Training

DOM Basics

DOM / JavaScript Basics

DOM Basics Review

- The Document Object Model
- Using Built-In Objects Document, Window.
- Java Script basics
- Java Script cross browser issues
- Built-in Objects
- Manipulating Document structure dynamically
- Ajax functionalities and issues

Document Object Model (DOM)

When you load a document in a Browser, it creates a number of JavaScript Objects with Property values based on the HTML in the document and other pertinent information. These Objects exist in a Hierarchy that reflects the structure of the HTML page itself. The ability to change a Web page dynamically with a scripting language is made possible by the **Document Object Model** (**DOM**) which can connect any element on the screen to a JavaScript function. The **DOM** is the road map through which you can locate any element in your HTML document and use a script, such as JavaScript, to change the element's properties.

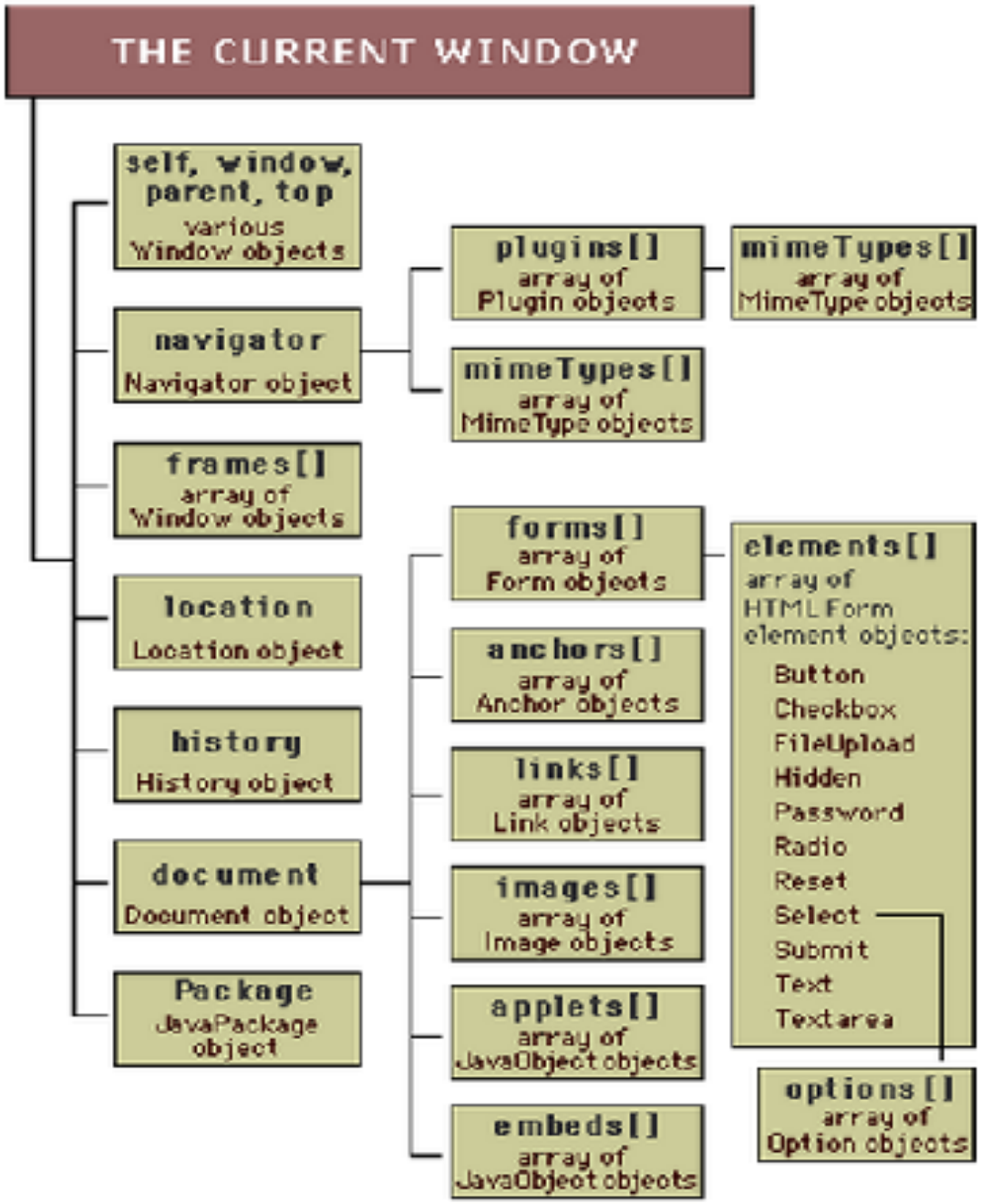
Document Object Model (DOM)

