Unit – IV: JavaScript

Copyright Guideline

Table of Contents

Introduction to Scripting

Scripting

Introduction

Web scripting languages are programming languages which support logic building to make web pages dynamic and interactive

Some web scripting languages : VBScript, JavaScript, Jscript and ECMA Script

Browser includes scripting interpreter

Choosing a scripting language for any web application is primarily based on :

- Browser compatibility
- Programmer familiarity

Scripts can be executed on client or server (JavaScript can be used with client or server, but mainly used for client side scripting)

JavaScript – Core Features

Core Features	
	An interpreted scripting language
	Embedded within HTML
	Minimal syntax - Easy to learn (C syntax and java OOC)
	Mainly used for client side scripting because it is supported by all the browsers
	Designed for programming user events
	Platform independent / Architecture neutral
	JavaScript is object based and action-oriented

Embedding JavaScript into HTML page

- <SCRIPT>.....</SCRIPT> tag
- TYPE - the scripting language used for writing scripts

EMBEDDED JavaScript

```
<SCRIPT TYPE="text/javascript">
        ----
                (JavaScript code goes here)
        ----
</SCRIPT>
```

EXTERNAL JavaScript

```
<SCRIPT TYPE="text/javascript"  src="External.js"></script>
```

- JavaScript can be enabled or disabled in browsers

Embedding JavaScript into HTML page - Example

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
      <head>
            <link rel="stylesheet" href="../css/style.css" type="text/css" />
            <script type="text/javascript" src="../js/scripting.js"></script>
            <title>Adding Supplier</title>

            <script type="text/javascript">

                  /* Validating Supplier Details */
                  function supplierDetailsValidation(){
                        var sname = document.supplierForm.supplierName.value;
                        var emailId = document.supplierForm.emailId.value;
                        var contactNo =
                                    document.supplierForm.contactNo.value;

                  //Checking if Supplier Name is empty
                        if(validateEmpty(sname)){
                              alert("Supplier name cannot be blank");
                              return false
                        }
```

External JavaScript

Embedded JavaScript

JavaScript – Statements & Comments

Statements & Comments

A semicolon ends a JavaScript statement

Comments

- Supports single line comments using //
  and multi line comments using /*…..*/

JavaScript is case sensitive

Statements & Comments - Example

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
      <head>
            <link rel="stylesheet" href="../css/style.css" type="text/css" />
            <script type="text/javascript" src="../js/scripting.js"></script>
            <title>Adding Supplier</title>

            <script type="text/javascript">

                  /* Validating Supplier Details */
                  function supplierDetailsValidation(){
                        var sname = document.supplierForm.supplierName.value;
                        var emailId = document.supplierForm.emailId.value;
                        var contactNo =
                                    document.supplierForm.contactNo.value;

                  //Checking if Supplier Name is empty
                        if(validateEmpty(sname)){
                              alert("Supplier name cannot be blank");
                              return false
                        }
```

Multiline Comment

Semicolon acting as the delimiter

Single line Comment

write & writeln methods

- These methods are used to display HTML output to the user.

write(string)	Writes one or more HTML expressions to a document in the specified window.
writeln(string)	Writes one or more HTML expressions to a document in the specified window and follows them with a new line character.

Form Object

- Each form in a document creates a form object.
- A document can have more than one form
- Form objects in a document are stored in a forms[ ] collection.
- The elements on a form are stored in an elements[ ] array.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
      <head>
            <link rel="stylesheet" href="../css/style.css" type="text/css" />
            <script type="text/javascript" src="../js/scripting.js"></script>
            <title>Adding Supplier</title>

            <script type="text/javascript">

                  /* Validating Supplier Details */
                  function supplierDetailsValidation(){
                        var sname = document.supplierForm.supplierName.value;
                        var emailId = document.supplierForm.emailId.value;
                        var contactNo =
                                    document.supplierForm.contactNo.value;
```

'supplierForm' is the form object

'supplierName' is the text object

'emailId' is the text object

Functions

Functions

- A function is a block of code that has a name. It is a way to organize the code
- JavaScript has in-built functions. User can define his own functions(User defined functions)
- A function has a name, arguments(optional) and function body (statements). Arguments are local to the function

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

- JavaScript functions are used to link actions on a web page with the JavaScript code

Eg : JavaScript function can be used to link the user action like clicking on the 'Submit' button with the code validating the various fields of the form

Note: Built-in Functions in JavaScript will be revisited later in this course

Variables and Data Types

Variables (1 of 2)

Variables

- Must start with a letter or an underscore and can have digits.
- Can be defined in 2 ways:
  - Using keyword 'var' and Without using keyword 'var'

Using keyword 'var' :

```
<script type= "text/javascript">
    var num1=10; // Global variable
    function demoOne() {
            var num2 = 20; // Local to the function
    }
    function demoTwo() {
            num1=30; //'num1' is accessible
            num2=40; //'num2' is NOT accessible
    }
</script>
```

Variables (2 of 2)

Variables

Without using keyword 'var' :

```
<script type= "text/javascript">
    num1=10; // Global variable
    function demoOne() {
            num2 = 20; // Global variable
    }
    function demoTwo() {
            num1=30; //'num1' is accessible
            num2=40; //'num2' is accessible
    }
</script>
```

Implicit Data Types

Implicit Data Types

Variables don't have explicit data types. The data type is automatically decided by the usage and hence implicit.

Implicit data types include:

- number
- string
- boolean
- object

Type Conversion

- Automatically converts between data types
- Eg:

