DOM, Javascript, and jQuery

Aryo Pinandito, ST, M.MT

HTML DOM

- DOM stands for the Document Object Model.
- The HTML DOM is the Document Object Model for HTML.
- The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML objects.
- Traversing, editing and modifying DOM nodes
- Editing text nodes

HTML DOM

- The HTML DOM is a platform and language independent API (application program interface) and can be used by any programming language
- The HTML DOM is used to manipulate HTML documents
- DOM makes all components of a web page accessible
  - HTML elements
  - their attributes
  - text
- They can be created, modified and removed with JavaScript
  - We will use Javascript to interface with the HTML DOM

DOM Objects

- DOM components are accessible as objects or collections of objects
- DOM components form a tree of nodes
  - relationship parent node – children nodes
  - **`document`** is the root node
- Attributes of elements are accessible as text
- Browsers can show DOM visually as an expandable tree
  - Firebug for Firefox
  - in IE -> Tools -> Developer Tools

DOM Standards

- W3C www.w3.org defines the standards
- DOM Level 3 recommendation
  - www.w3.org/TR/DOM-Level-3-Core/
- DOM Level 2 HTML Specification
  - www.w3.org/TR/DOM-Level-2-HTML/
  - additional DOM functionality specific to HTML, in particular objects for XHTML elements
- But, the developers of web browsers
  - don't implement all standards
  - implement some standards differently
  - implement some additional features

Accessing Nodes by id

- Access to elements by their id
  - **`document.getElementById(<id>)`**
    - returns the element with **`id <id>`**
  - **`id`** attribute can be defined in each start tag
    - **`div`** element with **`id`** attribute can be used as an root node for a dynamic DOM subtree
    - **`span`** element with **`id`** attribute can be used as a dynamic inline element
  - The preferred way to access elements

Other Access Methods

- Access by elements' tag
  - there are typically several elements with the same tag
  - **`document.getElementsByTagName(<tag>)`**
    - returns the collection of all elements whose tag is <tag>
    - the collection has a **`length`** attribute
    - an item in the collection can be reached by its index
  - e.g.

```
html = document.getElementsByTagName("html")[0];
```

- Access by elements' name attribute
    - several elements can have the same name

```
document.getElementsByName(<name>)
```

    - returns the collection of elements with name <name>

Other Node Properties

- `nodeName` property
- `nodeValue` property
- `attributes` property
- `innerHTML` property
  - not standard, but implemented in major browsers
  - very useful
- `style` property
  - object whose properties are all style attributes, e.g., those defined in CSS

Accessing JS Object's Properties

- There are two different syntax forms to access object's properties in JS (
  - **`<object>.<property>`**
  - dot notation, e.g., **`document.nodeType`**
  - **`<object>[<property-name>]`**
    - brackets notation, e.g., **`document["nodeType“]`**
    - this is used in **`for-in`** loops

- this works for properties of DOM objects, too

Attributes of Elements

- Access through **`attributes`** property
  - **`attributes`** is an array
  - has a **`length`** attribute
  - an item can be reached by its index
  - an item has the properties **`name`** and **`value`**
  - e.g.

```
src=document.images[0].attributes[0].value;
```

- Access through function **`getAttribute(<name>)`**
  - returns the value of attribute **`<name>`**
  - e.g.

```
src=document.images[0].getAttribute("src");
```

Text Nodes

- Text node
  - can only be as a leaf in DOM tree
  - it's **`nodeValue`** property holds the text
  - **`innerHTML`** can be used to access the text