So think of each Web page as a collection of several individual elements, which are called Objects. For example, every Image on the page is an Object, every Link on the page is an Object. Even this Document itself is an Object. At its most basic level, JavaScript allows you to control the appearance of many of the Objects that make up a Web page as we previously saw.

Document Object Model (DOM)

Objects are storage containers that have Properties (data values associated with Objects) and Methods (functions associated with Objects) that operate on that data. Objects may also have certain Events that are associated with them. Events are special signals or messages which occur when certain pre-defined actions take place within a Web browser, or when the user interacts with a Web page. When an event message has been triggered, you need a way to intercept the message and react to it. This is achieved through the use of Event Handlers.

HTML DOM


THE CURRENT WINDOW
self, window, parent, top
various Window objects
navigator
Navigator object
plugins[]
array of Plugin objects
mimeTypes[]
array of MimeType objects
mimeTypes[]
array of MimeType objects
frames[]
array of Window objects
location
Location object
history
History object
document
Document object
Package
JavaPackage object
forms[]
array of Form objects
elements[]
array of HTML Form element objects:
Button
Checkbox
FileUpload
Hidden
Password
Radio
Reset
Select
Submit
Text
Textarea
options[]
array of Option objects
anchors[]
array of Anchor objects
links[]
array of Link objects
images[]
array of Image objects
applets[]
array of JavaObject objects
embeds[]
array of JavaObject objects

Object names include the names of Objects that contain them

The name of each Object is prefaced by the names of all the Objects that contain it, in order, separated by dots. For example, the window Object contains the document Object, and the document Object contains a form Object, and the form Object contains text input fields, buttons, select Objects, etc... — window.document.forms[0].elements[3]

Leave off the word "window."

All Object names officially start with the highest Object name, "window". But since all names start with "window", JavaScript allows you to leave that off. (document.forms[0].elements[3])

Multiple Objects are organized by Arrays

Any type of Object, such as a Form, an Image, or Links may have multiple instances on a page is referenced by an Array. Array names are formed by the plural of the Object name (i.e. "the forms Array" or "the images Array" or "links Array") and indexed by number using square brackets [ ]. In an Array, counting — begins at zero — and the items in the array are sequenced in the same order they are in the HTML code that created them.

Example: document.**forms**[0].**elements**[1].**value**

Form elements

The numerous Object types included in the Form**s** Array are collectively referred to as "elements" and they can all be referred to by means of the "Element**s** Array" (e.g.: element**s**[1]).

all four rules

- It includes the names of all the Objects that contain it,
- the word "window" has been left off,
- the form and element Objects are shown as indexed arrays (which start at 0), and
- the text input Object is referred to as a part of the elements array.

Every page has the following objects:

- navigator: has properties for the name and version of the Navigator being used, for the MIME types supported by the client, and for the plug-ins installed on the client.
- window: the top-level object; has properties that apply to the entire window. There is also a window object for each "child window" in a frames document.
- document: contains properties based on the content of the document, such as title, background color, links, and forms.
- location: has properties based on the current URL.
- history: contains properties representing URLs the client has previously requested.

