**Web application - 3 tier Architecture**

1. **Front-end:**

* Anything that a user faces is a part of Frontend.
* Frontend development has everything to do from design to dynamism of a web application

1. **Back-end:**

* Backend is like the brain.
* It has everything to do with the logical.
* It also takes care of data storage and management by connecting to the database.
* It can combine various services to produce the desired results.

1. **Database:**

* Used to store data.

**JavaScript Introduction**

* JavaScript is a client side scripting language (interpreted programming language).
* Open source and cross-platform.
* Case sensitive.
* Most commonly used as a part of web pages.
* JavaScript was initially known as LiveScript**.**
* Supported by all major browsers and enabled by default.
* JavaScript was initially created as a browser-only language, but now it is used in many other environments as well.

**What JavaScript can do?**

* Add new HTML to the page, change the existing content, modify styles.
* React to user actions, Execute on mouse clicks, pointer movements, key presses.
* Send requests over the network to remote servers, read and write files (Ajax).
* Get and set cookies, ask questions to the visitor, show messages.
* Remember the data on the client-side (“local storage”).

**Advantages of JavaScript**

* Executed on the client side.
* Instance response to the visitors.
* Rich interfaces.
* Speed.
* Less server interaction.

**Disadvantages of JavaScript**

* Code Always Visible.
* **Stop Render** : JavaScript single error can stop to render with entire site. However browsers are extremely tolerant of JavaScript errors.
* Less Security.

**Where to write JavaScript**

1. **Internal JS**

Internal Javascript code is code that's placed anywhere within the web page between the HTML tags.

<script>

alert("Hello Javatpoint");

</script>

**2. External JS**

Javascript code placed in a file separate from the HTML code is called external Javascript. External Javascript code is written and used in the same way as internal Javascript. The file should have the ".js" extension.

<script>

**JavaScript Variables**

**Declaring JavaScript Variables and Rules**

* Declare a JavaScript variable with the **var** keyword.
* Variables can contain letters, digits, underscores, and dollar signs.
* Variables must begin with a letter.
* Variables are case sensitive.
* Can not use Reserved words as Variables.
* Use proper names and **camelcase** if it has more than one word.

A JavaScript variable is simply a name or container for storing data values.

There are two types of variables in JavaScript : **local variable** and **global variable**.

**Local Variable**: A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

**Global Variable**: A JavaScript global variable is declared outside the function or declared with window object. It can be accessed from any function.

**Data Types**

Data types plays an important role in programming Language.

To be able to operate on variables, it is important to know something about the type.

Without knowing data types, a program cannot handle safely.

**List of Data Types:**

1. **String:** Sequence of characters which is denoted by single or double quotes.
2. **Number**: Represent a numeric value with decimal or without decimal .
3. **Array**: Array is a special variable used to store multiple values in a single variable.
4. **Object:** Object is a variable containing many variables.
5. **Boolean:** Represents value in two states: true or false.
6. **Null**: null is "nothing". Something that doesn't exist.
7. **Undefined:** variable can be emptied, by setting the value to undefined.

**JavaScript** **Display Methods**

1. alert()
2. document.write()
3. innerHTML
4. console.log()

**JavaScript Function**

Functions are one of the fundamental building blocks in JavaScript. A function is a JavaScript procedure—a set of statements that performs a task or calculates a value.

**How to create a Function?**

JavaScript functions are defined with the **function** keyword and followed by the functionName.

There are 3 different types of functions,

1. **Function declaration // Named function**

Function declarations load before any code is executed.

***function functionName()***

***{***

***}***

1. **Function expression // anonymous** function

Function expressions load only when the interpreter reaches that line of code.

var a = function() { };

Also this function is called as **anonymous** function (a function without a name).

**3**. **Constructor function**

Constructors are like regular functions, but we use call them with the "new" keyword. There are two types of constructors:

1. Native (built-in) constructors like **Array, Number** and **Object**
2. Custom Constructor functions.

**Function Invocation methods**

1. Self Invocation
2. By Events
3. Invoking a function from another one function

**String Inbuilt Methods:**

Sequence of characters which is denoted by single or double quotes.

**To Find String Length**

String.length;

var str = “Hi”;

str.length; // 2

**charAt()**

Returns the character at the specified index position of the string.