Modifying DOM Structure

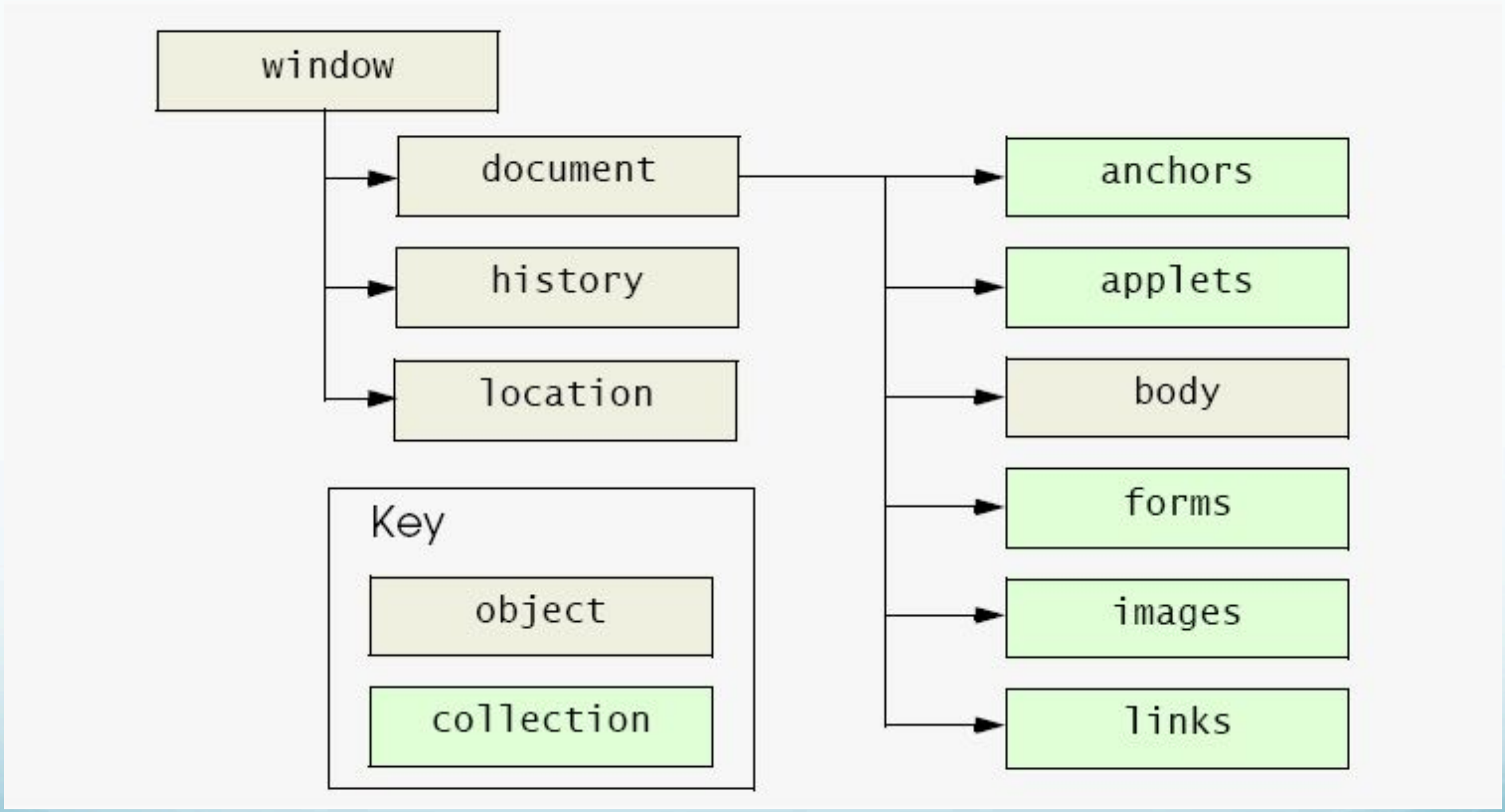
- **`document.createElement(<tag>)`**
  - creates a new DOM element node, with <tag> tag.
  - the node still needs to be inserted into the DOM tree
- **`document.createTextNode(<text>)`**
  - creates a new DOM text with <text>
  - the node still needs to be inserted into the DOM tree
- **`<parent>.appendChild(<child>)`**
  - inserts <child> node behind all existing children of <parent> node
- **`<parent>.insertBefore(<child>,<before>)`**
  - inserts <child> node before <before> child within <parent> node
- **`<parent>.replaceChild(<child>,<instead>)`**
  - replaces <instead> child by <child> node within <parent> node
- **`<parent>.removeChild(<child>)`**
  - removes <child> node from within <parent> node

Modifying Node Attributes

- `<node>.setAttribute(<name>,<value>)`
  - sets the value of attribute <name> to <value>
  - e.g.
    - `document.images[0].setAttribute("src","keiki.jpg");`

- That's the standard
  - but it doesn't work in IE, there you have to use
    - `setAttribute(<name=value>)`
  - e.g.
    - `document.images[0].setAttribute("src=\"keiki.jpg \"");`

