  <script type="text/javascript" src="my_jquery_functions.js"></script>
</head>
```

Ex of event methods:

$(document).ready(function) → binds a function to the ready event of a document
$(selector).click(function) → triggers, or binds a function to the click event of selected elements
$(selector).dblclick(function) → triggers a function to the double click event of selected elements
$(selector).focus(function) → binds a function to the focus event of selected elements

Ultimately, most of the value of jQuery lies in its ability to perform a lot of cool effects.
Ex: hide, show, toggle, slide, fade, animate

For further reference, please visit http://www.w3schools.com/jquery/jquery_effects.asp

7. SQL

SQL, which stands for structured query language, is the standard language to access and manipulate a database. Knowing SQL will let you use MySQL, SQL Server, Access, Oracle, etc…

Starter SQL syntax:

```
SELECT Company, Country FROM Customers WHERE Country <> 'USA'
```

**note - SQL is pretty standard but most SQL database programs have their own extensions in addition to the SQL standard.*

To build a website that shows data from a database, you will need the following:

- A database program such as MySQL
- A server side scripting language like PHP (Ruby also works)
- SQL
- HTML/CSS

RDBMS - stands for relational database management system

- This is the basis for SQL and for all modern database systems
- The data in RDMS is stored in database (db) objects called tables
- A table is a collection of related data entries and it consists of columns and rows

A database will oftentimes consist of more than one table:

- Each table is defined by a name such as "Customers" and "Orders"
- Tables contain records (which are rows) of data

For example, in a Customers table of 3 rows:

- Each row is a different person and is also a different record.
    - Thus, record is kind of like a record of what we have in our table.
- The table may also have numerous columns where each column represents an attribute of each customer.

Most of the actions you need to perform on a database are done with SQL statements.

Ex: the following SQL statement will select all the records in the Persons table

```
SELECT * FROM Persons
```

**notes - SQL is not case sensitive, and some db systems require a semicolon at the end of each SQL statement*

SQL can be divided into two parts: DML and DDL:

- DML - data manipulation language
- DDL - data definition language

For instance, the DML part of SQL deal largely with the query and update commands:

SELECT - extracts data from the database
UPDATE - updates data in a database
DELETE - deletes data from a database
INSERT INTO - inserts new data into a database

**note - these commands are very similar to CRUD of REST API actions and get, post, put, delete of HTTP requests*

The DDL part of the SQL permits database tables to be created or deleted:
CREATE DATABASE - creates a new db
MODIFY DATABASE - modifies a db
CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX (an index is a search key), DROP INDEX

SQL SELECT statement:
SELECT - used to select data from a database
The result is stored in a result table called a result set

Ex:
SELECT column_name(s)
FROM table_name
SELECT * FROM table_name (* means grab everything)

To select columns from a table:
SELECT LastName,FirstName FROM Persons
**note - you are grabbing the LastName and FirstName columns from the Persons table*

When you want to select all the columns: SELECT * FROM Persons

There is also the SELECT DISTINCT Statement:
In some columns you may have duplicate values. If you only want to get the different values in the table then you want to use the SELECT DISTINCT

Syntax:
SELECT DISTINCT column_name(s)
FROM table_name

Ex: SELECT DISTINCT City FROM Persons

The WHERE clause is used to filter records (remember that records are just rows, i.e. instances of a person)
The WHERE clause is used to extract only those records that fulfill a specified criterion

Syntax:
SELECT column_name(s)

```
FROM table_name
WHERE column_name operator value
```

Ex:

```
SELECT * FROM Persons   //select every column from persons
WHERE City='Sandnes'   //where the columns must have the city be Sandnes
```

**note - SQL uses single quotes around text values but numeric values should not be enclosed in quotes*

Here are the operators allowed in the WHERE clause:

= : equal
<> : not equal
> : greater than
< : less than
<= : less than or equal
>= : greater than or equal
BETWEEN : between an inclusive range
LIKE : search for a pattern
IN : to specify multiple possible values for a column

AND & OR operators are used to filter records based on more than one condition:

AND displays a record if both the first condition and the second condition is true
OR displays a record if either the first condition or the second condition is true

Example of using the AND operator to link two conditions:
**note - conditions are basically WHERE statements*

```
SELECT * FROM Persons
WHERE FirstName='Tove'
AND LastName='Svendson'
```

**note - you could do the same thing with the OR operator*

You can also combine AND and OR. Just make sure you use parenthesis to form complex expressions.

Ex. of combining the operators:

```
SELECT * FROM Persons WHERE
LastName='Svendson'
AND (FirstName='Tove' OR FirstName='Ola')
```

Order By keyword is used to sort the result set:
By default, this keyword sorts the records in ascending order (you can use the DESC keyword to order in descending order)

Ex of using SELECT and ORDER BY keywords together:

```
SELECT * FROM Persons
```

```
ORDER BY LastName
```

Explanation: You grab all the information from the PERSONS table and then you order them by last name.

The INSERT INTO statement is used to insert new records (new row) in a table:

There are 2 ways to write the INSERT INTO statement. One is more explicit than the other.

Form 1: do not specify the column names where the data will be inserted, only their values.

```
INSERT INTO table_name
VALUES (value1, value2, value3,...)
```

Form 2: specify both the column names and the values to be inserted

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Ex. that inserts another record to the end of the Persons table:

```
INSERT INTO Persons
VALUES (4,'Nilsen', 'Johan', 'Bakken 2', 'Stavanger')
```

It is also possible to add data only to specific columns. For example, the following SQL statement will add a new row but only add data to the P_Id, LASTNAME, and FIRSTNAME columns:

```
INSERT INTO Persons (P_Id, LastName, FirstName)
VALUES (5, 'Tjessem', 'Jakob')
```

**note how the other columns / fields are left empty*

The UPDATE statement is used to updated records in a table

**note – it only applies to existing records*

Syntax:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

The WHERE clause is incredibly important because it specifies which record or records that should be updated. If you do omit the WHERE clause, then all the records get updated.

Ex.:

```
UPDATE Persons
SET Address='Nissestien 67', City='Sandnes'
WHERE LastName='Tjessem' AND FirstName='Jakob'
```

The DELETE statement is used to delete rows in a table.

Syntax:

```
DELETE FROM table_name
WHERE some_column=some_value
```

**note the importance of the WHERE clause, as it is what specifies what records will be deleted; without it, all records will be deleted*

Ex:

```
DELETE FROM Persons
WHERE LastName='Tjessem' AND FirstName='Jakob'
```

To delete everything: DELETE * FROM table_name

**note - you must be very careful when deleting records because you cannot undo the statement*

More Advanced SQL commands:

TOP clause:

Used to specify the number of records to return

This can be important when you have a large database and you don't want to return so many records as that drives performance down

Ex:

```
SELECT TOP number|percent column_name(s)       //either select a number of a percentage
FROM table_name
```

**note - unfortunately, not all databases support this exact TOP method. mySQL has something a little bit different and you'll have to go into the specifics for each db.*

LIKE operator:

Used in a WHERE clause to search for a specific pattern in a column

For further reference, please visit http://www.w3schools.com/sql/default.asp