JavaScript

Lakshman M.N.
Technology Evangelist
Lakshman.mn@gmail.com

What is JavaScript

- Scripting language developed by Netscape.
- Used with HTML.
- Helps create interactive Web pages.
- JavaScript is not Java.

JavaScript, the Core Language

- Keywords, statement syntax, and grammar
- Rules for expressions, variables, and literals
- Underlying object model
- Predefined objects and functions

Into JavaScript

JavaScript Syntax

Using the SCRIPT Tag

The <SCRIPT> tag is an extension to HTML that can enclose any number of JavaScript statements

<SCRIPT>

JavaScript statements...

</SCRIPT>

Embedding JavaScript into a HTML-page

```
<html>
<head>
 <script language="JavaScript">
    document.write("This is JavaScript!")
  </script>
</head>
<body>
<br>
This is a normal HTML document.
<br>
</body>
</html>
```

CheckOut

Non-JavaScript browsers

```
<html>
<head>
<script language="JavaScript">
  <!-- hide from old browsers
    code goes here
  // -->
  </script>
</head>
<body>
This is a normal HTML document.
<br>
</body>
</html>
```

Syntax and Conventions

- Case-Sensitivity
- Semi-colons
- Braces
- Comments

Summary

- Script placed within `<SCRIPT>` tag.
- Javascript is case-sensitive.
- All Javascript statements end with semi-colon.
- Comments in Javascript.

Basic programming constructs

Control flow statements

Variables

- Declaration using **`var`** keyword.

```
var x;
```

- Default datatype is **variant.**

```
x = 10;
x = "ten";
```

JavaScript Operators

Computational

- Unary negation (-)
- Increment (++)
- Decrement (--)
- Multiplication (*)
- Division (/)
- Addition (+)
- Subtraction (-)

JavaScript Operators

Logical

- Logical NOT (!)
- Less than (<)
- Greater than (>)
- Less than or Equal to (<=)
- Greater than or Equal to (>=)
- Equality (==)
- Inequality (!=)

JavaScript Operators

Assignment

- Assign (=)
- Addition (+=)
- Subtraction (-=)
- Multiplication (*=)
- Division (/=)
- Modulo Arithmetic (%=)

Functions in JavaScript

- Programmer defined functions
- Built in Functions

Functions

- The `function` **`keyword`**
- A function name
- A comma-separated list of arguments to the function in parentheses
- The statements in the function in curly braces.

Functions

```
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!-- Hide script from old browsers
function square(number) {
   return number * number;
}
alert("The function returned ", square(5),
".");
// End script hiding from old browsers -->
</SCRIPT></HEAD>
<BODY>
<P> All done.
</BODY>
```

CheckOut

Summary

- Datatype - **Variant**
- Operators (Arithmetic, Logical, Assignment)
- Functions

Objects, Events and DOM

- Objects
- The "new" operator
- Document Object Model
- Arrays
- Events

Objects

- Collection of variables (parameters) and functions (methods).
- Built-in Objects
  - String
  - Math
  - Date
  - Array
  - Object
  - RegExp

The "new" operator

- Objects need to be created.
- The generic syntax is

```
o = new Object ( );
o.parameter = value;
o.method ( );
```

Array Object

- Collection of data
- Each item referred by index value.
  - Index starts from 0

```
var arr = new Array ();
a[0] = 1;
a[1] = 2;
```

OR

```
var arr = new Array("1","2");
```

Date Object

- An object used to work with dates and times
- Methods
  - getHours ( )
  - getSeconds ( )
  - getMonth ( )
  - getDate ( )
  - getYear ( )

Math Object

- An object that helps perform Mathematical tasks.
- Methods
  - cos ( )
  - sin ( )
  - tan ( )
- Properties
  - PI

String Object

- An object used to store and manipulate text.
- Methods
  - charAt (*index*)
  - indexOf (*substr, startindex*)
  - toUpperCase ( )
  - toLowerCase ( )
- Properties
  - length

Custom Objects

- Creating an object requires
  - Declaration of the object by using the object function
  - Instantiation by using the "new" keyword

```
var manager;
manager = new Object();
manager.empid = "E101";
```

Custom Objects…Constructors

```
function employee(empid,empname,dept)
{
  this.empid = empid
  this.empname = empname
  this.dept = dept
}
var manager = new
  employee('E101','Lakshman','Tech');
```