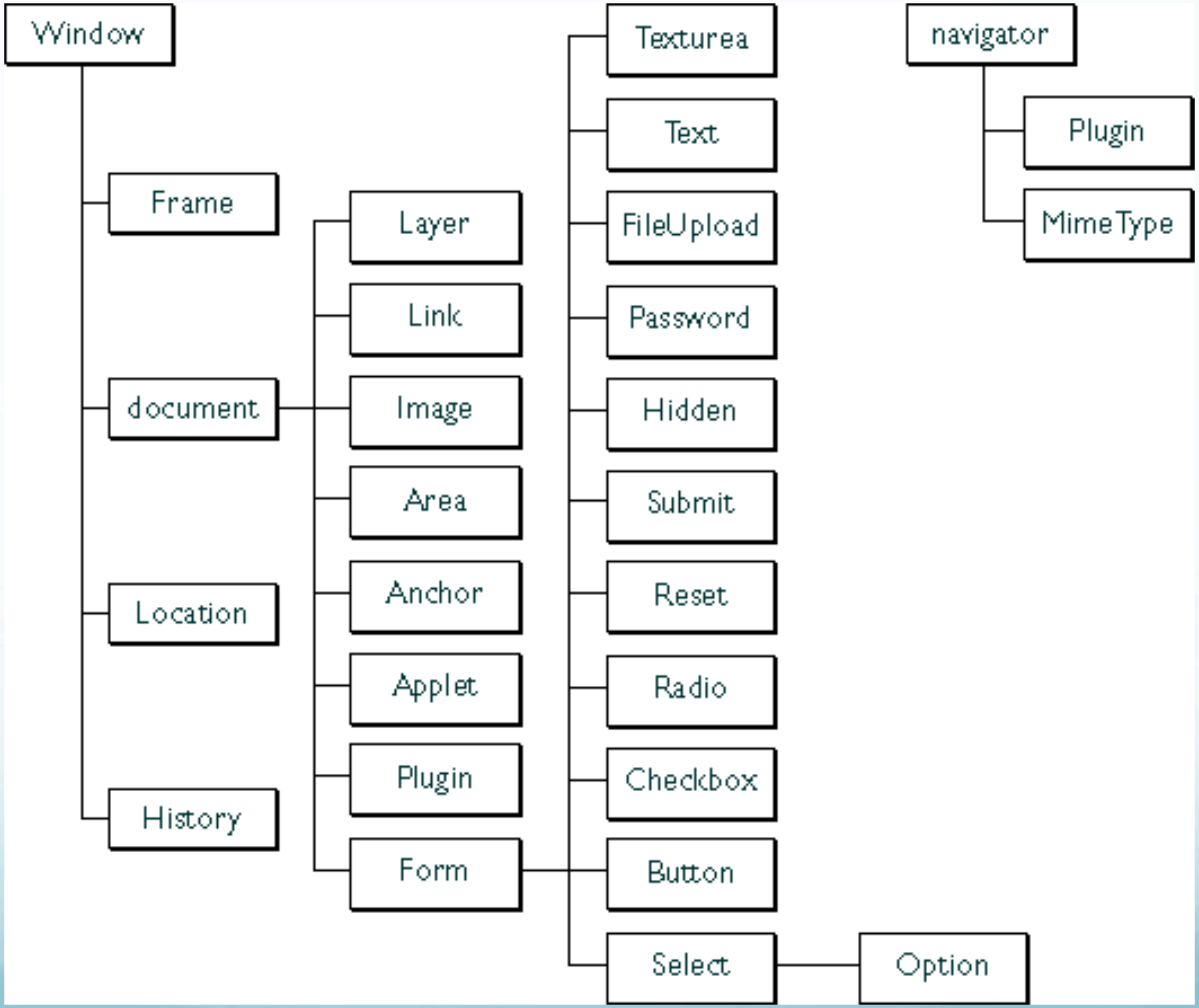
W3C Document Object Model



Special DOM Objects

- `window`
  - the browser window
  - new popup windows can be opened
- `document`
  - the current web page inside the window
- `body`
  - <body> element of the document
- `history`
  - sites that the user visited
  - makes it possible to go back and forth using scripts
- `location`
  - URL of the document
  - setting it goes to another page

HTML DOM



Window
Frame
document
Location
History
Layer
Link
Image
Area
Anchor
Applet
Plugin
Form
Texturea
Text
FileUpload
Password
Hidden
Submit
Reset
Radio
Checkbox
Button
Select
Option
navigator
Plugin
MimeType

An HTML DOM Example

This coding example shows how the background color of an HTML document can be changed to yellow when a user clicks on it:

```
<html>
<head>
  <script language = “javascript">
    function ChangeColor() {
      document.body.bgColor="yellow" ;
    }
  </script>
</head>
<body onclick="ChangeColor()">
  Click on this document!
</body>
</html>
```

- http://www.w3schools.com/js/tryit.asp?filename=try_dom_change_color

HTML DOM

- DOM Event
  - onBlur, onClick, onChange, onFocus, onKeyDown, onKeyUp, onKeyPress, onLoad, onMouseDown, on MouseMove, onMouseOut, onMouseOver, onSubmit, ...
  - http://science.slc.edu/~sallen/s05/examples/events.html

JavaScript

Introduction to JavaScript

- NOT Java
  - JavaScript was developed by Netscape
  - Java was developed by Sun
- Designed to ‘plug a gap’ in the techniques
- available for creating web-pages
  - Client-side dynamic content
- Interpreted

JavaScript

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language - a scripting language is a lightweight programming language
- A JavaScript is lines of executable computer code
- A JavaScript is usually embedded directly in HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license
- JavaScript is supported by all major browsers.