```
val=123 ;          // Here 'val' is of number type
val="Hello";       // Here 'val' is of  string type
val=true ;         // Here 'val' is of boolean type
```

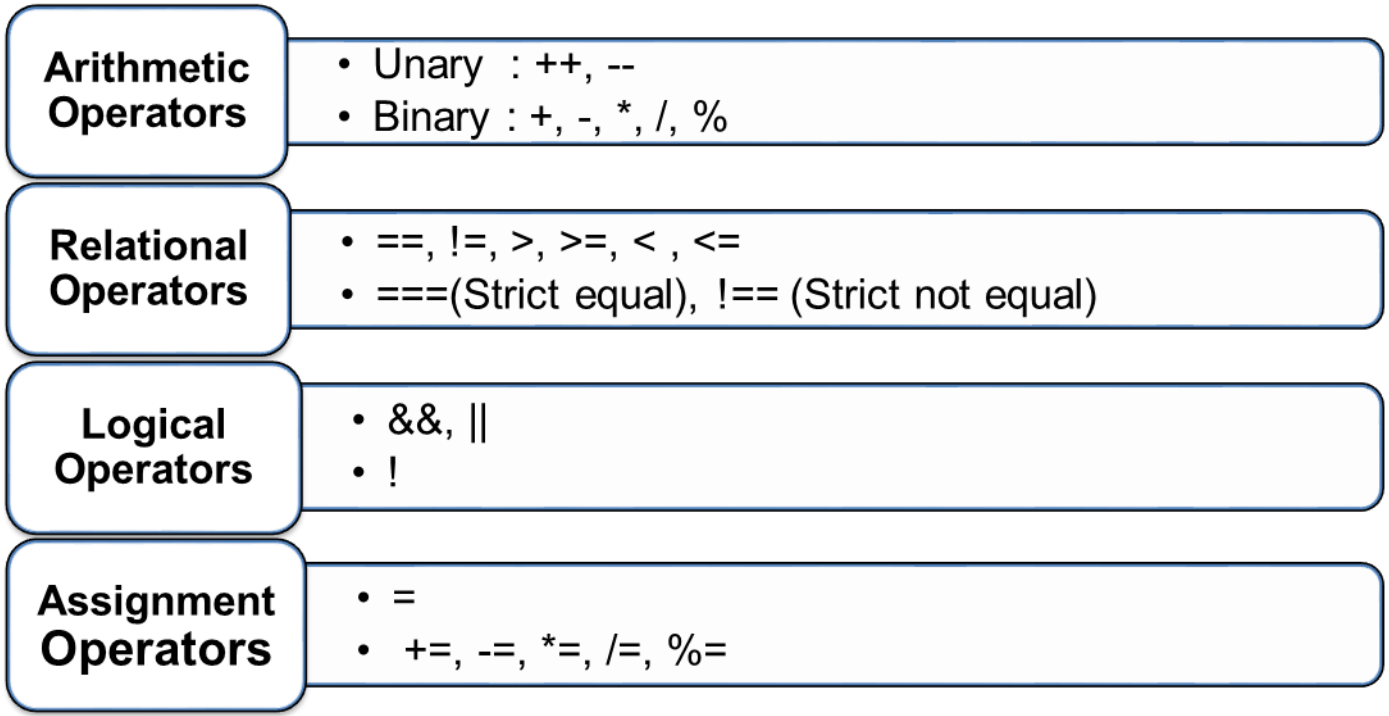
typeof & new

- **typeof**
  - Unary operator
  - Indicates the data type of the operand.
    - Eg: `val=123;  alert(typeof(val));` // Displays 'number'
      `val="Hello"; alert(typeof(val));` // Displays 'string'
      `val=true ; alert(typeof(val));` // Displays 'boolean'
- **new**
  - Used for instantiation of objects.

    Eg: `val = new Date( ) ; alert(typeof(val));` // Displays 'object'

Operators

Operators



Control Statements

Control statements - Conditional

- Control structure in JavaScript is as follows:

if	• Is used to conditionally execute a single block of code
if…else	• A block of code is executed if the test condition evaluates to a boolean true; else another block of code is executed
switch	• Switch statement tests an expression against the case options • Executes the statements associated with the first match and break may be used to quit from the switch statement

Control statements - Iteration (1 out of 2)

for loop

- Iterate through a block of statement for some particular range of values
- **Syntax :**

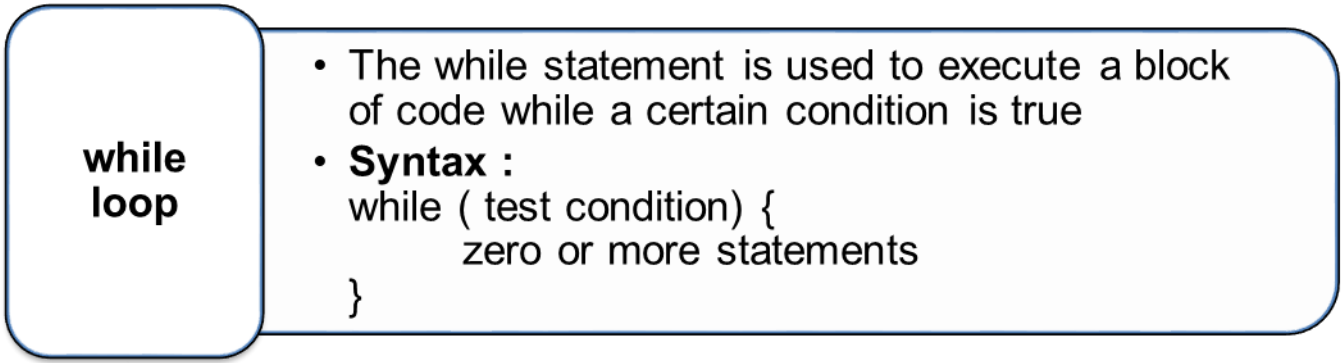
```
for(initialization ; condition ; increment/decrement) {
zero or more statements                                }
```

do while loop

- Block of statements is executed first and then condition is checked
- **Syntax :**

```
do {
        zero or more statements
}while ( test condition)
```

Control statements - Iteration (2 out of 2)



Expressions

Built-in functions – eval – Evaluating Expressions

- **Eval:** Evaluates an expression provided as a string argument
  - **Syntax : eval(str)**
  - Eg:

```
alert(eval("123"))                          //Displays 123
alert(eval(123))                            //Displays 123
alert(eval("Hello"));                       // Displays error
alert(eval(Hello));                         // Displays error
str= "Hello";  alert(eval("str"));          // Displays Hello
alert(eval(123+456))                        //Displays 579
str1= "Hello"; str2 = "World"
alert(eval("str1+str2"));                   // Displays HelloWorld
```

Built-in Functions

Built-in functions – parseInt /parseFloat

- **parseInt / parseFloat** : Parses the given string and returns a numeric value if the given string starts with number
  - **Syntax : parseInt(str)**
  - Eg:

```
str= 123;    alert(parseInt(str));          // Displays 123

str= “123”;  alert(parseInt(str));          // Displays 123

str= “abc”;  alert(parseInt(str));          // Displays NaN

str= “123abc”;  alert(parseInt(str));       // Displays 123

str= “abc123abc”;  alert(parseInt(str));    // Displays NaN
```

*NaN stands for **N**ot **a** **N**umber

Built-in functions - isNaN

- **isNaN:** Evaluates an argument to determine if it is “NaN” (Not a Number)
  - **Syntax : isNaN(str)**
  - Eg:

```
str= "123";  alert(isNaN(str));          // Displays false

str= 123;  alert(isNaN(str));            // Displays false

str= "abc";  alert(isNaN(str));          // Displays true

str= abc;  alert(isNaN(str));            // Displays error

str= "123abc";  alert(isNaN(str));       // Displays true

str= "abc123abc";  alert(isNaN(str));    // Displays true
```

Built-in functions - isFinite

- **isFinite:** Evaluates an argument to determine whether it is a finite number
  - **Syntax : isFinite(str)**
  - Eg:

```
str= "123";  alert(isFinite(str));          // Displays true
str= 123;  alert(isFinite(str));            // Displays true
str= "abc";  alert(isFinite(str));          // Displays false
str= abc;  alert(isFinite(str));            // Displays error
str= "123abc";  alert(isFinite(str));       // Displays false
str= "abc123abc";  alert(isFinite(str));    // Displays false
```

Built-in functions – Number & String

- **Number:** Functions let you convert an object to number
  - **Syntax : Number(str)**
  - Eg:

```
str= "123";  alert(Number(str));          // Displays 123
str= 123;  alert(Number(str));            // Displays 123
str= abc;  alert(Number(str));            // Displays error
str= "abc";  alert(Number(str));          // Displays NaN
str= "123abc";  alert(isFinite(str));     // Displays NaN
str= "abc123abc";  alert(isFinite(str));  // Displays NaN
```

- **String:** Functions let you convert an object to string
  - **Syntax : String(num)**

Dialog boxes (Window Object methods)

alert	• Takes in a string argument and displays an alert box • **Syntax :** alert(message)
prompt	• Displays a message and a data entry field • **Syntax :** prompt(message,[inputDefault])
confirm	• Serves as a technique for confirming user actions • **Syntax :** confirm(message)

JavaScript - Modes

- SCRIPT tag can be placed in HEAD or BODY tag
- Placing JavaScript in the HEAD tag ensures readability

Immediate	• Script gets executed as the page loads
Deferred	• Script gets executed based on user action

JavaScript – 3 Ways of Inclusion in HTML(1 of 3)

- 3 ways in which JavaScript can be included in HTML
  - Inline
  - Embedded
  - External file
- Inline JavaScript
  - Scripts are included inside the HTML tag

```
<html>
    <body onLoad="alert('document loaded')";>
        <h1>Inline JavaScript</h1>
        <input type="button" name="but1"
        value="click"
onClick="window.close()">
    </body>
</html>
```

JavaScript – 3 Ways of Inclusion in HTML (2 of 3)

- **Embedded**
  - Embedding JavaScript code into an html page is done using <script> tag
  - Embedded code is not accessible from other pages

```
<html>
    <head>    <script>
                function demo(){
                  document.write("Embedded Mode");
                }
    </script></head>
    <body> … </body>
</html>
```

JavaScript – 3 Ways of Inclusion in HTML (3 of 3)

- **External**
  - Can be achieved by using the SRC attribute of the <script> tag.
  - External Javascript file should have an extension .js
  - Should not use <script> tag inside the .js file

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

Client Vs. Server Scripting

Client Side Scripting	Server Side Scripting
Runs on the user's computer i.e. Browser interprets the script.	Runs on the Web server and sends the output to the browser in HTML format.
Source code is visible to the user. (Source code is downloaded to the client and executed in browser).	Source code is not visible to the user. Server side source is executed on server.
Used for client side validations and functionality for the user events.	Used for business logic and data access from the database. The pages are created dynamically.
Depends on the browser and version.	Do not depend on the client, any server side technology can be used.

Objects

Objects

- Variables are containers for data values. Objects are variables and can contain many values.
- The name:value pairs in JavaScript objects are called properties.
- Actions that can be performed on objects are called methods.
- Objects are of 3 types
  - Global objects like number, array, etc
  - User-defined objects like customer, vehicle etc
  - Browser defined objects like window, document etc

Object Creation

- **Using literal notation:** Values are written as name : value pairs separated by a colon

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

- **Using new operator with a constructor of pre-defined object**

```
var person = new Object();
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

- **Using new operator with a constructor of custom object**

```
function Person(first, last, age, eye) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eye;
}
```

```
var myFather = new Person("John", "Doe", 50, "blue");
var myMother = new Person("Sally", "Rally", 48, "green");
```

Object Invocation

1. **Syntax:** object.property
   - **Example:**

```
person.firstname + " is " + person.age + " years old.";
```

2. **Syntax:** object [property]
   - **Example:**

```
person["firstname"] + " is " + person["age"] + " years old.";
```

Method inside objects

- JavaScript Methods are actions that can be performed on objects
- It is a property containing function definition
- Methods are functions stored as object properties.
- **Object Creation:**

```
var person = {
    firstName: "John",
    lastName : "Doe",
    id       : 5566,
};
person.name = function() {
    return this.firstName + " " + this.lastName;
};
```

Method definition inside object
Property value : Function definition

- **Object Invocation**

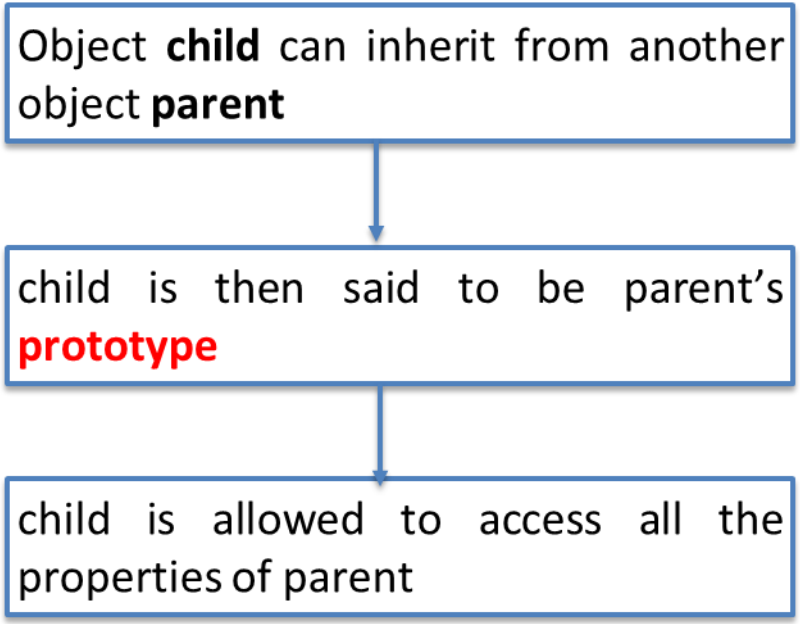
```
alert("My father is " + person.name());
```

Property Name : Function Name

Objects - Inheritance

- JavaScript supports inheritance by using the prototype property
- Prototype is a property of a function that establishes the inheritance between a derived and a parent class
- All JavaScript objects inherit properties and methods from a prototype
- Math objects inherit from Math.prototype. Array objects inherit from Array.prototype. Person objects inherit from Person.prototype
- The Object.prototype is on top of the prototype inheritance chain:
  Math objects, Person objects and Array objects inherit from Object.prototype.