RegExp Object

- An object that describes a pattern of characters
- Used to perform powerful "pattern-matching" and search-and-replace" functions on text.

```
var reEx = new RegExp("pattern");
```

Or

```
var reEx = /pattern/;
```

Regular Expression Methods

- **`exec()`**
  - Takes one argument, a string
  - Checks whether the string contains one or more matches of the pattern specified
  - Returns a result array with the starting points of the matches
  - Returns null if no match is found

Regular Expression Methods

- **`test()`**
  - Takes a string argument
  - Checks whether the string contains a match of the pattern specified by the regular expression
  - Returns true if it does and false if not
  - Useful in form validation

Flags

- Flags appearing after the end slash modify the behavior of a regular expression
- **`i`**
  - Makes the regular expression case insensitive
  - Matches all lowercase and uppercase usages
- **`g`**
  - Specifies a global match
  - All matches of the specified pattern are returned

String Methods

- Several String methods use regular expressions
- **`search()`**
  - Expects a regular expression as an argument
  - Returns the index of the first character of the substring matching the expression
  - If no match is found, returns -1
  - **`myStr.search(/cat/);`**

String Methods

- **`split()`**
  - Expects a regular expression as an argument
  - Uses the regular expression as a delimiter to split the string into an array
  - `myStr.split(/he/gi);`
- **`replace()`**
  - Accepts two arguments, a regular expression and the string

String Methods

- **`match()`**
  - Accepts a regular expression as an argument
  - Returns the substring that matches the pattern
  - `myStr.match(/he/gi);`

Regular Expression Syntax

- Start and End
  - **^** at the beginning of a regular expression indicates that the string being searched must start with this pattern
    - **^res** can be found in "rest" not in "stress"
  - **$** at the end of a regular expression indicates that the string being searched must end with this pattern
    - **at$** can be found in "fat" not in "ate"

Occurrences

- **?** Indicates zero or one appearances of the preceding character.
  - `fee?` could be `fed` or `fee` or `feed` etc
- **+** indicates one or more occurrences of the preceding character in the pattern
  - `fa+` could be `fast` or `faast` or `fat` not `fiat`
- ***** indicates zero or more occurrences of the preceding character in the pattern
  - `fe*` could be `fend` or `feet` or `find`

Common Characters

- `.` represents any character except a newline
- `\d` represents any digit
- `\D` represents any character except a digit
- `\w` represents any word character
- `\W` represents any character except a word character
- `\s` represents any whitespace character
- `\S` represents any character except a whitespace character

Prototype

Overview

- JavaScript allows addition of custom properties to both prebuilt and custom objects.

```
var img = new Image();
img.size="18k";

function circle(){
}
var smallcircle = new circle();
smallcircle.pi = 3.14159;
```

- Custom properties added in this manner exist only for that instance of the object.

Prototype Object

- The prototype object is used to add a custom property to an object that is reflected on all instances of it.
- The keyword "prototype" is referenced on the object before adding the custom property to it
- This property is instantly attached to all instances of the object.

Prototype…

```
function circle(){
}
circle.prototype.pi = 3.14159;
```

- All instances of the **`circle()`** now have the **`pi`** property prebuilt
- JavaScript permits the "prototype" of prebuilt objects that are created with the **`new`** keyword

Prototype…

- Prototype object helps add a custom method to an object.
- This method is reflected on all instances of the object.

```
function circle(){
}
circle.prototype.pi = 3.14159
function dispmessage(){
alert(this.pi);
}
circle.prototype.alertpi = dispmessage
```

Extending pre-built Objects

- Prototype can be used on pre-built JavaScript objects created with the **new** keyword.

```
function writeCharCode(){
    var tmpCode="";
    for(i=0;i<this.length;i++){
        tmpCode += this.charCodeAt(i);
    }
     return(tmpCode);
}
String.prototype.charCode = writeCharCode;
```

Inheritance using Prototype

- JavaScript is a class-free language and uses prototypal inheritance instead of classic inheritance.
- An object inherits from another object.
- When an object **rabbit** inherits from another object **animal** in JavaScript, it is indicated by

