Copyright Notice

JavaScript Review

Prakash Badhe

prakash.badhe@vishwasoft.in

About Me

- ✓Director Technology Practices - AgileSoft Methodologies
- ✓Web Technologies consultant and mentor
- ✓Enjoying the IT for 15 years
- ✓Passion for technologies and frameworks
- ✓Worked on medium and large sized dynamic web projects.
- ✓ Skills in java script frameworks
- ✓ Proficient with Html5.0,Ajax,XML standards
- ✓ Working with YUI,GWT ,AngualrJS,Ext JS, JQuery JavaScript frameworks
- ✓ Developed applications with Ajax, JQuery, EXT JS, YUI and AngularJS frameworks.

About you..

- **Prerequisite** : Awareness about internet applications and programming skills.
- Job Role and skill-Sets
- Training Objective
- Your job experience in Web applications development
- Awareness about server side programming platform java/MS.NET.
- Prior experience with Java Script, JQuery, AngularJS or any other JavaScript library or framework.

Agenda

- JavaScript language review
- JavaScript Cross Browser issues
- JavaScript DOM manipulation
- JavaScript Applications
- Ajax review
- Ajax applications with JavaScript
- Ajax with JSON
- The AngularJS framework introduction
- Building AngularJS applications.
- AngularJS best practices and tools.

Setup

- Windows 7 64 bit
- Adobe PDF Reader
- JetBrains WebStorm Editor
- JDK1.8 64 bit
- Eclipse-Jee-Luna
- Apache Tomcat 8.0 Web Server
- Firefox Mozilla, Google Chrome browser
- Node JS,MS Visual Studio 2013.
- Karma, Protractor and other tools.
- Live Internet connection

Three Pillars of Web

- HTML
- JavaScript
- CSS

HTML for the Web

- HTML is the language defined using markup tags.
- The tags in html use English words to represent html keywords.
- The tags have pre-defined meaning for visual graphic content to be displayed.
- HTML is used as a communication language between the web server and web client, the browser.
- The server sends the html contents and browser interprets and displays it as graphics like buttons, form, checkbox, radio buttons, image, text etc.

Web Client Dynamics

- Html is static!
- For dynamic html responses, we need dynamic html generation from *server* side by applying active programming with java/asp.net/cgi/pearl /php etc.
- Inside the browser on *client* side dynamic graphics, effects, animations and much more active behavior is done in the browser by programming java Script along with html in web pages.
- For those client side dynamics by JavaScript and CSS, the server side communication is minimized and is responsive to the user, ***Always***!.

Java Script

- NetScape created JavaScript.
- The JavaScript has been defined to add dynamism to html pages in the form of user interactions, event listeners, validation, server interactions etc.
- Java Script code is included within html/dynamic web pages by using <script> tags.
- The JavaScript code in the html page is interpreted at runtime by the JavaScript interpreter engine in the browser and executed line by line.
- The java script makes most part of the application client centric with minimum server interaction.
- Nothing to do with java.

Web Standards

- JavaScript is the open source standard scripting language and most of recent versions of web browsers support java script.
- Earlier MS IE browsers had their own scripting language VbScript which was not supported on other browsers.
- A common standard ‘ECMA 3/4/5.0’ is defined for java script.
- HTML5.0 for universal web across different devices
- Cascaded Style sheet (CSS3)

JavaScript Applications

- It’s widely used in tasks ranging from the validation of form data to the creation of complex user interfaces inside the browser, dynamically.
- JavaScript supports dynamic manipulation (update/delete/create) of the visual graphics contents of the web page.
- JavaScript combined with CSS is used to create animations and graphic effects in web pages.
- JavaScript supports dynamic event handling for graphic components like button, check box etc.

Java script apps..

- JavaScript supports user interaction in the form of taking inputs from the user, display messages, dialogs and submitting data to server programs.
- JavaScript also supports pulling data from the server and process it on the client side asynchronously(New addition with Ajax).
- By utilizing the Java script the page running in browser is self contained with minimum server interactions.

Java Script in html

```
<html>
<head>
<title>JavaScript Program</title></head>
 <body>
  <h1 align="center">Welcome!</h1>
  <script type="text/JavaScript">
      alert(“Welcome to JavaScript!");
  </script>
</body>
</html>
```

The same script tags can be used inside <head> tags and anywhere in body tags as well

External Java Script

- Sometimes java script code is defined in external files as .js and referred in html pages.
- The html page can include multiple js external files in the head tag.
- *<script type="text/JavaScript“ src=“myScript.js">*
- *</script>*
- This allows reuse of same js code in multiple web pages.

Java Script Core Features

- JavaScript is case-sensitive language.
- JavaScript is a dynamic language.
- Supports variables and procedural functions
- Supports logical, relational, conditional and mathematical and loop operators.
- All JavaScript keywords are lowercase.
- JavaScript is an object-based language
- JavaScript code in HTML pages is interpreted and executed line by line sequentially in the page.
- Supported on mobile browsers.

Java script compatibility

- Browsers add the extensions to JavaScript which most times is not supported by other browsers.
- Some of the java script behavior is being implemented differently in different browsers and some is not supported.
- This makes the web page dependent on the particular browser for execution.
- Techniques/frameworks to make web page browser independent.