**Syntax**:

string.charAt(index)

**indexOf()**:

Returns the index position of the first occurrence of a specified value in a string.

**Syntax**:

string.indexOf(specifiedSubString, startingIndexPosition);

**lastIndexOf():**

Returns the index position of the last occurrence of a specified value in a string.

**Syntax**:

string.lastIndexOf(specifiedSubString, EndIndexPosition)

**Extracting a Substring from a main string:**

string.slice(startIndexPosition, EndIndexPosition)

string.substring(startIndexPosition, EndIndexPosition)

string.substr(startIndexPosition, length)

\*Exclude EndIndexPosition

**replace()**

Searches in a string for a specified value and returns a new string where the specified values are replaced.

**Syntax**:

string.replace(searchString, newString)

**startsWith()**

Checks whether a string begins with specified characters.

**Syntax**:

string.startsWith(searchString, startIndexPosition)

**endsWith()**

Checks whether a string ends with specified characters.

**Syntax**:

string.endsWith(searchString, Length)

**includes()**

Checks whether a string contains the specified string/characters

**Syntax**:

string.includes(searchString, startIndexPosition)

**match()**

Searches a string for a match against a regular expression, and returns the matches

**Syntax**:

string.match(regexp)

It will return an Array with matched values

Regular expression for email:

**^([a-zA-Z0-9\_\-\.]+)@([a-zA-Z0-9\_\-\.]+)\.([a-zA-Z]{2,5})$**

**repeat()**

Returns a new string with a specified number of copies of an existing string

**Syntax**:

string.repeat(count)

**split()**

Splits a string into an array of substrings

**Syntax**:

string.split(separator, limit)

**toString()**

Returns the value of a String object

**Syntax**:

string.toString()

**toUpperCase()**

Converts a string to uppercase letters

**Syntax**:

string.toUpperCase()

**toLowerCase()**

Converts a string to lowercase letters

**Syntax**:

string.toLowerCase()

**trim()**

Removes whitespace from both ends of a string

**Syntax**:

string.trim()

**Number Inbuilt Methods:**

**isFinite()**

Checks whether a value is a finite number

**Syntax**:

Number.isFinite(num)

**isInteger()**

Checks whether a value is an integer

**Syntax**:

Number.isInteger(num)

**toFixed(x)**

converts a number into a string, keeping a specified number of decimals

**Syntax**: num.toFixed(2)

**toString()**

Converts a number to a string

**Syntax**: num.toString();

**Converting String to Numbers**

1. **Number()** Returns a number, converted from its argument.
2. **parseInt()** parses a string and returns a whole number. Spaces are allowed. Only the first number is returned.
3. **parseFloat()** parses a string and returns a number. Spaces are allowed. Only the first number is returned.

**JavaScript Math Object**

Math object used to do mathematical operations on numbers.

**Math.round()**

Math.round(num) returns the value of x rounded to its nearest integer.

**Math.pow()**

Math.pow(x, y) returns the value of x to the power of y.

**Math.sqrt()**

Math.sqrt(x) returns the square root of x.

**Math.abs()**

Math.abs(x) returns the absolute (positive) value of x.

**Math.ceil()**

Math.ceil(x) returns the value of x rounded **up** to its nearest integer.

**Math.floor()**

Math.floor(x) returns the value of x rounded **down** to its nearest integer.

**Math.random()**

Math.random() returns a random number between 0 to 1.

**Javascript Events**

Events are the actions which is happened on HTML

Onclick

Onblur

Onchange

Onsubmit

Onfocus

Onmouseover

onmouseleave

onmousemove

ondblclick

Oncontextmenu

onselect

Onkeypress (character oriented)

Onkeydown (key oriented)

onkeyup

onload

oncopy

oncut

Onpaste

**Array Inbuilt Methods**

Array is a special variable, used to store many values in a single Variable.

**push()**

The **push()** method adds a new element at the end of an array

**Syntax**: ArrayName.push(“New Value”);

It will return the length of an Array.

**pop()**

The **pop()** method removes the last element from an array

**Syntax**: ArrayName.pop();

It will return the removed value of an Array.