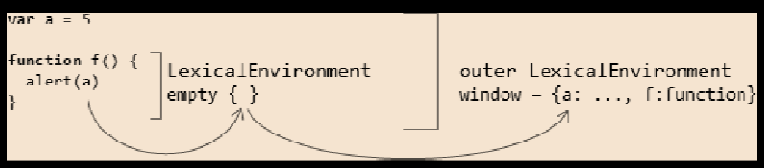
```
rabbit.prototype = new animal();
```

Closures

Access to outer variables

- If a variable that is accessed is not local the interpreter finds it in the outer `LexicalEnvironment` object

```
var a = 5
function f() {
  alert(a)
}
```



Nested functions

- Functions can be nested one inside another.

```
var a=1
function f() {
  function g() {
     alert(a)
   }
    return g
}
var func = f()
func()  //returns 1
```

Closure

- A closure is the local variables of a function kept alive after the function has returned.

```
function User(name) {
  this.say = function(phrase) {
    alert(name + " says: " + phrase)
  }
}
var user = new User("Lakshman")
```

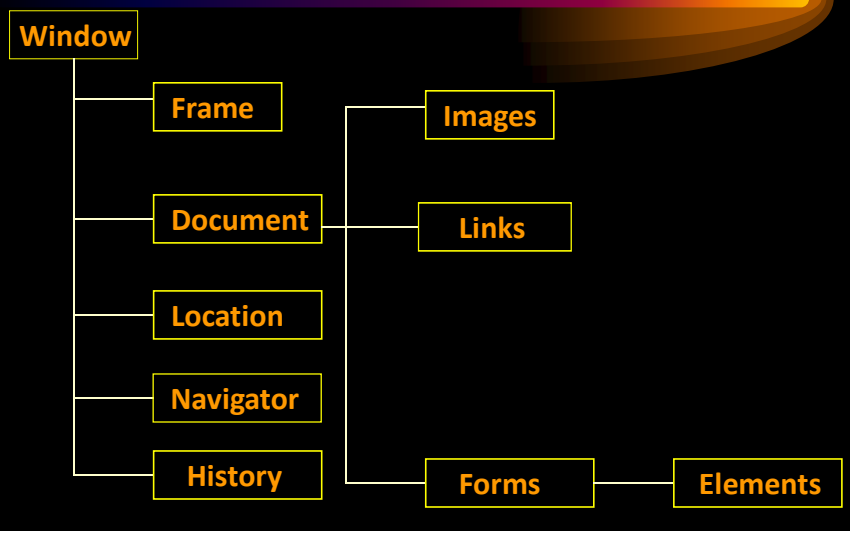
Closure…

- The inner function keeps a reference to the outer Environment.
- The inner function may access variables from it any time even if the outer function is finished.
- The browser keeps the Environment and all it's properties(variables) in memory until there is an inner function which references it.
- This is called a *closure*.

Document Object Model (DOM)

- Series of objects provided by browser.
- Hierarchical structure.
- Consists of embedded objects also.

DOM Level 1 (BOM)



Window

- Represents the browser window.
- Events
  - load
  - unload
- Methods
  - open()
  - close()
- Properties
  - innerHeight
  - innerWidth
  - status

Location

- Contains information about the current URL
- Properties
  - href
  - hash
  - port
  - protocol
  - hostname
- Methods
  - reload()

Navigator

- This object contains information about the browser.
- Properties
  - appName
  - appVersion
  - userAgent
- Methods
  - javaEnabled()

History

- Contains the URLs visited by the user.
- Properties
  - length
- Methods
  - back()
  - forward()
  - go()

Document

- Represents the HTML document loaded into the browser
- Properties
  - lastModified
  - doctype
- Methods
  - open()
  - close()
  - write()

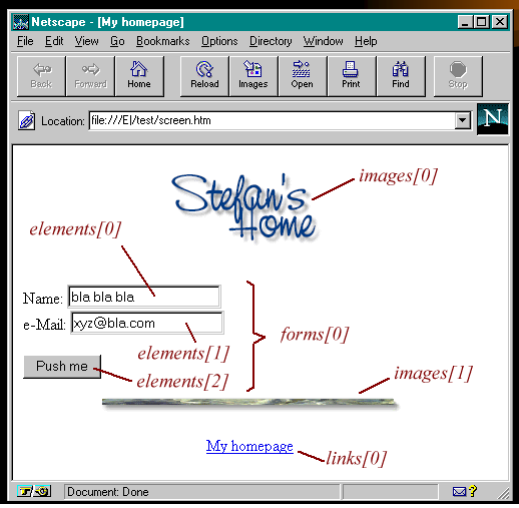
Image

- This object represents an embedded image.
- For every <img> tag in the HTML document an image object is created.
- Properties
  - src
  - alt