Java Script core..

- JavaScript program contains statements and functions that work with objects and other functions.
- The java script engine supports built-in objects as well as user defined custom objects.
- Certain code such as the bodies of functions and actions associated with event handlers is not immediately executed.
- They are invoked when events are triggered, functions are to be invoked.

Statements in JavaScript

- Statement is an execution unit in JavaScript.
- They are instructions to the interpreter to carry out specific actions and tasks.
- For example, one of the most common statements is variable declaration and *assignment*. X = 10;
- The statements that make up the body of a function are enclosed in curly braces.
- Semicolons indicate the end of a JavaScript statement ,but is not required if you put every statement on new line
- Multiple statements can be grouped in one line by separating them by semicolons.

Comments in JavaScript

- An aspect of good programming style is commenting your code. Commenting allows to insert remarks and commentary directly in source code, making it more readable to yourself as well as to others.
- Any comments you include is ignored by the JavaScript interpreter.
- Comments in JavaScript are similar to those in C++ and Java. Comments that run to the end of the current line and those that span multiple lines.
- Single-line comments begin with a double fore slash **(*//*)**
- Comments spanning multiple lines are enclosed C-style between a slash-asterisk **(*/**)** and asterisk-slash **(***/*)** pair.

Variables..

- A variable stores data values in a program.
- Every variable has a name, called its *identifier*.
- Variables are declared in JavaScript using **var** keyword. i.e var name;
- Variables can be assigned initial values when they are declared.i.e var x = 12;
- The variables assigned with values gets allocated in program memory space and indicates to the interpreter that a new identifier is in use. i.e *var x=12*
- *The variables can get overwritten with other values.*

The keyword var

- The variables can be declared without *var* keyword. also i.e. simply.. X=10.(Global variables).
- With var keyword the variables inside functions become local variables.
- Without the var keyword, it implicitly defines global variable even inside functions…
- The multiple variables can be declared with one **var** statement if the variables are separated by commas:
- var x, y = 2, z;

Global, implicit and local Scope

- The variables declared as implicit global without var keyword can be deleted by ‘delete’ operator.
- The global variables declared with var keyword are deleted when browser is closed or new page is loaded.
- The variables declared inside functions with var keyword are deleted when the function local scope is destroyed.

Global and local

- How to differentiate between global and local variables defined by same name in a function and outside it ?
- The *window* is the container for all global declared variables and functions.i.e window.name
- The local variable simply accessed by name..

Functions in JavaScript

- Functions are used to carry out certain processing work, handle certain events ,interact with the user etc.
- JavaScript functions are declared with the **function** keyword, name, arguments and the statements that carry out their operations are listed in curly braces following declaration.
- Function does not specify return and parameter types.
- *function* add(x, y)

```
{
  var sum = x + y;
  return sum;
}
```

Functions..

- Functions are defined to perform a specific task and are reusable by calling any number of times by their names.
- The functions enhance reusability and program modularity.
- The JavaScript functions can be assigned as callback for event notifications.
- Functions can be defined as self executing/calling functions.

JaavScript Basic Data Types

- Every variable has a *data type* that indicates the type of data the variable holds and decides on memory allocation.
- The basic data types are **string, number, and boolean and objects.**
- A string is a array of characters, and a string literal is indicated by enclosing the characters in single or double quotes.
- Numbers are integers or floating-point numerical values.
- Booleans take on one of two values: true or false.
- Two other types: **undefined** and **null** for system**.**
- Custom data types are also defined.

The undefined and null

- Window.alert(val);
- The system finds the *val* as the undefined type if it is not defined/not assigned value.
- The **null** value indicates an special value represents ‘nothing’ and not zero value..
- The distinction between **undefined** and **null** values is *undefined* means the variable hasn’t been initialized/not declared, whereas *null* means the variable is declared but has been set to be null value to prevent its re-use.

Declare Data Type?

- The variables are declared in java script as
  - var stringName = "JavaScript has strings\n It sure does";
  - var numericData = 3.14;
  - var booleanType = true;

- To define variables in JavaScript, there is no need to indicate their type in the declarations such as ‘int num = 12;’ instead it is simply declared as ‘*num =12*’ or *var num=12*
- The data type of variable is **not** defined in the var declaration!
- How the data type is identified ?

Weak Data typing

- The data type of variable is determined at run time!
- ---*depending on the value assigned…*
- Every JavaScript variable will have a data type, but the type is identified from the variable's content value at runtime..
- For example, a variable that is assigned a string value assumes the string data type and so on.
- The java script is **weakly typed** language.

Dynamic Data Typing

- JavaScript is *dynamically typed since the types are identified at runtime due to its interpreted nature of execution*.
- A consequence of JavaScript's automatic type detection is that a variable's type can change during script execution according to the value assigned...
- For example, a variable can hold a string value at one point and then later be assigned a boolean value as true/false.
- How to identify the current data type of the variable then ?
- *The 'typeof' operator is used to identify the current data type of variable.*

The typeof Operator

- If want to know about the type of variable, use the **typeof** operator to examine it.
- Applied to a variable or literal, it returns a string indicating the type of its argument.