JavaScript

- JavaScript gives HTML designers a programming tool.
- JavaScript can put dynamic text into an HTML page like this:

```
document.write("<h1>" + name + "</h1>")
```

  - can write a variable text into an HTML page
- JavaScript can react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element
- JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server, this will save the server from extra processing

JavaScript vs. Java

- JavaScript
  - Cannot draw, multi-thread, network or do I/O
- Java
  - Cannot interact with Browser or control content
- JavaScript is becoming what Java was originally intended to be
  - Java Applets are meant to be lightweight downloadable programs run within the browser for cross-platform compatibility
  - Java = Bloated
  - JavaScript is actually lightweight and accomplish most of what Applets do with a fraction of the resources

What is it used for today?

- Handling User Interaction
  - Doing small calculations
  - Checking for accuracy and appropriateness of data entry from forms
  - Doing small calculations/manipulations of forms input data
  - Search a small databased embedded in the downloaded page
  - Save data as cookie so it is there upon visiting the page
- Generating Dynamic HTML documents
- Examples
  - Bookmarklets
  - Google Maps
  - Google Suggest

JavaScript

- How to Put a JavaScript Into an HTML Page

```
<html>
 <body>
 <script language=“javascript">
   document.write("Hello World!");
 </script>
 </body>
</html>
```

JavaScript

- Scripts in a page will be executed immediately while the page loads into the browser.
- This is not always what is wanted. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.
- Scripts in the head section will executed when they are called, or when an event is triggered
- When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

JavaScript

- If you want to run a script on several pages, you can write a script in an external file, and save it with a .js file extension, like this:
- document.write("This script is external") Save the external file as externalJS.js.
- Note: The external script cannot contain the <script> tag
- This script can be called using the "src" attribute, from any of your pages:

```
<html>
  <head>
  </head>
  <body>
  <script src="externalJS.js"></script>
  </body>
</html>
```

JavaScript

- Variables
- A variable is a "container" for information you want to store. A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.
- Rules for Variable names:
  - Variable names are case sensitive
  - They must begin with a letter or the underscore character

  - http://www.w3schools.com/js/tryit.asp?filename=tryjs_variable

JavaScript

- You can create a variable with the var statement:

```
var strname = some value
```

- You can also create a variable without var:

```
strname = some value
```

- Assigning a Value to a Variable

```
var strname = “Sam"
```

- Or like this:

```
strname = “Sam"
```

- The variable name is on the left side of the expression and the value you want to assign to the variable is on the right. Now the variable "strname" has the value “Sam".

JavaScript

- Functions
- A function contains some code that will be executed by an event or a call to that function.
- A function is a set of statements. You can reuse functions within the same script, or in other documents.
- You define functions at the beginning of a file (in the head section), and call them later in the document.

JavaScript

- To create a function you define its name, any values ("arguments"), and some statements:

```
function myfunction(argument1,argument2,etc) {
  // some statements
}
```

- A function with no arguments must include the parentheses:

```
function myfunction() {
  // some statements
}
```

JavaScript

- Arguments are variables used in the function. The variable values are values passed on by the function call.
- By placing functions in the head section of the document, you make sure that all the code in the function has been loaded before the function is called.

JavaScript

- A function is not executed before it is called.
- You can call a function containing arguments:

  ```
  myfunction(argument1,argument2,etc)
  ```

- To call a function without arguments:

  ```
  myfunction()
  ```

jQuery

write less, do more.