Example:

- For example, suppose you create a page named **simple.htm** that contains the following HTML:
- <HTML>
  <HEAD><TITLE>A Simple Document</TITLE></HEAD>
- <BODY bgColor="White">
- <FORM NAME="myform" METHOD="POST" ACTION="/cgi-bin/mail.cgi">
  Enter a value:
  <INPUT TYPE="text" NAME="text1" VALUE="blahblah">
  Check if you want:
  <INPUT TYPE="checkbox" NAME="Check1" VALUE="ON" CHECKED
  onClick="**update**(this.form)">Option #1
  <INPUT TYPE="button" NAME="Button1" VALUE="Press Me"
  onClick="**update**(this.form)">
  </FORM>
- </BODY>
  </HTML>

Example:

- As always, there would be window, location, history, and document objects. These would have properties such as:
- location.**href** is "http://pyther.com/samples/simple.htm"
- document.**title** is "A Simple Document"
- document.**fgColor** is "#000000"
- document.**bgColor** is "#FFFFFF"
- history.**length** is "3"

Example:

These are the full names of the objects, based on the DOM Hierarchy.

- document.myform => the form
- document.myform.text1 => the text field (elements[0])
- document.myform.Check1 => the checkbox (elements[1])
- document.myform.Button1 => the button (elements[2])

JavaScript:

JavaScript

- Originally developed by Brendan Eich of Netscape
- No Java (no VM)
- Scripting Language
- Mostly used by browsers
- JavaScript is a Object Oriented Language

For statement

- Iterate through all of the members of an object:

```
for (var name in object) {
    if (object.hasOwnProperty(name)) {

        // within the loop,
        // name is the key of current member
        // object[name] is the current value

    }
}
```

typeof

The **typeof** prefix operator returns a string identifying the type of a value.

type	typeof
object	'object'
function	'function'
array	'object'
number	'number'
string	'string'
boolean	'boolean'
null	'object'
undefined	'undefined'

```
var myvar=5;
alert(typeof myvar) //alerts "number"
```

Inner functions

- Functions do not all have to be defined at the top level (or left edge).
- Functions can be defined inside of other functions.

Scope

- An inner function has access to the variables and parameters of functions that it is contained within.
- This is known as Static Scoping or Lexical Scoping.
- If `new` is omitted, the global object is clobbered by the constructor, and then the global object is returned instead of a new instance.

Examples

simple-js.html
browserinfo.html
if-else.html
try-catch.html
js-throw.html

built-in JavaScript objects

JS String

```
var txt="Hello world!";
document.write(txt.length);
document.write(txt.toUpperCase());
```

JS Date

```
var today = new Date()
   var d1 = new Date("October 13, 1975 11:13:00")
   var d2 = new Date(79,5,24)
   var d3 = new Date(79,5,24,11,33,0)
```

built-in JavaScript objects

JS Array

```
var myCars=new Array(); // regular array (add an
   optional integer
   myCars[0]="Saab";       // argument to control
   array's size)
   myCars[1]="Volvo";
   myCars[2]="BMW";
```

JS Boolean

```
var myBoolean=new Boolean();
```

Default is false

built-in JavaScript objects

JS Math

```
document.write(Math.round(4.7));
document.write(Math.random());
```

built-in JavaScript objects

JS RegExp

var patt=new RegExp(pattern,modifiers);
or more simply:
var patt=/pattern/modifiers;

js-reg-exp-1.html

js-reg-exp-2.html

Cookies

Variable that is stored on the visitor's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With JavaScript, you can both create and retrieve cookie values.

js-cookies.html

Cross Browser Javascript

Why bother?

If you don’t have to, don’t.

If you think you might have to, it’s better to do it up front.

What are the big differences?

- DOM (Document Object Model) support
  - How to get a reference to an element.

What are the big differences?

- DOM support
  - How to get a reference to an element.
  - Cross Browser

    ```
    document.getElementById( id );
    document.getElementsByTagName( name );
    ```

  - IE 5.5+, Mozilla

What are the big differences?

- DOM support
  - How to get a reference to an element.
  - IE only

```
document.elementName;
document.all[name];
```

What are the big differences?

- DOM support
  - How to get a reference to an element.
  - Netscape only.
    `document.layers[`*name*`];`

What are the big differences?