**shift()**

The **shift()** method removes the very first element of an Array.

**Syntax**: ArrayName.shift();

It will return the removed value of an Array.

**unshift()**

The **unshift()** method adds a new element at the beginning of an array.

**Syntax**: ArrayName.unshift();

It will return the length of an Array.

**splice()**

The **splice()** method is used for add / remove values in anywhere of an Array.

**Syntax**:

ArrayName.splice(Index position of where the new element should be add/delete, how many number of elements should be removed, New elements to be add);

**concat()**

The **concat()** method creates a new array by concatenating existing arrays.

**Syntax**: Array1.concat(Array2, Array3);

**slice()**

The **slice()** method slices out a piece of an array into a new array.

**Syntax**: Array1.slice(Number of elements to be remove, excluding end index position);

**fill()**

Fill all the array elements with the specified value.

**Syntax**: ArrayName.fill(“New Value”);

**forEach()**

Calls a function for each array element

**Syntax**: ArrayName.forEach(functionName);

function functionName(value, index)

{

}

**indexOf()**

Return an index position of the 1st occurrence of the value

**Syntax**: ArrayName.indexOf(“Value”);

**lastIndexOf()**

Return an index position of the last occurrence of the value

**Syntax**: ArrayName.lastIndexOf(“Value”);

**join()**

The join() method joins the elements of an array into a string, and returns the string.

The elements will be separated by a specified separator. The default separator is comma (,).

**Syntax**: ArrayName.join(“separator”);

**sort()**

the sort() method sorts the values as strings in alphabetical and ascending order.

**Syntax**: ArrayName.sort();

**reverse()**

The reverse() method reverses the order of the elements in an array.

**Syntax**: ArrayName.reverse();

**Conditional Statements**

Based on condition we can execute a particular task.

1. if .. else
2. switch

**Iteration Statements**

Execute set of statements several times based on the given condition is true.

1. For
2. While
3. Do while
4. forEach
5. For in

**Object introduction**

\*Object is a special variable, used to store many values in a single variable.

\*Also called as Named Index.

\*Object is self describing.

\*Object starts with { and ends with }

\*Data are stored as 'key:value' format.

\*Data are separated by ','

\*Object has 2 factors,

**i. Property**

Values which is assigned on the object.

**ii. method**

Actions which is performed by the object.

\*You can’t compare 2 objects directly

\*You can’t find length for an object

**3 ways to create an Object**

1. Create an Object by using Literal method.
2. Create an Object by using “new” keyword.
3. Create an Object by using Constructor.

**Timer Functions**

There are 2 timer functions available in JavaScript

1. setTimeout()

setTimeout allows you to execute the statement once after a given period of time.

**Syntax:**

setTimeout(function, milliseconds);

2. setInterval()

setInterval allows you to execute the statement repeatedly after a given time delay.

**Syntax :**

setInterval(*function, milliseconds*);

**DOM Manipulation**

* changing an element's style
* adding/removing classes
* changing the content of a tag
* changing attributes(src, href, etc.)

**Ways to Find HTML Elements:**

document.getElementById("id");

document.getElementsByName("elementName");

document.getElementsByTagName("tagName");

document.getElementsByClassName("className");

document.querySelector("");

document.querySelectorAll("");

**getElementById**

var tag = document.getElementById("highlight");

Returns the one element with a matching ID

*<body>*

*<h1>Hello</h1>*

*<h1>Goodbye</h1>*

*<ul>*

*<li id="highlight">List Item 1</li>*

*<li class="bolded">List Item 2</li>*

*<li class="bolded">List Item 3</li>*

*</ul>*

*</body>*

**getElementsByClassName**

*var tags = document.getElementsByClassName("bolded");*

console.log(tags[0]);

Returns a list of elements that have a matching class

*<body>*

*<h1>Hello</h1>*

*<h1>Goodbye</h1>*

*<ul>*

*<li id="highlight">List Item 1</li>*

*<li class="bolded">List Item 2</li>*

*<li class="bolded">List Item 3</li>*

*</ul>*

*</body>*

**getElementsByTagName**

var tags = document.getElementsByTagName("li");

console.log(tags[0]);