Javasript Framework

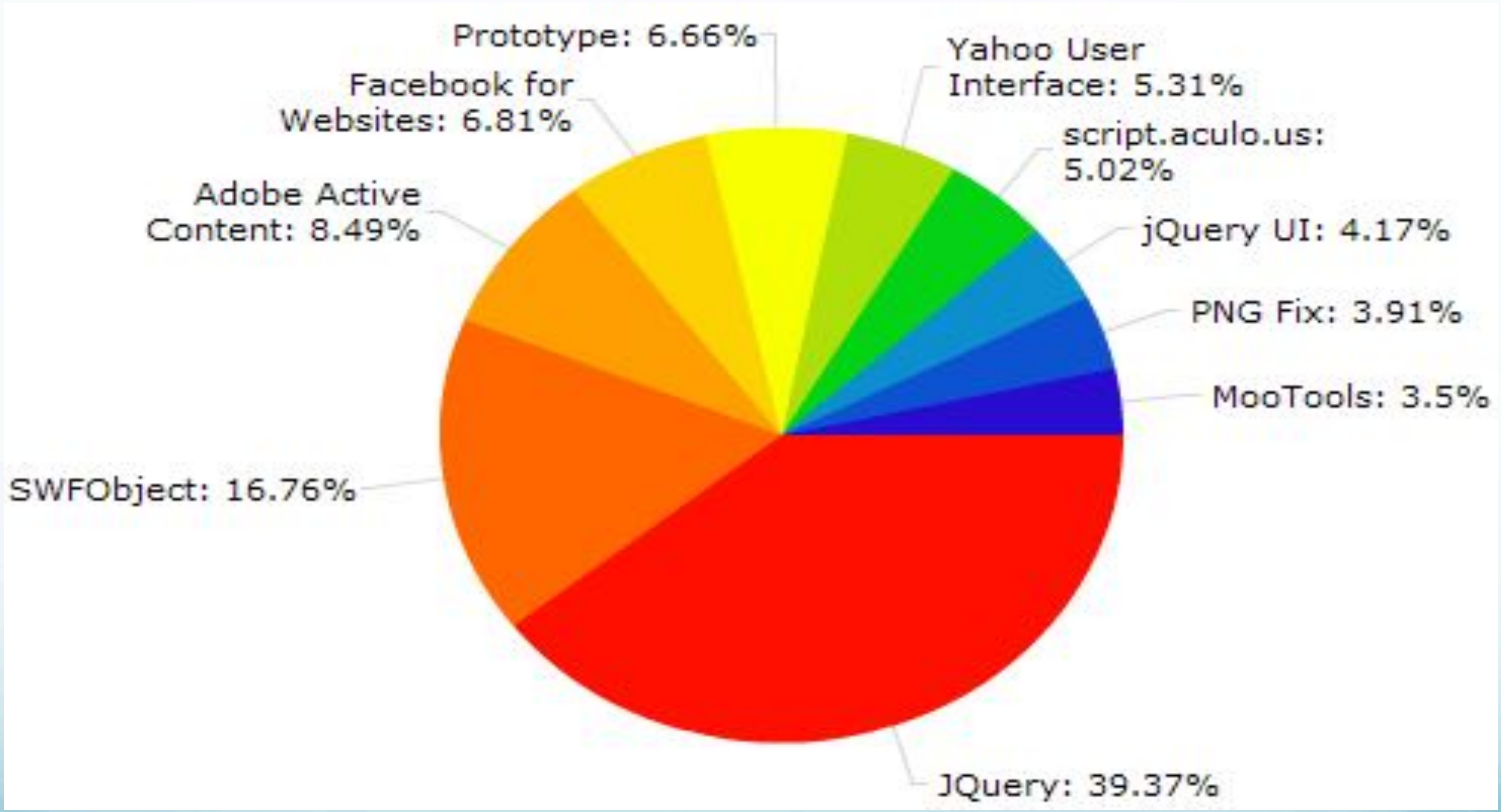
A Little Bit About jQuery

- What is jQuery?
  - jQuery is an Open-Source JavaScript framework that simplifies cross-browser client side scripting.
    - Animations
    - DOM manipulation
    - AJAX
    - Extensibility through plugins
  - jQuery was created by John Resig and released January 2006
  - Most current release is 1.8.3 (2012)

Why should you use it?

- Easy to learn! It uses CSS syntax for selection
- Its tiny 252KB (32KB, minified and Gzipped)
- Documented api.jquery.com & Supported forum.jquery.com
- Cross browser compatibility: IE 6+, FF 2+
- It is the most used JavaScript library on the web today
  - 39% of all sites that use JavaScript use jQuery.
    - trends.builtwith.com/javascript/JQuery

PWNS All Other Frameworks



Who Uses jQuery?

Google

Amazon

IBM

Microsoft

Twitter

Dell

docs.jquery.com/Sites_Using_jQuery

What is the DOM?

Document Object Model (DOM): noun
Blah blah blah long definition that makes little sense….

What Is The DOM?

- Long story short, the DOM is your html document code. From the
- <!DOCTYPE> to the </html>

- The DOM is loaded top to bottom, so include your scripts at the bottom of the page for best performance.