The return of type..

Type	Result
Undefined	**undefined**
Null	**object**
Boolean	**boolean**
Number	**number**
String	**string**
Object	**object**
Function	**function**

Arguments types and return types

- How the types of function arguments are determined in function definition ?
- At run time depending on the values passed by the caller ..
- Similarly how the function return type is determined ?
- Again at run time depending on the value type assigned to return statement.

Return of the function

- What can be the type of return data from the function to the caller ? :any type
- The ‘return’ statement is used to explicitly return from the function execution at any moment.
- The return can also be used to specify returning value to the caller.

Number of arguments…

- How many arguments needed to be passed by the caller to invoke a typical function?
- It can be the number of arguments defined in function definition.
- Can it be dynamic and any number of parameters ?
- *The function can be called by passing any number of arguments…and any type of arguments!!!*
- Reason the JavaScript code is interpreted and NOT compiled and then run..NOT as in typical programming languages..

Function dynamics

Because of the dynamic and interpreted nature, the java script functions can be called with

- ✓any types of arguments
- ✓any number of arguments
- ✓any order of arguments
- ✓return any type of values to the caller.

How to read the number of arguments ?

- The java Script engine supports built-in useful properties for each function that can be utilized inside the function.
- The **arguments[ ]** is function property that represents the array of arguments currently passed to the function by the caller. Inside the function you can access it by using index and length.
- i.e num = **arguments[0];**
- **name= arguments[1]** and so on
- numParameters = **arguments. Length;**

Lets look at an example

- **Regular sum function**
- function Sum(num1, num2)
- {
- var answer = num1 + num2;
- alert ("The Sum in Two argument function Is " + answer);
- }

Dynamic Sum function..

- **The function summing all the number of arguments passed by the caller**
- function Sum ()

```
{
 var len = Sum.arguments.length;
 var ans = 0;
 for (i=0; i<len; i++){
    ans += Sum.arguments[i];
    alert ("\nThe Sum in of args in function Is " + ans);
}
```

Function Call..

- **Now identify which function gets invoked..**
- <H1>Function To Add the Numbers</H1>
- <SCRIPT LANGUAGE = "JavaScript">

  Sum (1,2);

  Sum (10,3,30)

  Sum (10,34,2,4)

  Sum(23,”Hello”,true,null)

  Sum();
- </SCRIPT>

Result..

- The Sum function last loaded by the interpreter gets called for all the Sum calls..
- That means the second function loaded by the interpreter overwrites the earlier loaded function by the same name!
- The order of functions/variables is important in the **Interpreted World.**

Local and Global Variables

- Function bodies can have local and global variables.
- Local variables defined inside the functions are treated as locals.
- To persist values across function invocations generally we use global variables and manipulate across the functions/calls.

Variable Scope and Context

- When a function is invoked, the interpreter creates a new local context for the duration of its execution.
- All variables declared inside the function (including its arguments) exist only within this context.
- When the function returns, the context is destroyed
- So, if you wish to preserve a value across multiple function calls, you need to declare a global variable.

Define Global variable

- **Example** : function to calculate the sum recursively..
- var totalSum = 0; //global variable
- function getSum(x, y)
  {
    totalSum = totalSum + x + y;
    return(totalSum);
  }

Global side effect

- Are you going to populate the global space with any number of variables and functions for the JavaScript program ?
- What if another script/block for the same page re-defines the same variable/function and assigns different values ?

Function as a Container

- Every function can have its own properties and values.
- Instead of populating directly the global space with number of variables, define/add variables as property of functions which will be identified as function properties in global space.
- Each function will have a different name and each function properties will be identified by the container function names as unique values.
- The function object is referred as namespace for storage of function properties.

Add custom properties

- **Example** : function to calculate the sum recursively..
- ***totalSum = 101; //Global variable***
- //variable defined in getSum context in the global scope
- **getSum.totalSum = 0; //function property**
- function getSum(x, y)
  {
  getSum.totalSum = getSum.totalSum + x + y;
  return(getSum.totalSum);
  }
- The ‘totalSum’ is variable added in global space as part of getSum function object.

Parameter passing

- JavaScript supports ‘pass by value’ for simple data types, so the values of the parameters passed to function remain unchanged regardless of what happens inside the called function.
- The other data types such as objects, arrays are passed by reference, so their properties values can be changed inside the functions and made visible outside the functions.

Input and Output

- Interacting with the user is typically achieved through the **window** object methods.
- The **alert()** method of **Window**, which displays its argument message in a dialog box that includes an OK button.
- **The confirm()** method, which displays its argument message in a dialog box with both OK and Cancel buttons and returns a Boolean value that indicates whether or not the information was confirmed.
- Confirm (“Are you sure you want to continue?");

Taking an input value

- **prompt()** method used to collect some data from the user.
- A prompt displays its argument message in a dialog box and allows the user to enter data into a text field and hit enter key to submit.
- var answer = prompt ("What is your favorite color?","");

Operators

- Mathematical Operators
- (assignment =), **+** (addition), – (subtraction or unary negation), ***** (multiplication), ***/*** (division), and **%** (modulus)
- Operator precedence as usual applies.
- The modulus(%), which returns the remainder of an integer division.