Objects – Inheritance Concept(1/2)



Objects – Inheritance Concept(2/2)


```
// Initialize constructor functions
function Hero(name, level) {
  this.name = name;
  this.level = level;
}

function Warrior(name, level, weapon) {
  Hero.call(this, name, level);

  this.weapon = weapon;
}

function Healer(name, level, spell) {
  Hero.call(this, name, level);

  this.spell = spell;
}
```

```
// Link prototypes and add prototype methods
Warrior.prototype = Object.create(Hero.prototype);
Healer.prototype = Object.create(Hero.prototype);

Hero.prototype.greet = function () {
  return `${this.name} says hello.`;
}

Warrior.prototype.attack = function () {
  return `${this.name} attacks with the ${this.weapon}.`;
}

Healer.prototype.heal = function () {
  return `${this.name} casts ${this.spell}.`;
}

// Initialize individual character instances
const hero1 = new Warrior('Bjorn', 1, 'axe');
const hero2 = new Healer('Kanin', 1, 'cure');
```

Objects – Adding/deleting properties

- Properties can be added/deleted to/from individual objects or parent constructor
- Adding Property to parent constructor

```
function Person(first, last, age, eye) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eye;
}
Person.prototype.nationality = "English";
```

Add property nationality to individual object

- Deleting Properties

```
delete Person.firstname;
```

Delete a property firstname

The prototype chain

- Prototype Chain indicates the inheritance of any object.
- Property – "prototype" of each object defines this prototype chain
- Denoted as __proto__.
- To view the prototype chain use this __proto__ property of objects.
- Property look up happens across the prototype chain

```
child.__proto__ = parent
parent.__proto__ = object
object.__proto__ = null
```

Array

Array

- Array objects are used to store multiple values in a single variable.
- They store a fixed-sixe sequential collection of elements of the same type.

Syntax:

var odd_nums = new Array (1,3,5,7) or var odd_nums = [1.3.5.7]

- Array values can be accessed using index values like odd_nums[2] = 5

Array Properties

Properties	Description
constructor	Returns a reference to array function that created a object
index	Represents zero-based index of the match in the function
input	Property is present only in arrays created by regular expression matches
length	Returns number of elements in an array
prototype	Used to add properties and methods to an object

Array Methods

Method	Description
pop()	Removes the last element from an array and returns that element.
push(element)	Adds one or more elements to the end of an array and returns the new length of the array.
indexof(element)	Returns the least index of the element within the array equal to specified value, or -1 if it is not found
toString()	Returns a string representing array and its elements
join(seperator)	Joins all elements of an array into a string
shift()	Removes first element from an array and returns that element
sort(compareFuction)	Sorts the elements of an array

Date

Date

- Date object is built-in object. They can be created using new Date() constructor.
- 4 ways of initiating a date:
  - new Date()
  - new Date(milliseconds) Ex: new Date(100000000000)
  - new Date(dateString) Ex: new Date("October 23, 2016 11:19:00")
  - new Date(year, month, day, hours, minutes, seconds, milliseconds) Ex: new Date(99, 5, 24, 11, 33, 30, 0)

Date Methods	Description
getDay(), getDate(), getMonth(), getFullYear(), getTime()	Getters
setDate(num), setHours(h, m, s, ms), setMinutes(m, s, ms)	Setters
now()*	Numeric representing current time
parse (string)*	Accepts a string representation of a date and returns No. of milliseconds since Jan 1, 1970

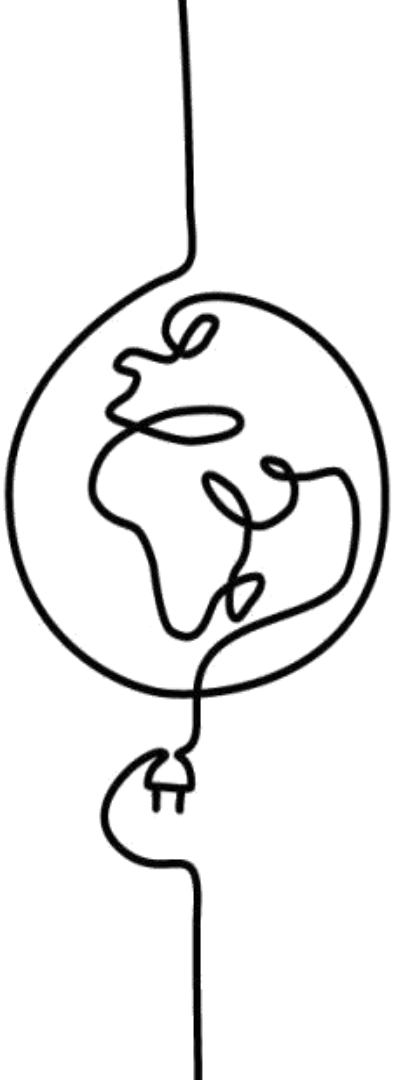
Math

Math

- Math object provides properties and methods for mathematical constants and functions.
- Used to perform mathematical tasks on numbers.

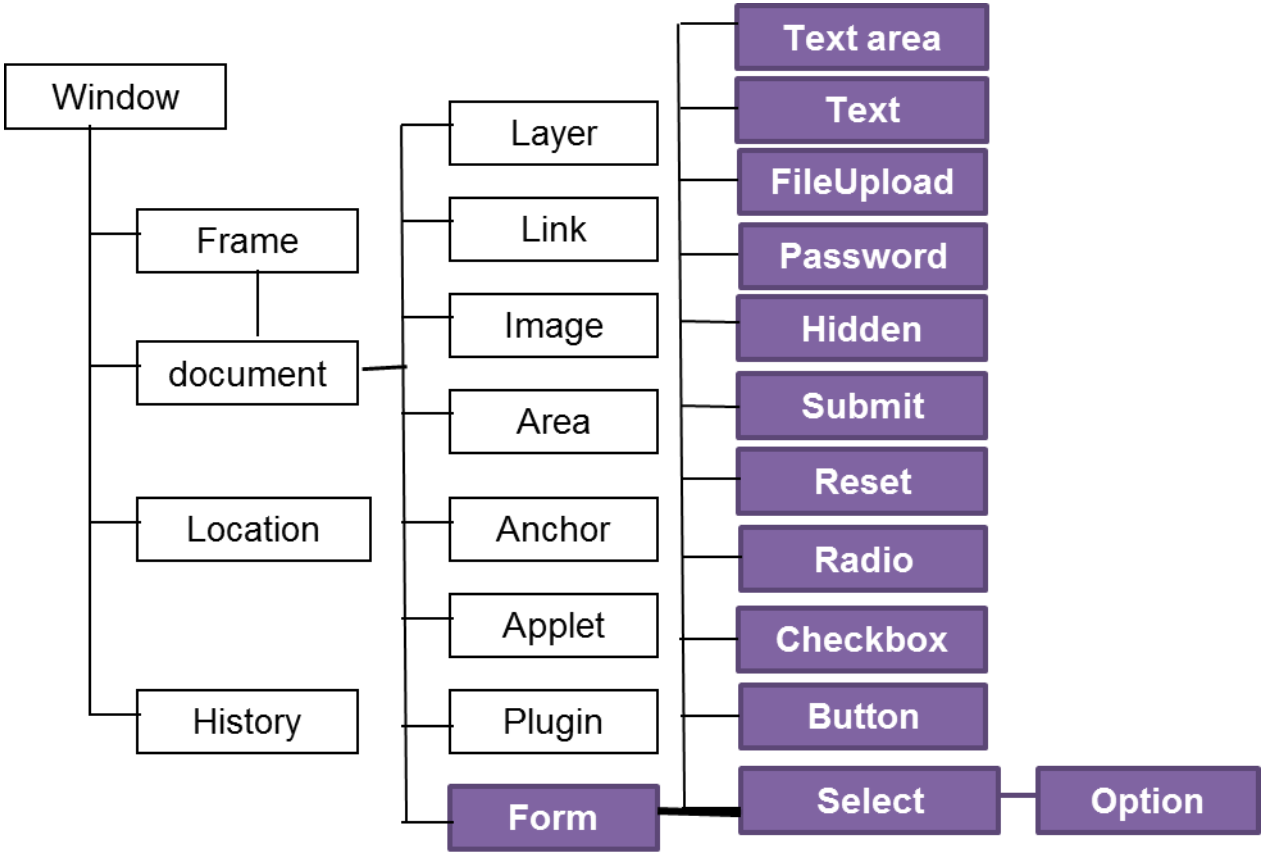
Properties	Mathematical Functions
• E • PI • LN10 • SQRT2	• abs() • ceil() • floor() • round() • log() • exp() • sin() • sqrt() • pow()