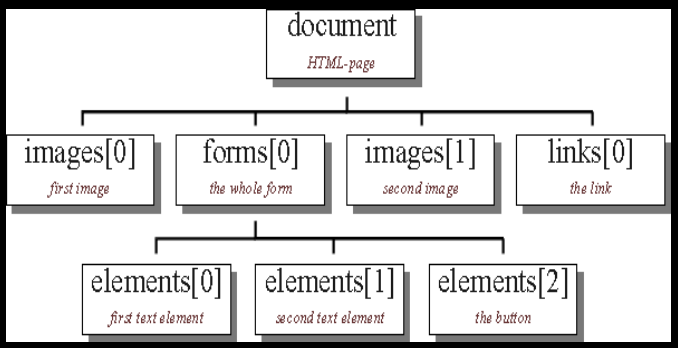
Link (anchor)

- This object represents an HTML hyperlink
- Every <a> results in the creation of an anchor object.
- Properties
  - href
  - target

Browser Object Model Contd..



JavaScript hierarchy Contd…



Events

- User actions in JavaScript are called events.
- The browser constantly monitors document elements capable of reacting to user actions.
- The event model connects actions (events) that occur on the page with scripts

Event Order

- Consider an element placed within another and both have an onclick handler.
- A click on element2 causes a click event on both elements.
- Netscape and Microsoft decided to use different models for the order of the event capture.

Event Order Models

- Netscape said that the event on element1 takes place first.
  - This is called event *capturing*.
- Microsoft maintained that the event on element2 takes precedence.
  - This is called event *bubbling*.

Event Models

- In *Capturing*, event handler of element1 fires first, the event handler of element2 fires last.
- In *Bubbling*, the event handler of element2 fires first, the event handler of element1 fires last.

W3C Model

- Any event taking place in the W3C event model is
  - first captured until it reaches the target element
  - and then bubbles up again.
- The developer can choose whether to register an event handler in the capturing or in the bubbling phase

W3C Event Model

- The order is indicated through the **`addEventListener()`** method
- If its last argument is **`true`** the event handler is set for the *capturing* phase.
- If it is **`false`** the event handler is set for the *bubbling* phase.

Compatibility

- In the browsers that support the W3C DOM, a traditional event registration

```
element1.onclick = doTask;
```

is seen as a registration in the *bubbling phase*.

Switching it off

- Turn all *capturing* and *bubbling* off to keep functions from interfering with each other.

```
function doTask(e) {
  if (!e) var e = window.event;
  e.cancelBubble = true;
  if (e.stopPropagation)
   e.stopPropagation();
}
```

Events

- Triggers that invoke functions
  - onClick ( )

  Occurs when a button, radio button, ckeckbox or submit button is clicked.
  - onSubmit ( )

  This event occurs when the user submits the form.
  - onMouseOver ( )

  Occurs when the user positions the mouse over the current object. Checkout

Events Contd..

- onMouseOut ( )

Occurs when the mouse moves off a hyperlink or an object.
- onFocus ( )

Occurs when an object recieves focus.
- onChange ( )

Occurs when the value of an input element is changed.

CheckOut

Events Contd….

- onLoad ( )

It is triggered when the browser finishes loading a document or all the frame pages within a <FRAMESET>

- onUnLoad ( )

Occurs when you move to a new document.

DOM Level 2

- Every HTML document is loaded into memory as a tree of objects.
- Scripts can dynamically access and update the content, structure and style of the document.
- Changes made reflect in the in-memory tree and therefore the view.

DOM Level 2…

- Searching elements
  - getElementsByTagName()
  - getElementById()
- Changing the structure
  - createElement()
  - createTextNode()
  - setAttribute()
  - appendChild()
  - removeChild()
  - innerHTML

Summary

- Describe an Object
- Use the new operator
- Understand DOM
- Create and use arrays
- Use built-in events

Alerts, Prompts and Confirms

- Creating alerts
- Accepting input
- Creating confirms

Alert

- window.alert (*message*)

Displays a dialog box with the *message* and an 'OK' button to dismiss it.