- DOM Support
- Navigating the DOM
  - IE and Mozilla support the standard methods

What are the big differences?

- DOM Support
- Event Model
  - IE uses window.event
  - Mozilla passes an event argument to the event handler.

What are the big differences?

- Event model
- Cross browse way to handle events

```
<a href="" onclick="handleEvent(event);"/>
<script>
function handleEvent(aEvent)
{
     var theEvent = aEvent ? aEvent :
window.event;
}
</script>
```

Common Tips

- Don’t test for specific browsers, test for functionality.

```
this.ie     = ((agt.indexOf("msie") != -1) &&
(agt.indexOf("opera") == -1));
this.ie3    = (this.ie && (this.major < 4));
this.ie4    = (this.ie && (this.major == 4) &&
(agt.indexOf("msie 4")!=-1) );
this.ie4up  = (this.ie  && (this.major >= 4));
this.ie5    = (this.ie && (this.major == 4) &&
(agt.indexOf("msie 5.0")!=-1) );
this.ie5_5  = (this.ie && (this.major == 4) &&
(agt.indexOf("msie 5.5") !=-1));
this.ie5up  = (this.ie  && !this.ie3 && !this.ie4);
this.ie5_5up =(this.ie && !this.ie3 && !this.ie4 && !this.ie5);
this.ie6    = (this.ie && (this.major == 4) &&
(agt.indexOf("msie 6.")!=-1) );
this.ie6up  = (this.ie  && !this.ie3 && !this.ie4 && !this.ie5
&& !this.ie5_5);
```

Common Tips

- Quirks mode vs Standards mode
  - Mozilla will adapt based on the DOCTYPE

- The many modes of Mozilla
  - Standards Mode
  - Almost Standards mode
  - Quirks mode

The many modes of Mozilla

- Standards Mode
  - text/xml (xhtml)
  - Unknown doctype
  - Doctype html system

The many modes of Mozilla

- Almost Standards mode
  - Any “loose” doctype
  - The IBM doctype

The many modes of Mozilla

- Quirks mode
  - emulates some IE “quirks”
  - triggered by no doctype or a doctype without “system”

Common Tips

- Whitespace nodes
  - Mozilla skips some whitespace nodes
  - Check “nodeType” property and only process type 1 nodes.

Common Tips

- OuterHTML and InnerText
  - only supported in IE
  - solution: roll your own using `__defineGetter__` and `__defineSetter__` in Mozilla.

InnerText for Mozilla

```
<script language="JavaScript">
<!--
if(HTMLElement.innerText == undefined)
{
     HTMLElement.prototype.__defineGetter__
("innerText", function () {
     var r = this.ownerDocument.createRange();
   r.selectNodeContents(this);
   return r.toString();
     });
}
//-->
</script>
```

Common Tips

- getYear() == getFullYear() in IE
- getYear() != getFullYear() in Mozilla
  - Mozilla returns 1900 - the current year. IE for 2005 it returns “105”.

What is Ajax?