Event Handling, Form Handling and Validations



Infosys
be more

Hierarchy of Objects



Text, Textarea, Password Objects

- Properties
  - defaultValue : Reflects the VALUE attribute.
  - name : NAME attribute.
  - type : Reflects the TYPE attribute
  - value : Reflects the current value of the Text object's field
- Methods
  - focus() : Gives focus to the object
  - blur() : Removes focus from the object
  - select() : Selects the input area of the object
- Event Handler
  - onBlur : When a form element loses focus
  - onChange : Field loses focus and its value has been modified
  - onFocus: When a form element receives input focus
  - onSelect : When a user selects some of the text within a text field.

Radio Object

- Properties
  - name, type, value, defaultChecked, defaultvalue, checked
  - checked property will have a Boolean value specifying the selection state of a radio button. (true/false)
- Methods
  - click( ), focus( ), blur( )
- Event Handler
  - onClick, onBlur, onFocus()

Checkbox Object

- Properties
  - checked, defaultChecked, name, type, value
- Methods
  - click()
- Event Handler
  - onClick, onBlur, onFocus()

Select , Option Objects for Drop Down List

- Properties for 'Select' Object:

length	Reflects the number of options in the selection list.
options	Reflects the OPTION tags.
selectedIndex	Reflects the index of the selected option (or the first Selected option, if multiple options are selected).

- Methods for 'Select' Object: blur(), focus()
- Event Handler for 'Select' Object : onBlur, onChange, onFocus
- Properties for 'Option' Object

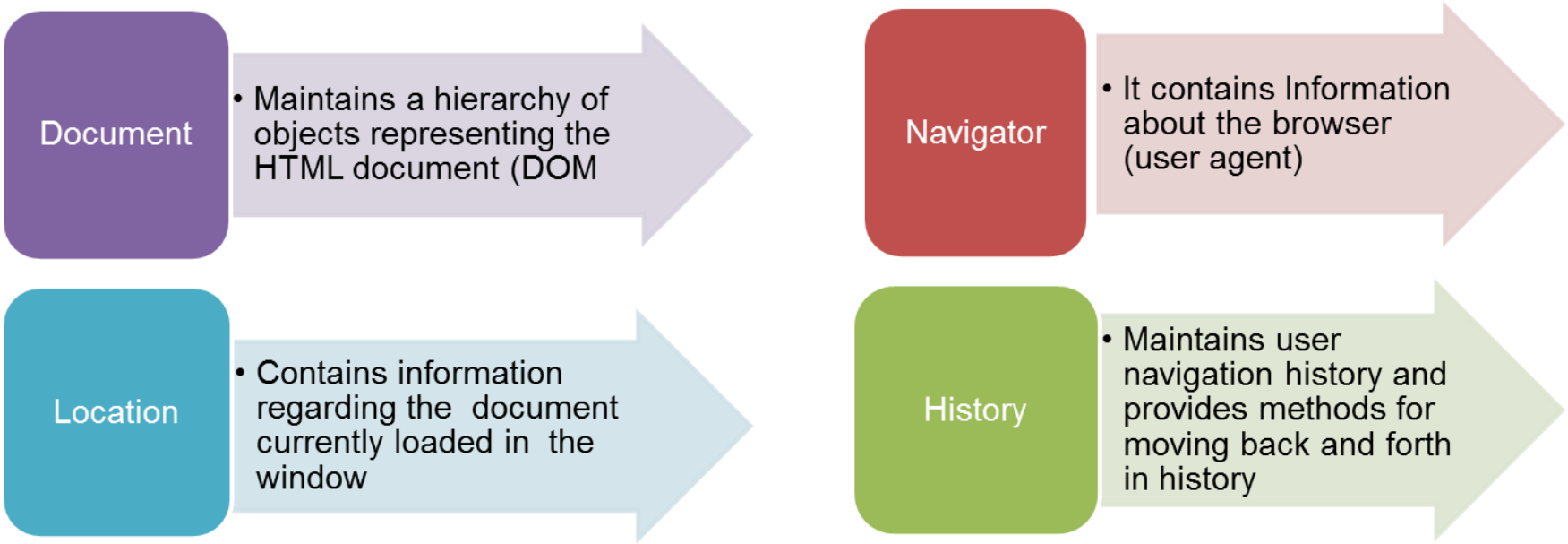
index	The zero-based index of an element in the select.options array.
selected	Specifies the current selection state of the option
text	Specifies the text for the option
value	Specifies the value for the option
length	The number of elements in the select.options array.

Reset, Submit Objects

- Properties
  - form, name, type, value
- Methods
  - click( ), blur( ), focus( )
- Event Handler
  - onClick, onBlur, onFocus, onMouseDown, onMouseUp
- The disabled property (Applicable for all the form controls)
  - If this attribute is set to true, the button is disabled

Browser Object Model

Browser Object Model (BOM)



Windows and Documents

Window Object

- All Global variables and global functions are properties and methods respectively of window object.