Exercise

- Identify all the contents of window and document object and print it.
- Use window methods to interact with user and display messages.
- Use function namespace to differentiate between the variables of same name and use them in global world.

JavaScript Objects

- Objects can hold any type of data and have some useful methods/functions by which certain tasks are carried out.
- The browser provides a number of built-in objects for java script functions to use.
- Data contained in an object are said to be *properties* of the object.{functions are also treated as properties of objects}

Built-in Objects

- The ***window***, the object represents the current graphic page window and supports interaction with the user.
- *The* ***document*** object represents the html components of page and represents the Document Object Model (DOM), as defined by the W3C and is property of the window object (**window.document**).
- The **navigator**, object represents current browser on client side and provides information about client browser type , version and browser configurations.
- Other objects such as string, math, array etc.
- These objects have predefined functions and properties.
- In addition custom objects can also be defined.

The window Object

- The window object is the parent object for loaded documents, we do not explicitly refer to the window object when referring to its properties or invoking its methods.
- For example, window.alert(“”) can be called by as alert(“”) directly.
- Window object supports properties and methods to access the status bar, display messages, take inputs from user, timer functions, opening other windows etc.

The window- properties

- name - The name of the window by which it is identified
- self - This identifies the current window being referenced.
- Top - It refers to topmost parent window.
- parent - It refers to a window which contains another window.

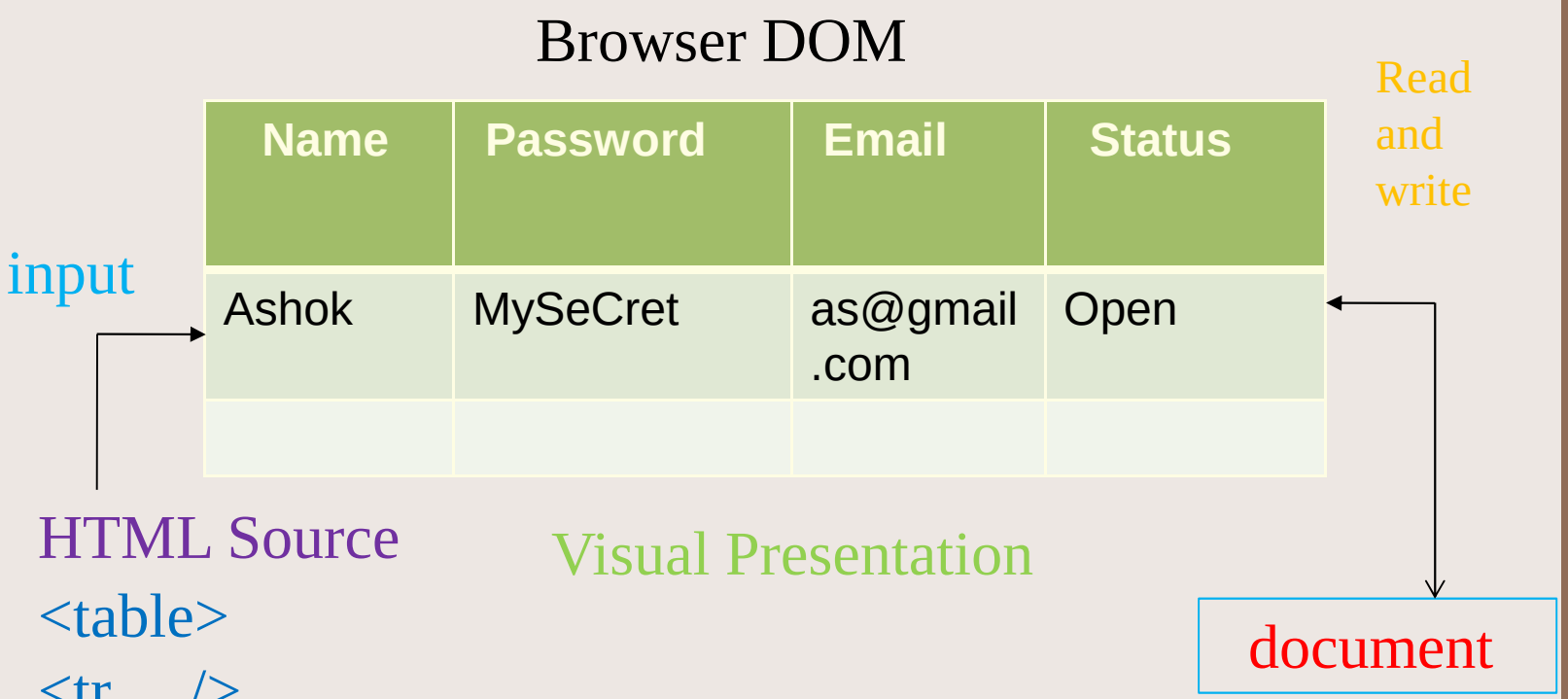
The window functions

- The window object provides the other methods are alert(), confirm(), open(), close(), prompt(), and setTimeout() methods.
- The window object also supports methods o start a new window and interaction with other existing windows, dialogs etc.