- The DOM is "ready" when everything on the page has loaded.
- Stylesheets
- JavaScripts
- Images

Wait!!

- In order to make sure that jQuery can find the element you asked it for, your browser needs to have loaded it (the DOM needs to be ready).

- Q. How can I be sure my code runs at DOM ready?
- A. Wrap all your jQuery code with the document ready function:

```
$(document).ready(function(){
  // insert javascript/jQuery code here
});
```

And What If I Don't Wanna, Huh?

1 of 3 things will happen:

1. Code doesn't work, throws an error (90%)
2. Code works...
   this page load, next page load see #1. (~9%)
3. Code opens a worm hole that transports your page back to 1990 revolutionizing the Web as we know it. While seemingly great, it also creates a paradox and destroys the universe. * (<1%)
   - *(has yet to be fully verified)

Loading jQuery

- In order to use jQuery you need to load it.
- You can include it locally on your own server:

```
<script src="/js/jquery.js">
```

- Or use one of the CDN's made available:
  - ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js
  - ajax.microsoft.com/ajax/jquery/jquery-1.4.2.js
  - CDN's are Gzipped and minified

Load Scripts At The Bottom

Problem:
When scripts are downloading they block everything else in almost all browsers!

Solution:
Best practice: Load your scripts at the bottom of your page so they don't interrupt page content downloads.

I COULD RESTRUCTURE THE PROGRAM'S FLOW
OR USE ONE LITTLE 'GOTO' INSTEAD.
EH, SCREW GOOD PRACTICE. HOW BAD CAN IT BE?
goto main_sub3;
COMPILE

jQuery = $

- The dollar sign is a synonym for the jQuery function

And BOOM! Goes The Dynamite.

- Html:

```
<p>Hello World! I'm Aryo</p>
```

- Script:

```
    $(function(){
            $("p").addClass("isCool");
            //keep telling yourself that..
    });
```

- Resulting html:

```
<p class="isCool">Hello World! I'm Aryo</p>
```

Break It Down Now!

```
$(function(){// = $(document).ready(function(){
```

$

Initiates the jQuery function

$
=
jQuery

("p")

Grabs a DOM element using a CSS selector.

Selector is in quotes.

Creates a jQuery “Collection”

<p>

.addClass("isCool");

Built in method that adds a class to the jQuery Collection

Class is in quotes.

ends with a semicolon.

```
});
```

All Your Basic Selectors Are Belong To Us

- Uses the same syntax you use to style elements in CSS!

$("p")	$("div")	$("#foo")	$(".foo")
<p>	<div>	id="foo"	class="foo"

api.jquery.com/category/selectors/

Get Classy <p>

- jQuery:

```
$("p").addClass("sophisticated");
```

- Before:

```
<p>
```

- After:

```
<p class="sophisticated">
```

This <p> Has No Class At All!

- jQuery:

```
$("p").removeClass("sophisticated");
```

- Before:

```
<p class="sophisticated">
```

- After:

```
<p class="">
```

<div> Hide and Seek

- jQuery:

```
$("div").show();
```

- Before:

```
<div style="display:none;">
```

- After:

```
<div style="display:block;">
```

I'm Not Lame, Am I?

- jQuery:

```
$("#aryo").text("Is Cool");
```

- Before:

```
<p id="aryo">Is Lame</p>
```

- After:

```
<p id="aryo">Is Cool</p>
```

You Can Chain Most Methods Together

```
$("p")
     .addClass("sophisticated")
     .text("Hello World!")
     .show();
```

"Daisy Chain!"

Some of Basic Methods

`.show()`	• Show a hidden element
`.wrap("<a></a>")`	• wrap an element with <a>
`.parent("p")`	• Select parent <p>
`.html()`	• Get/Set innerHTML
`.val()`	• Get/Set Value

api.jquery.com/

Getters and Setters

Dual Purpose Methods

Getter

```
$("#foo").text();
```

Setter

```
$("#foo").text("foo");
```

Use jQuery To Get

- `<p>Panda</p>`

```
$("p").text();
```

- === "Panda"

```
myVar = $("p").text();
```

- myVar === "Panda"

Use jQuery To Set

- `<p>Panda</p>`

```
$("p").text("BigPanda");
```

- `<p>BigPanda</p>`

```
myVar = "BigPanda";
$("p").text(myVar);
```

- `myVar === "BigPanda"`
  `<p>BigPanda</p>`

jQuery: Get and Set

```
<a href="http://berkeley.edu">UC Berkeley</a>
```

```
var a = $(‘a’).text();

$(‘a’).text(‘Hello world’);

var href = $(‘a’).attr(‘href’);

$(‘a’).attr(‘href’, ‘http://google.com’);
```