Ex: window.document.getElementById("header") is same as window.document.getElementById("header")

Window Size: can be determined by using following properties

window.innerHeight – inner height of browser window (in pixels)

Window.innerWidth – inner width of browser window (in pixels)

Other window Objects:

window.open() – open a new window

window.close() – close the current window

window.moveTo() – move the current window

window.resizeTo() – resize the current window

Window Object

Window Screen:

- window.screen object properties are used to display the properties of viewport/users screen.
- All properties can be window.screen object can be invokes without window prefix

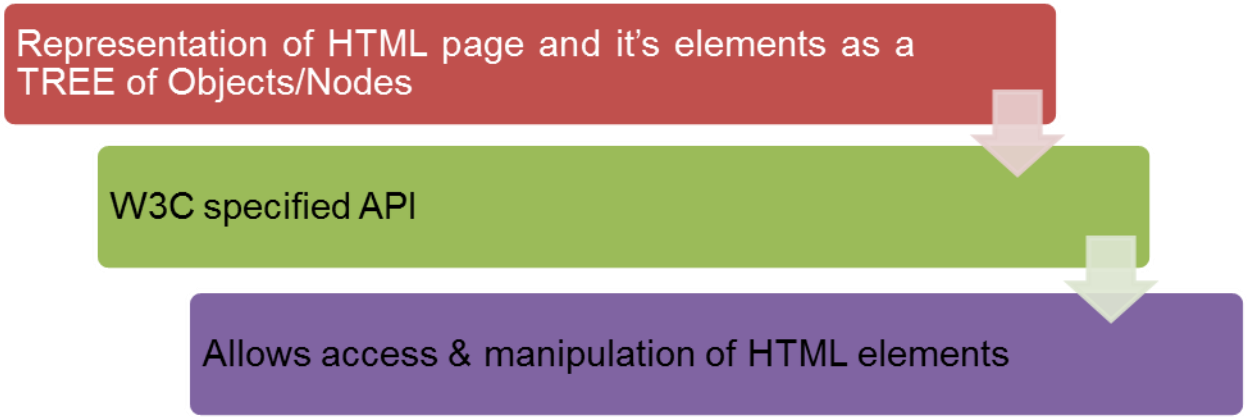
Properties	Description
screen.width	Returns width of viewport in pixels
screen.height	Returns height of viewport in pixels
screen.availWidth	Returns width of viewport in pixels excluding interfaces like windows taskbar
screen.availHeight	Returns width of viewport in pixels excluding interfaces like windows taskbar
screen.colorDepth	Returns number of bits used to display one color
screen.pixelDepth	Returns pixel depth of the screen

Introduction to DOM

Document Object Model (DOM)

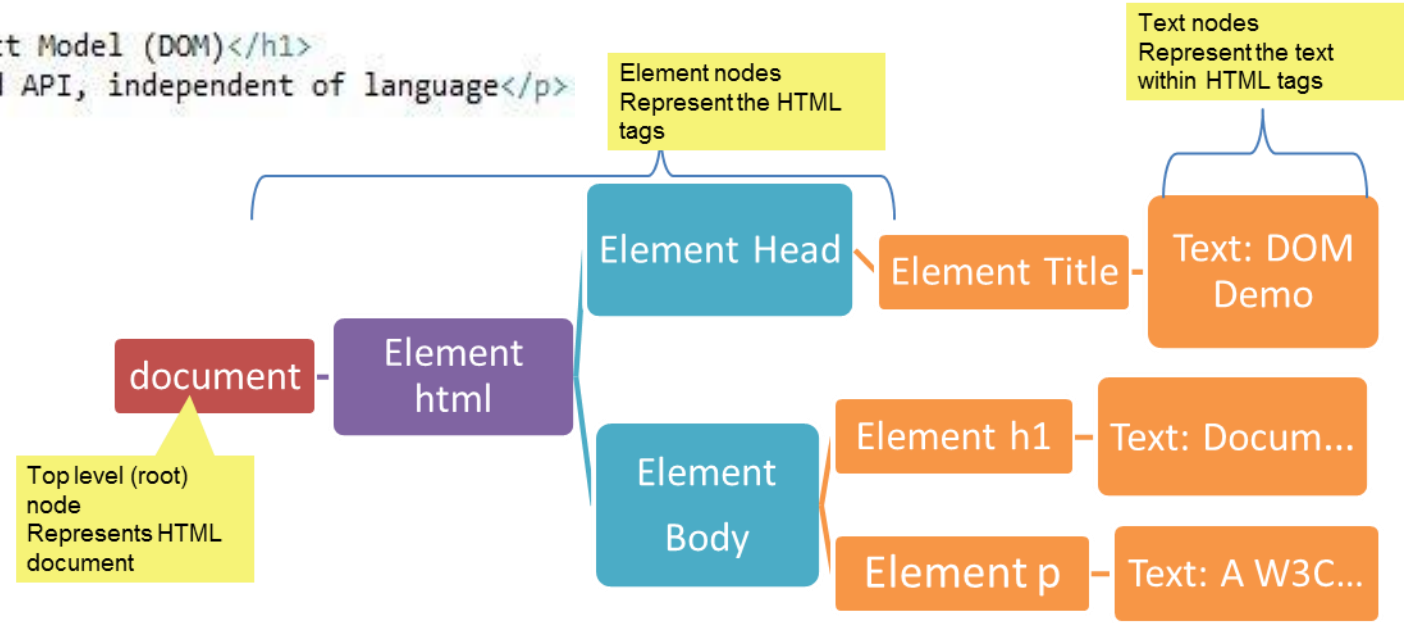
- The World Wide Web Consortium (W3C) is the body which sets standards for the web
- W3C has defined a standard Document Object Model, known as the W3C DOM for accessing HTML elements from a browser
- It views HTML documents as a tree structure of elements and text embedded within other elements
- DOM stands apart from JavaScript because other scripting languages can also access it
- All the browsers are following DOM and because of this the JavaScript code will work on all the browsers in the same way.

Document Object - What is DOM?