More window functions

- alert(message) - Displays an alert dialog box.
- prompt(message, default_text) -Displays a prompt with message.
- confirm(message) -Displays a confirm message box and returns user confirmation values.
- blur() - Removes focus from a window.
- focus() - Gives focus to a window.
- close() - Closes the specified window.
- open(url, name) - Opens the specified URL in a new window with new name specified by the window name.

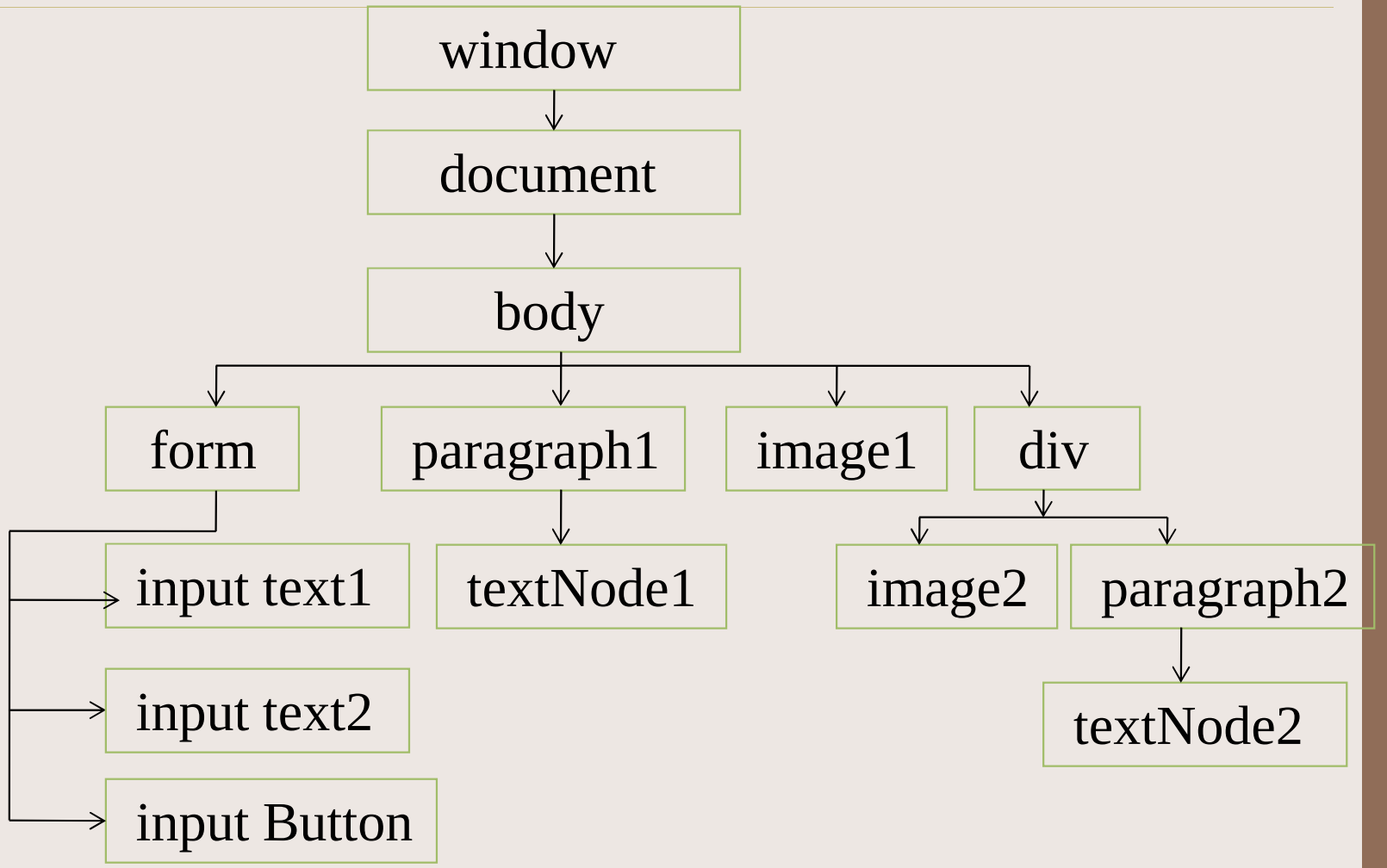
DOM Access


Browser DOM
Name
Password
Email
Status
Ashok
MySeCret
as@gmail
.com
Open
Read
and
write
input
HTML Source
<table>
<tr…./>
<.table>
Visual Presentation
document

The Document object

- The document object contains a tree that represents the entire html components of the current web page.
- The document is contained in window object.
- The document object follows the DOM specifications for html/xml structure.

DOM Tree Structure



The document is DOM

- The document object provides the properties and methods to work with numerous aspects of the current document, including information about anchors, forms, links, the title, the current location and URL, and the current colors.
- The document object supports variety of functions to access, update, delete, create new elements /attributes/child/text/ui nodes in current page.