Complete list at http://api.jquery.com/category/attributes/

jQuery: Events

```
$(element).eventType(function(){
    // JavaScript
});
```

General Events	ready, load, scroll
Mouse Events	click, hover, mouseenter, mouseleave
Keyboard Events	keypress, keydown, keyup
Forms Events	submit, focus, blur

Complete list at http://api.jquery.com/category/events/

jQuery: Live Events

```
$('li').click(function(){
     // Do something
});


$('li').live('click', function(){
     // Do Something
});
```

A normal event binding attaches to all matched elements when it is called. A live event calls the callback function when the event occurs on all matched element, *current and future*.

Click Events Are Awesome!

```
$("#panda").click(function(){
     $(this).text("Is Cool"); // this = #panda
     alert("Take that Zoo!");
});

$("#panda").click(function(event){
     $(this).text("Is Cool"); // this = #panda
     alert("Take that Zoo!");

     //Prevents default action
     event.preventDefault();
});
```

jQuery: Forms

```
<input id="name" type="text" value="John">
```

```
$(‘#name’).val();

$(‘#name’).val(‘Doe’);

$(‘#name’).attr(‘value’);

$(‘#name’).attr(‘value’, ‘Doe’);
```

Complete list at http://api.jquery.com/category/forms/
See the documentation for .val() in particular: http://api.jquery.com/val/

jQuery: CSS

```
<h1>Hello world</h1>
```

```
$(‘h1’).css(‘color’, ‘red’);

$(‘h1’).addClass(‘important’);

$(‘h1’).hide();

$(‘h1’).fadeIn();
```

Complete list at http://api.jquery.com/category/css/

“this” in JavaScript

```
var person = {
    name: 'Mohit',
    sayHello: function(){
        alert('Hello, ' + this.name);
    }
}
```

this is a special variable.
It is the object in the current context.

“this” in jQuery

```
$('li').click(function(){
    $('li').hide();
});

$('li').click(function(){
    this // DOM element
    $(this) // jQuery object
});
```

Plugins

UNLEADED FUEL ONLY
PREMIUM 91
CARBURANT SANS PLOMB
SUPER 95

Viva Variety!

- "If you want to create an animation, effect or UI component, chances are pretty good that someone has done your work for you already.”
- -Eric Steinborn 2010

- plugins.jquery.com

AJAX and Cross-site Scripting

- Web 2.0 FTW

AJAX

- *Web 3.0? – More Semantic!*

AJAX What?