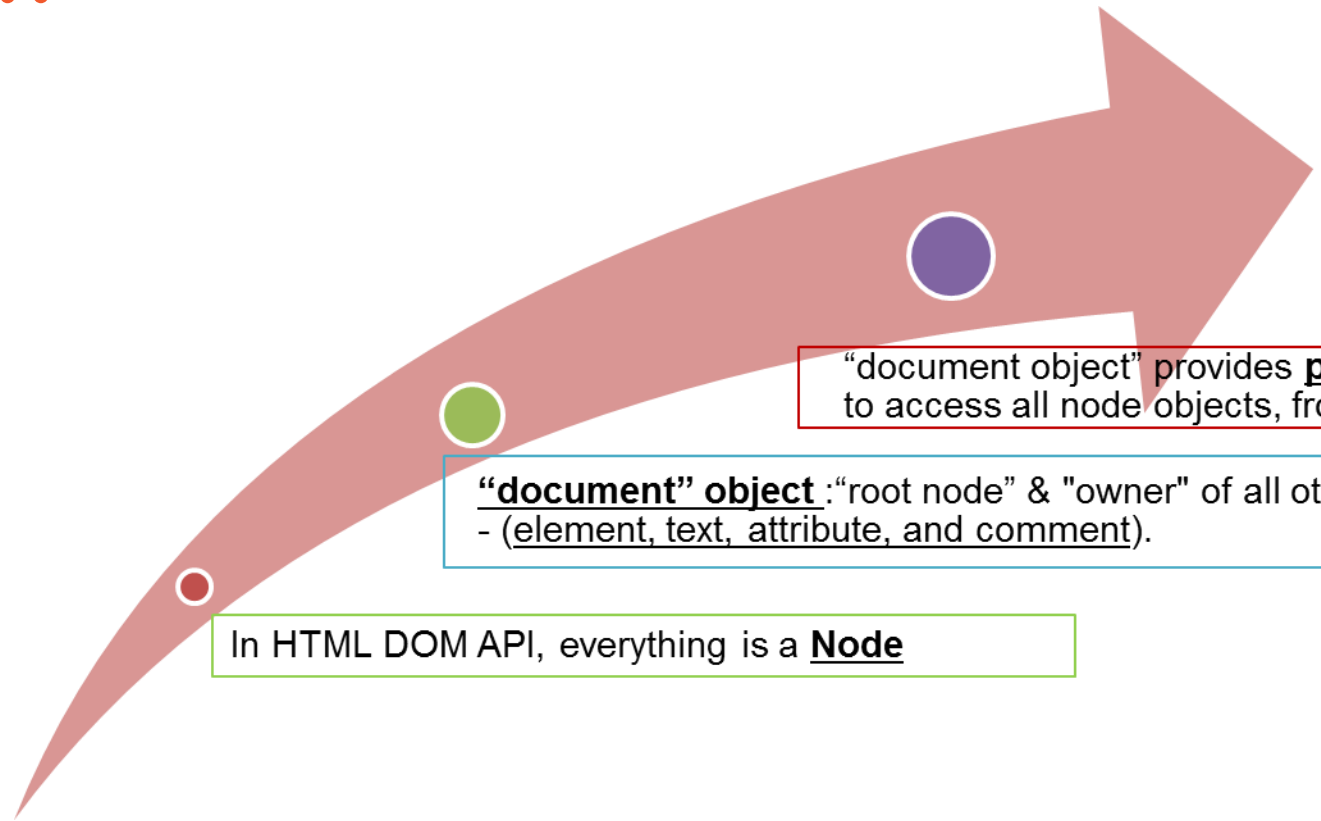


DOM Hierarchy

```
<html>
    <head>
        <title>DOM Demo</title>
    </head>
    <body>
        <h1>Document Object Model (DOM)</h1>
        <p>A W3C specified API, independent of language</p>
    </body>
</html>
```



DOM API



"document object" provides **properties** & **methods** to access all node objects, from within JavaScript.

"document" object :"root node" & "owner" of all other nodes - (element, text, attribute, and comment).

In HTML DOM API, everything is a **Node**

DOM API – “document” object

Properties	Description
document.documentElement	The HTML’s head node
document.body	The HTML’s body node
document.doctype	The HTML’s DOCTYPE node
document.title	The HTML’s title node (title of page)
document.forms	Returns collection of form elements in the documents.

DOM API – “document” object

Methods	Description
getElementById(strID)	Returns the element that has the ID attribute with the specified value
getElementsByTagName(strTag)	Returns a NodeList containing all elements with a specified name
getElementsByName(strName)	Returns a NodeList containing all elements with the specified tag name
getElementsByClassName(strTag)	Returns a NodeList containing all elements with the specified class name
document.write(strTxt)	Writes HTML expressions or JavaScript code to a document

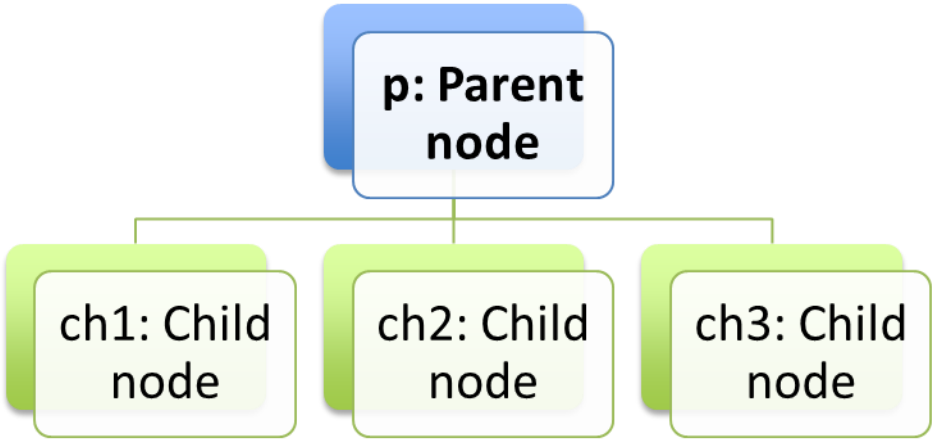
DOM API – “element” object

Properties	Description
element.attributes	Returns a NamedNodeMap of an element's attributes
element.childNodes	Returns a collection of an element's child nodes
element.firstChild	Returns the first child node of an element
element.innerHTML	Sets or returns the content of an element
element.lastChild	Returns the last child node of an element
element.nodeName	Returns the name of a node
element.nodeType	Returns the node type of a node
element.nodeValue	Sets or returns the value of a node

DOM API – “element” object

Methods	Description
addEventListener()	Attaches an event handler to the specified element
appendChild()	Adds a new child node, to an element, as the last child node
getAttribute()	Returns the specified attribute value of an element node
insertBefore()	Inserts a new child node before a specified, existing, child node
removeChild()	Removes a child node from an element

Node relationships



- p.childNodes* = [ch1,ch2,ch3]
- p.firstChild=ch1
- p.lastChild=ch3
- ch1.nextSibling=ch2
- ch2.previousSibling=ch1
- [ch1|ch2|ch3].parentNode=p
- ch3.nextSibling
- ch1.previousSibling = null

JSON

JSON

- JSON is JavaScript Object Notation (key-value pair)
- It is a syntax/format for storing and exchanging data created by Douglas Crockford

- How a JSON code looks like
- How JSON can be called using JavaScript
- JSON.parse and JSON.stringify methods to be explained

Simple JSON Array:


```
{
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]
}
```

JSON to JavaScript Object

- We can convert JSON to JavaScript Object using Parse()

```
<script>
var text = '{"employees":[' +
'{"firstName":"John","lastName":"Doe" },' +
'{"firstName":"Anna","lastName":"Smith" },' +
'{"firstName":"Peter","lastName":"Jones" }]}';

obj = JSON.parse(text);
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;
</script>
```

JavaScript Object to JSON string

- We can convert JavaScript Object to JSON string using stringify()

```
<script>

var myObj = { "name":"John", "age":31, "city":"New York" };
var myJSON = JSON.stringify(myObj);

</script>
```