Document Properties

- forms[] – An array of form objects, one for each <FORM> and </FORM> in the document. The number of forms in the document is determined by forms.length.
- images[] – An array of image objects, one for each embedded image. The number of images can be determined using images.length statement.
- links[] – An array of link objects, one for each hypertext link (<A HREF>) in the document. The number of link objects can be determined using links.length statement.

Document link properties

For visual settings of the hyperlinks following properties are used

linkColor – A string that specifies the colour of the unvisited links.

alinkColor–A string that specifies the color of a link while it is being activated.

vlinkColor – A string that specifies the color of visited links.

bgColor – A string that specifies the background color of the document.

fgColor – A string that specifies the text color of the document.

DOM retrieval functions

The functions to retrieve elements from current DOM structure.

- ✓getElementbyId(idVal): Returns unique reference to element defined by the ‘idVal’ value.
- ✓getElementsbyName(nameVal): Returns array of elements reference which have the name value as ‘nameVal’
- ✓getElementsbyTagName(nameTag): Returns array of elements reference which have the type tag value as ‘nameTag’ i.e ‘input’,’form’,’image’ etc.

Direct Access

- ✓ Additionally the individual elements can also be accessed by their names in current document hierarchy.
- ✓ I.e. document.myForm.userName.value: reads the text from ‘userName ‘ input element contained within a form ‘myForm’ in current DOM.

DOM Manipulation…

- To add new elements createElement, appendChild addAttribute functions.
- To read/update the attribute values getAttribute and setAttribute functions are supported.
- These functions combined with css styles and elements produce dynamic graphics and effects.

Exercise

- Utilize window functions to open a new window, start/stop the timer, display message on the status bar and identify page properties.
- Utilize the document functions to identify the parts of current web page.
- Use document properties to manipulate the contents of web page.

JavaScript objects

- ***String*** object
- ***Array*** object
- ***Date*** object
- ***Boolean*** object
- ***Function*** object
- ***Math*** object
- ***Global*** object
- **Object** object

Composite Types

- *The composite types* are made up of combination of different data type elements in one unit.
- A composite type can contain not only strings, numbers, Booleans, undefined values, and null values, but even other composite types.
- JavaScript supports three composite types:
  - objects
  - arrays
  - functions.
- *[The arrays and functions are really just special kinds of objects]*

Arrays

- An *array* is an ordered set of values grouped together under a single identifier.
- There are different ways to create arrays, but the simplest is to define it like a standard identifier and then just group the values within brackets
- var myArray = [1,5,68,3];
- Arrays can also contain arbitrary data items with different data types due to interpreted nature
- var myArray = ["Thomas", true, 3, -47.6, "x"];

Array Instance

- To create the array object use the keyword **new** to invoke the **Array** object's constructor,
- var myArray = new Array();
- var myArray = new Array(4);
- var myArray = new Array(1,5,"Thomas", true);

Elements of Array

- To reference a particular element of the array, provide an index value within brackets
- var myArray = new Array(1,5,"Thomas", true);
- var x = myArray[2];
- var y = myArray[0];
- arrays are actually objects and have a variety of properties and methods that can be used to manipulate them.

Array functions

toString(): Converts an array to a string. The array elements are separated by commas.

join(separator): Joins array elements together but separated by the separator string. The function returns the output as string.

sort(): Sorts the contents of an array based upon the ascii value of each character.

reverse(): Reverses the contents of an array. The last element becomes first and the first element becomes the last.

The String Object

Strings are declared using String() constructor

var s = new String();

var s1=new String(“hello”);

toLowerCase(): Converts the entire string to lowercase.

toUpperCase(): Converts the entire string to uppercase

String functions

substring(index): Returns a sub-string from a string beginning at index.

substring(index1, index2): Returns a sub-string from a string beginning at index1 and ending at index2.

More String Methods

split(separator)
Separates the entire string into an array of sub strings based upon the separator (de-limiter) character.
charAt(index)
Returns the character at the specified index.
charCodeAt(index)
Returns the character code (UNICODE) of the character at the specified index.
indexOf(pattern)
Returns the index of first occurrence of the substring considering the initial index at zero(0).

Date Object

Dates are declared using Date() constructor

```
var rightNow = new Date();
var birthDay = new Date(“March 24, 1970”);
var someDate = new Date(1970, 2, 24);
var s = new Date(1970, 2, 24,15,0,0);
var someDate = new Date(1970, 2, 24,15,0,0,250);
```

Date object functions

getYear()
Returns the year from the Date object.
setYear()
Sets the year for the Date object.
getFullYear()
Returns the year in four digit format from the Date object.
setFullYear()
Sets the year in four digit format for the Date object.
getTime()
Returns the time in milliseconds since midnight January 1,1970 from the Date object.