Asynchronous **J**avascript **A**nd **X**ML

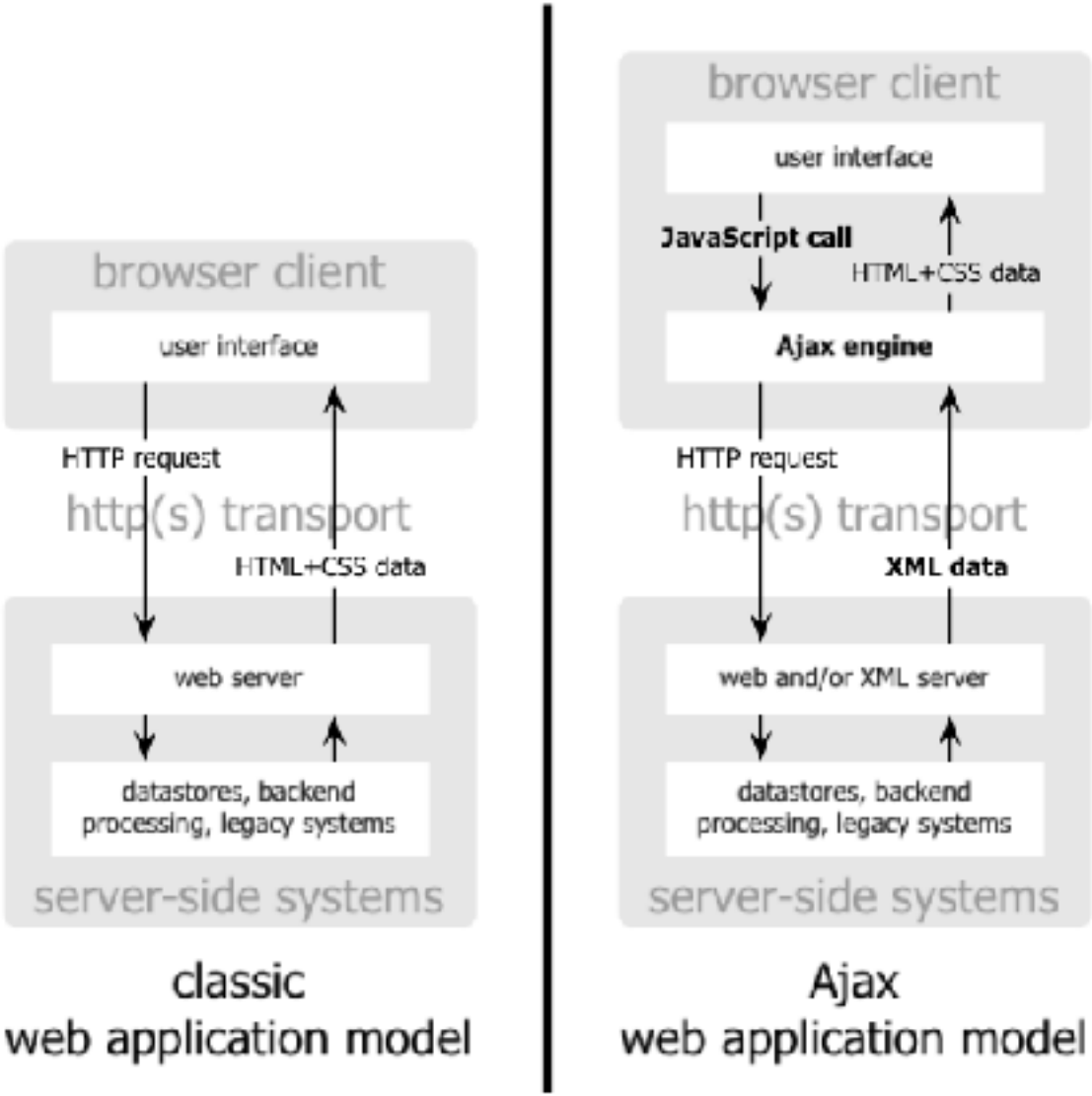
What is so cool about Ajax?

- Connection between client side script and server side script.
- Better user experience
- More flexibility
- More options

How does Ajax work?

A client script asynchronously calls a server side function.

Classic VS Ajax



Ajax In the Real World

- IRCTC (Indian Railways)
- Windows Live Mail (Hotmail Beta)
- Google Maps
- Gmail

Scenario

Car Classifieds website has a dropdown with the makes of all the cars. Based on the selection of the “makes” dropdown the “models” dropdown has to be populated with the correct models provided by the manufacturer.

Advantages of Ajax

1. With Ajax, the page can be refreshed dynamically.
2. Much faster response for user actions
3. Less bandwidth consumption

Disadvantages of Ajax

1. Complexity (applications have become complex, this is an issue that must be addressed in the future.) Now the separation of data and formats is also important.
2. The page that is dynamically created will not automatically register with the history engine of the browser, so pressing the "back" button might now allow them to get what they want.
3. No supported by Search Engine Crawler (SEO issues)