Example


```
<!DOCTYPE html>
<html>
<body>

<h2>Store and retrieve data from local storage.</h2>

<p id="demo"></p>

<script>
var myObj, myJSON, text, obj;

//Storing data:
myObj = { "name":"John", "age":31, "city":"New York" };
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

//Retrieving data:
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
</script>

</body>
</html>
```

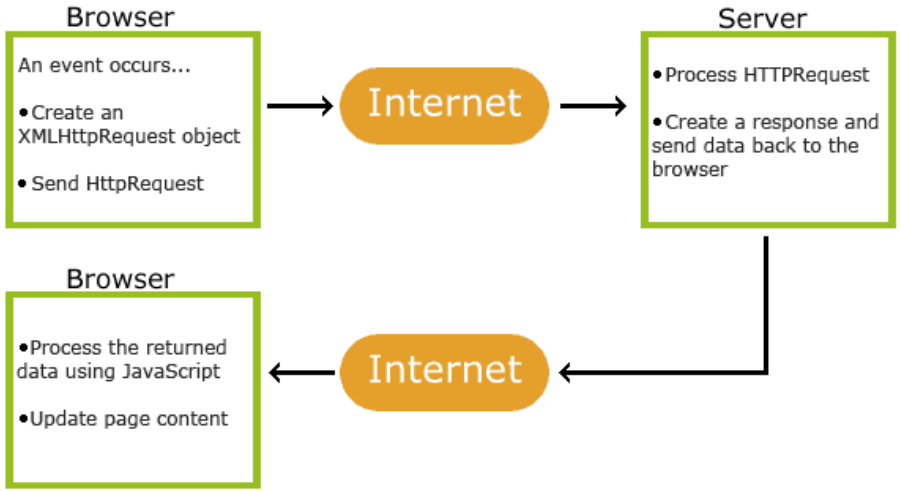
Output:

Store and retrieve data from local storage.

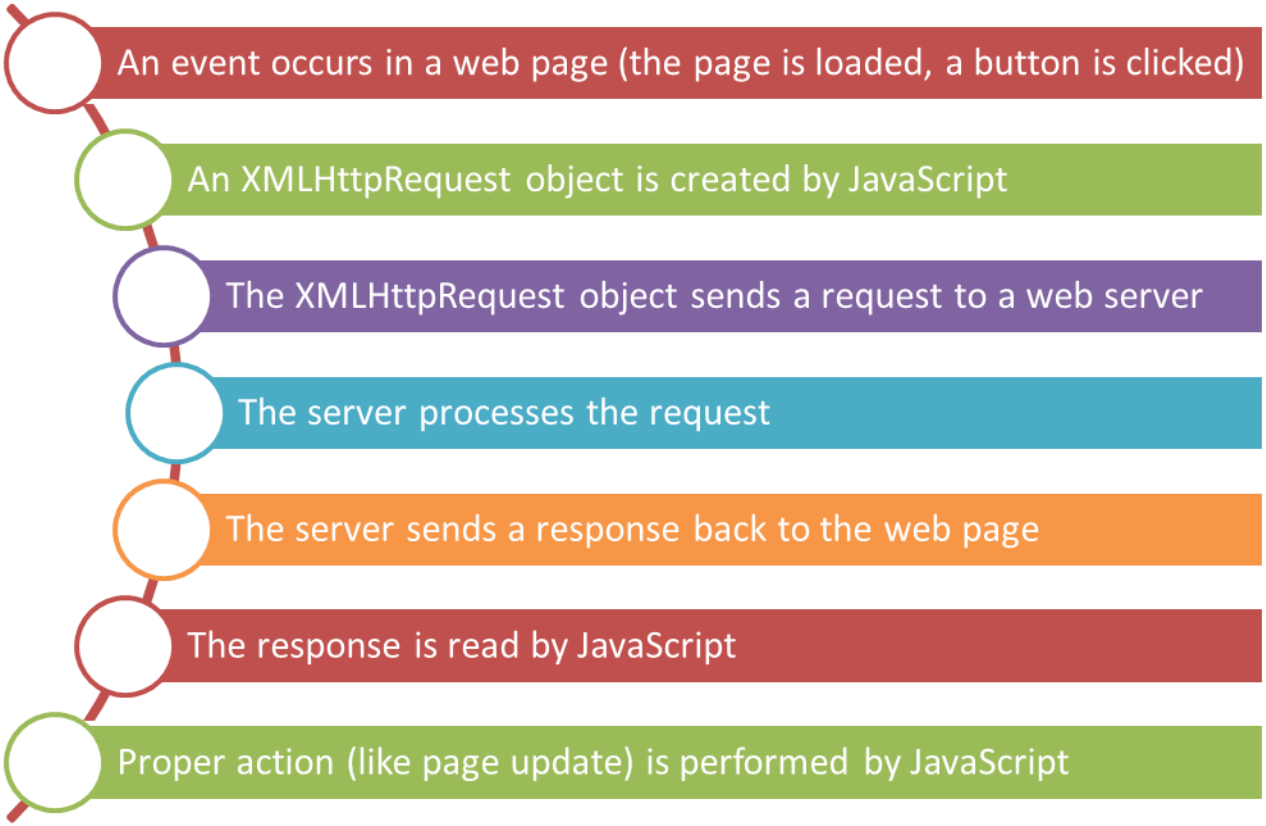
John

Introduction to AJAX

- AJAX (Asynchronous JavaScript and XML) is a developer's dream, because you can:
  - Read data from a web server - after the page has loaded
  - Update a web page without reloading the page – Partial page rendering
  - Send data to a web server - in the background



Process of AJAX



Send request to server using AJAX

- Open() and Send() is used to sent a request to the server

```
var xhttp = new XMLHttpRequest();


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

Receive a response from server using AJAX(1/2)

- **readyState** property holds the status of the XMLHttpRequest.
- **onreadystatechange** property defines a function to be executed when the readyState changes.
- **status** property and the **statusText** property holds the status of the XMLHttpRequest object

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
Status	200: "OK" 403: "Forbidden" 404: "Page not found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Receive a response from server using AJAX(1/2)


```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>

</body>
</html>
```

On-click of the button read the content from a text file

Summary

- You are now knowledgeable on:
  - Use of Scripting language in web pages
  - Functions, Variables and Data Types
  - Operators, Control Statements and Expressions
  - Objects
    - Array
    - Date
    - Math
  - Event Handling, Form Handling and Validations
  - Browser Object Model
  - Windows and Documents
  - Introduction to DOM
  - JSON retrieval using JavaScript
  - Partial Page rendering using AJAX

References

- https://www.w3schools.com/js/default.asp
- https://www.tutorialspoint.com/javascript/index.htm
- https://html.net/tutorials/javascript/lesson18.php
- https://www.javatpoint.com/ajax-json-example
- https://javascript.info/
- https://www.javatpoint.com/javascript-tutorial
- https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics
- http://www.tutorialsteacher.com/javascript/javascript-tutorials

Thank You