- Asynchronous
- Javascript
- and
- XmlHttpRequest

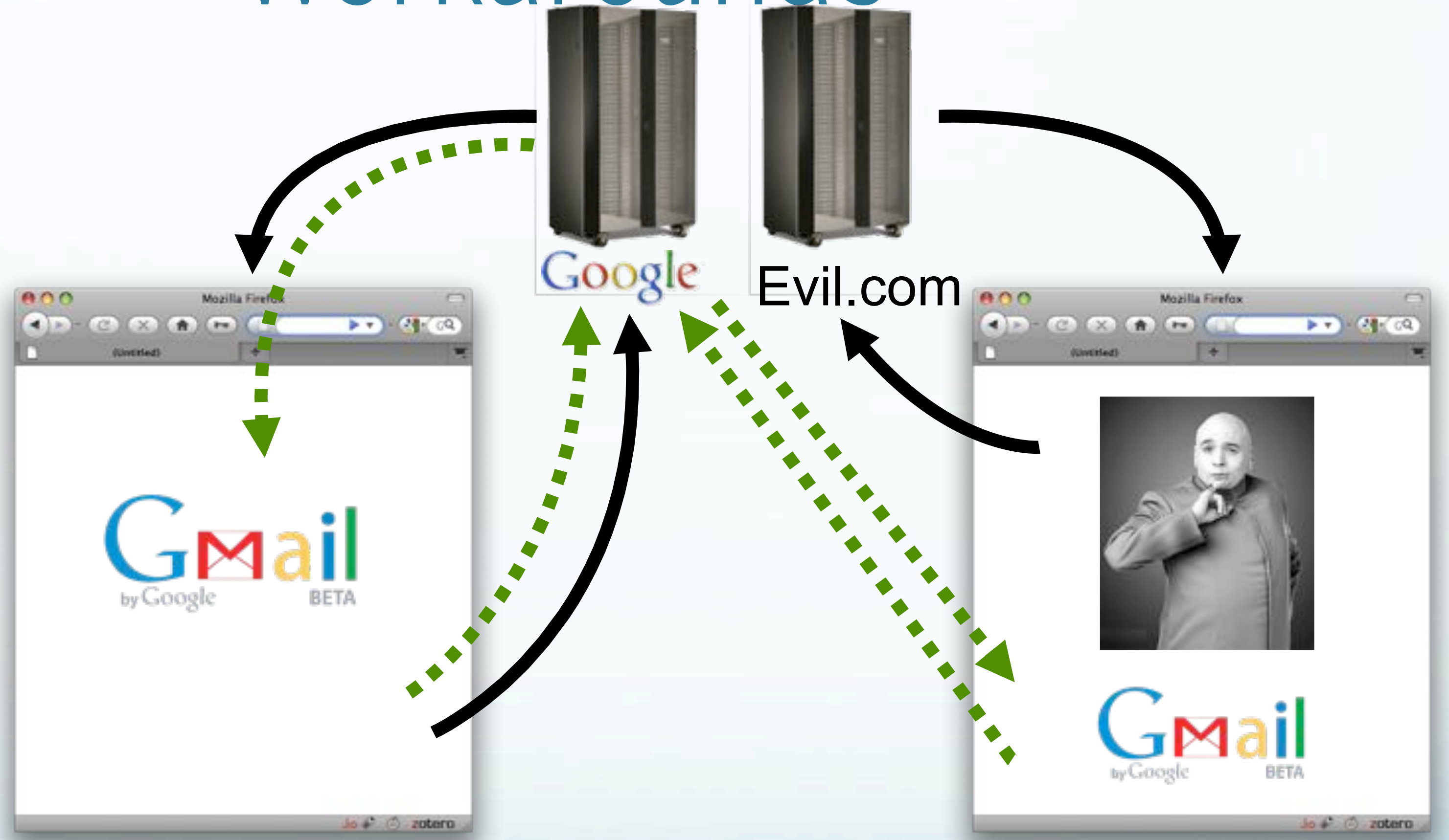
AJAX What?

```
$.get('http://gmail.com', function(xml){
   console.log(xml);
});
```

same-origin policy

- (Alas, no cross-site scripting!)

Cross-site scripting Workarounds



🟦 Normal Webpage 🟩 AJAX

- Proxy server
- JSONP
- Trusted contexts

Example – Show/Hide the old way

```
<a href="#"
onclick="toggle_visibility('foo');">Click here
to toggle visibility of #foo</a>

function toggle_visibility(id) {
  var e = document.getElementById(id);
  if(e.style.display == 'block')
    e.style.display = 'none';
  else
    e.style.display = 'block';
}
```

Example – Show/Hide with jQuery

```
$().ready(function(){
  $("a").click(function(){
     $("#more").toggle("slow");
          return false;
  });
});
```

Example – Ajax the Old Way

```
function GetXmlHttpObject(handler) {
    var objXmlHttp = null;    //Holds the local xmlHTTP object instance
    //Depending on the browser, try to create the xmlHttp object
    if (is_ie){
            var strObjName = (is_ie5) ? 'Microsoft.XMLHTTP' : 'Msxml2.XMLHTTP';
            try{
                        objXmlHttp = new ActiveXObject(strObjName);
                        objXmlHttp.onreadystatechange = handler;
            }
            catch(e){
            //Object creation errored
            alert('Verify that activescripting and activeX controls are enabled'); return;
            }
    }
            else{
            // Mozilla | Netscape | Safari
            objXmlHttp = new XMLHttpRequest();
            objXmlHttp.onload = handler;
            objXmlHttp.onerror = handler;
    }
    //Return the instantiated object
    return objXmlHttp;
}
```

Example – Ajax with jQuery

```
$.get(“controller/actionname",
  { name: "John", time: "2pm" },
  function(data){
    alert("Data Loaded: " + data);
  });

$.post(“controller/actionname",
  { name: "John", time: "2pm" },
  function(data){
      alert("Data Loaded: " + data);
});
```

Example – Form Validation

```
$().ready(function(){
   // validate the comment form when it is submitted
   $("#commentForm").validate();
});

<input id="cname" name="name" class="some other styles
   {required:true,minLength:2}" />
<input id="cemail" name="email"
   class="{required:true,email:true}" />
```

Great References

- John Resig's introduction slides
- jQuery 1.4 Cheat Sheet
- jQuery API
- jQuery Forums
- YAYquery Podcast (explicit)

FROM THE CREATORS OF LAST SUMMER'S HIT THRILLER SNAKES ON A PLANE COMES:
SNAKES...
ON
EVERY
PLANE!
MUCH WORSE THAN LAST TIME.

Questions?