Math Object methods

Methods of Math object :

abs(x): returns the absolute value of x
cos(x) : returns the cosine of x
exp(x) : returns the exponential of x
log(x) : returns the natural logarithm of x
max(x,y) : returns the maximum of x and y
min(x,y) : returns the minimum of x and y

Math Object functions

Methods of Math object:

random()
This method returns random number between 0 and 1

round(x)
This method returns x rounded to the closest integer

sin(x)
This method returns the sine of x

sqrt(x)
This method returns the square root of x

tan(x)
This method returns the tangent of x

pow(x,y)
This method returns x to power y

Navigator Properties

- appVersion –version information of the browser.
- appCodeName – the code name of the browser.
- appName – the name of the browser.
- userAgent – the user-agent header sent in the HTTP protocol from the browser to the web server.

Navigator Properties

- Plugins – array of all the plug-ins installed in the current browser.
- mimeTypes – array of all mime types currently supported by the current browser.

Events in JavaScript

- Events are triggers that start some action.
- Functions are assigned to be invoked whenever any such trigger occurs i.e. click of a button.
- This function is referred as event listener.
- Events can be triggered by the user
- Events can be triggered by the system
- Events can be triggered by another event

Event Processing Model

- **Bubbled Event**: The event is bubbled/propagated to parent hierarchy /up to the document parent from the current source.
- **Captured Event :** The event is first captured by top parent(i.e. the document) and then is passed to the individual child elements in hierarchy.
- The Internet Explorer supports event bubbling model.
- The Chrome, Safari, Opera and FireFox supports both event models.

Bubbled Events

Most of the events are bubbled to parent..

- **onactivate**
- **onclick**
- **oncontextmenu**
- **ondblclick**
- **ondrag**
- **ondrop**
- **onkeydown**
- **onkeyup**
- **Onmouseup**

Non-Bubbled Events

Following events are not bubbled to parent

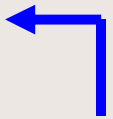
- **onload**
- **onfocus**
- **onblur**
- **onchange**
- **onmouseenter**
- **onreset**
- **onresize**
- **onscroll**
- **onselect**
- **onselectionchange**
- **Onsubmit**

Stop Bubbling the Event

- Event object has methods and properties to stop event bubbling
- To stop events from propagating up the hierarchy by setting the **cancelBubble** property to ***true*** of the **Event** object. This property is ***false*** by default.
- If the event is cancelable, setting the **event.cancelBubble** prevents the event from propagating.
- Same effect can be achieved by calling ***stopPropagation()*** on event object,but not supported by IE Browser.

Assign the Event Listener Code


```
<HTML>
<HEAD>
<TITLE> JavaScript Events </TITLE>
<SCRIPT LANGUAGE = “JavaScript”>
num=0
</SCRIPT></HEAD>
<BODY>
<A  HREF=“Hello.html”  OnMouseOver  =  ‘++num;  alert(“Number  :
”+num)’> Place Your Mouse Cursor Over This </A>
</BODY></HTML>
```



This event-handler is called when mouseOver event is triggered.

For event-handler any JavaScript Code/function, separated by semicolon, can be added but statements must be enclosed within single(‘) or double(“) quotes.

Event Listener Function

```
<HTML>
<HEAD>
<TITLE> JavaScript Events </TITLE>
<SCRIPT LANGUAGE = “JavaScript”>
num=0
function handler()
 {
  num++
  alert(“Number : “ + num)
}
</SCRIPT></HEAD>
<BODY>
<A HREF=“Hello.html” OnMouseOver = ‘handler()’> Place your mouse
cursor over this </A>
</BODY></HTML>
```

Events for Html Elements

- Events are used to make user interactions possible with the application.
- Each html element fires some common events as well as specialized events such as <body onload=…>.
- Common events such as onclick, onblur, onmouseover,onkKeyDown etc.
- These events trigger the listener functions.
- Event functions are called as callback functions(called by the system and NOT by the application).

Override the event behavior

- The default behaviour can be prevented by returning false on the event ort in the vent listener function .
- This is useful in client side validation functions to prevent data submissions to server if validation fails.
- <A HREF="" onClick='return false' onMouseOver='func("A")' onMouseOut = ‘func()’>A</A>

  <A HREF="" onClick='return false' onMouseOver='func("B")' onMouseOut = ‘func()’>B</A>
- *Here the click event for the hyper link is prevented to load the linked page/section*

Prevent the default behavior

✓When an event listener function returns false, the further event listeners for the current source are blocked

✓ The same effect can be achieved by calling function ***preventDefault*** on current event object inside the event listener function, but not supported by IE Browser

Custom Objects in JavaScript

What is an Object?

- An object is an unordered collection of data including primitive types, functions and even other objects.
  E.g Window, String etc.
- Types of JavaScript Objects
  - Built-in – Array, Math etc.
  - Browser – Window, Document etc.
  - User-Defined – Custom Objects

Custom Objects

- One of the more powerful capabilities of JavaScript is its ability to create custom object instances and manipulate them.
- Custom-made objects can include both properties (data), and methods (functions), which can be applied to the data stored in the object.
- Sounds complicated? If you are not familiar with object oriented programming you can think of objects as structures, which are common in C/VB.