Returns a list of all elements of a given tag name, like <li> or <h1>

*<body>*

*<h1>Hello</h1>*

*<h1>Goodbye</h1>*

*<ul>*

*<li id="highlight">List Item 1</li>*

*<li class="bolded">List Item 2</li>*

*<li class="bolded">List Item 3</li>*

*</ul>*

*</body>*

**querySelector**

//select by ID

var tag = document.querySelector("#highlight");

//select by Class

var tag = document.querySelector(".bolded");

//select by tag

var tag = document.querySelector("h1");

Returns the first element that matches a given CSS-style selector

*<body>*

*<h1>Hello</h1>*

*<h1>Goodbye</h1>*

*<ul>*

*<li id="highlight">List Item 1</li>*

*<li class="bolded">List Item 2</li>*

*<li class="bolded">List Item 3</li>*

*</ul>*

*</body>*

**querySelectorAll**

//select by tag

var tags = document.querySelectorAll("p.test");

Returns a list of elements that matches a given CSS-style selector

*<body>*

*<h1>Hello</h1>*

*<h1>Goodbye</h1>*

*<ul>*

*<li id="highlight">List Item 1</li>*

*<li class="bolded">List Item 2</li>*

*<li class="bolded">List Item 3</li>*

*</ul>*

*</body>*

**Window Object**

The window object represents a window containing a DOM document.

**window.location.href** sets or gets absolute url of the current page

**window.location.hostname** returns the domain name of the web host

**window.location.pathname** returns the path and filename of the current page

**window.location.protocol** returns the web protocol used (http: or https:)

**window.location.port** property returns the host port number

**window.history.back()** - goes to one step back

**window.history.forward()** - goes to one step forward

**Client / Web Storage**

There are 2 different kind of Storages are available in HTML 5.

1. **Local Storage:**

Permanent Storage.

No expire date unless you remove it.

Domain specific.

**Syntax:**

**window.localStorage.setItem(“key”, “value”);**

**window.localStorage.getItem(“key”);**

**window.localStorage.removeItem(“key”);**

1. **Session Storage**:

Temporary Storage.

Once you closed the browser or inactive of particular period.

Page specific.

**Syntax:**

**window.sessionStorage.setItem(“key”, “value”);**

**window.sessionStorage.getItem(“key”);**

**window.sessionStorage.removeItem(“key”);**

**window.sessionStorage.clear(); // logout**

**JSON**

* JSON stands for JavaScript Object Notation.
* JSON used for store and exchange data between servers or remote servers.
* Its light weight and self describing.
* JSON language independent.
* JSON object starts with ‘{‘ JSON array starts with ‘[‘.
* Easy to parse.
* JSON is purely a data format — it contains only properties, no methods.

**Rules:**

* Key must be a string which is denoted by “”.
* JSON supports the following data types,

I. string

Ii. number

Iii. Array.

Iv. Object

V. null

**JSON doesn’t support**

I. Function and inbuilt methods.

Ii. Undefined

Example:

{"city":{"id":1851632,"name":"Shuzenji"},  
"coord":{"lon":138.933334,"lat":34.966671},  
"country":"JP",  
"cod":"200",  
"message":0.0045,  
"cnt":38,  
"list":[{  
 "dt":1406106000,  
 "main":{  
 "grnd\_level":1005.93,  
 "humidity":87,  
 "temp\_kf":0.26},  
 "weather":[{"id":804,"main":"Clouds","description":"overcast clouds","icon":"04d"}],  
 "clouds":{"all":88},  
 "wind":{"speed":5.71,"deg":229.501},  
 "sys":{"pod":"d"},  
 "dt\_txt":"2014-07-23 09:00:00"}  
 ]}

**AJAX**

AJAX stands for Asynchronous JavaScript And XML.

Ajax used for get / post data from server without reloading the page.

Ajax uses, Browser's inbuilt Object **XMLHttpRequest**.

The XMLHttpRequest object is used to exchange data with a server.

Ajax send / receive data to / from the server by asynchronously.

SPA is the inspiration of Ajax.

Ajax has 5 states,

0: request yet not initialized

1: server connection established

2: request received

3: processing request

4: request completed and response is ready

Movie Api:

<http://www.omdbapi.com/?t=theri&apikey=c429066e>