The constructor function

- In order to create a custom object we must first define a constructor function for it (it will be called to create the instance of the current object):
- **The function constructor:**
  - Defines the properties and methods for the object instance created from it.
  - May set values to properties.
  - May contain function parameters, which can be used to set values to properties i.e parameterized constructor:

The Constructor

```
function personClass(param1){
        this.firstName = “Vijay” ;
        this.age = param1 ;
        this.gender=male ;
       }
```

Custom Object creation

- In order to instantiate a new object :
  - Define a variable to refer to the object.
  - Use the operator **new** with the function constructor defined (and supply arguments to it if required).

Example:

```
// Creating two objects:
var obj1 = new personClass(20) ;
var obj2 = new personClass(5.5) ;
// Using the objects:
obj1.age = obj2.age + 14.5 ;
```

The object instance

- When this constructor function is called with the **new** operator, an an instance of **personClass** object is created**,** which has three properties – **firstName** initialized to "**`Vijay`**", **age** initialized to the value of par1, and **gender** a value.
- The 'this' inside function represents the current instance.
- The function constructor identifies the object instance it creates as ***`this`***: Therefore, if **`this.p`** is used in the class definition:
  - A property named **p** will be created for the object, if it does not yet exist.
- The function constructor can include number of functions as well.

Shared values across instances

- Each instance is a separate copy of the object and has its own property values. i.e age in previous example
- To share the common property values across all the instances *prototype* keyword is used.
- The prototype is the shared memory area available for every function constructor and passed as a property for every function instance as shared area.
- The prototype works as static.

Function own values

- The function object can have its own property values which are defined within function object (namespace) which are not available to the instances.
- These properties acts as private properties and is only accessible though the function object name.

Instance properties

- Every object has three types of properties
  - Instance own values
  - Properties defined in function namespace(private)
  - Properties stored in shared prototype.

Custom object functions

- function personClass(){
  this.firstName="Vijay";
  this.age= 21;
  this.*earnMoney* = *work*;//assign predefined function
  this.*sayHello = function(message)*
  {
  alert("Hi "+message);
  };//assign anonymous function
  }

Link the external function

- A function be defined outside the constructor like any other function:

  function work(){

  …body of the function……

  }

  In order to add methods to a object, it needs to declare a regular property member (its name will be the method name) and assign it a function name (without parentheses, since it is not a call! Or assign a function in an an anonymous way)

Object Oriented Java Script?

- The Java Script code seems to be procedural code…
- Can it support the Object Oriented features such as inheritance, polymorphism, method overriding , encapsulation etc ?
- Given the dynamic nature of Java Script language, it is possible to create such scenarios at runtime!
- The custom library '**Prototype**' supports manipulating the dynamic nature of Java Script objects.

Applying inheritance

- When we say one object inherits another object what gets added in the child object structure?
- The parent object properties with their values and functions..
- The same we can create in run time object structures by injecting properties and function references.
- Two options available
  - Classical inheritance(Explicitly copy the values)
  - Prototype inheritance(Share the prototype)

Multiple inheritance

- The interpreted nature does not allow to have multiple inheritance from different base objects.
- In classical inheritance .. any properties can be inherited.

Super Base

- Can we have a super base to every java script object ? So that number of common functions can be available to every object.
- The super base is..Object.

Dynamic Prototyping

- Consider a situation : number instances of an object are running and one more new function added into the object structure, will all the instances get the new function available..?
- Conventional compiler driven logic mandates rebuilding and reloading…
- In Java script prototype is shared across all the instances. Hence newly added functions in prototype will be available immediately to all the instances.

Extending built-in objects?

- Can we extend the built-in objects and override the functions ?
- Can we add new functions into the built-in objects ?
- For example the array object doesn't provide for array bounds checking so we can inject another function reference that will check for bounds before displaying the contents.
- The same way we can override the write function of document object or override the alert function of window object.

Custom JavaScript libraries

- The support for defining and using custom object opens the vast opportunities to companies and professionals to create custom java Script objects for varies functionalities.
- The users just need to include these custom .js files in their pages and call the respective functions on the objects by creating and initializing them.
- Hundreds of such popular libraries and frameworks are available for java Script!
- YUI,jQuery,Ext-Js, Adobe Spry ,Prototype are customized java script libraries

JavaScript to Browser

- Do you have to write the java script code compatible to every web browser ?
- The JavaScript libraries/frameworks translate it.
  - Dojo : supports widgets and layouts
  - Prototype : supports manipulating the dynamic nature of Java Script objects.
  - ExtJs : supports layouts, controls and MVC extensions.
  - JQuery : supports selectors, ajax and events.
  - Rico : supports selectors, ajax and events.
  - Microsoft Atlas
  - Yahoo UI Library
  - Google web Toolkit : Java to Java script translations

Code compression

- The following frameworks support Java Script code minification and obfuscation.
- Yahoo compressor
- Google Closure Compiler
- JavaScript Code obfuscator

Testing the code

The following frameworks/libraries support running the test cases and displaying/calculating the test results.

- Qunit : unit Testing
- YUI Test : unit Testing
- JS Test Driver : Integration testing
- Jasmine/Karma : BDD Testing
- Selenium : Integration testing

Java script code quality

Static code quality analyzers for java Script

- JsLint
- jsHint
- Google Code Analyzer

Thanks to you!