**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

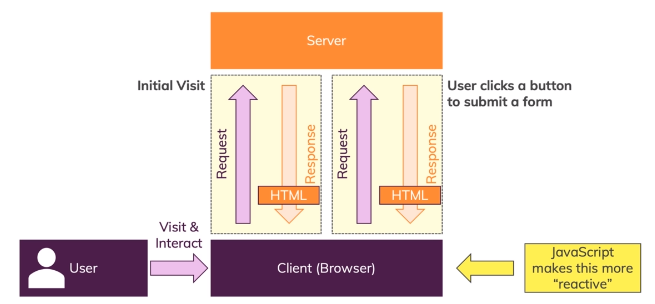
1. **Back-end:**

* 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.

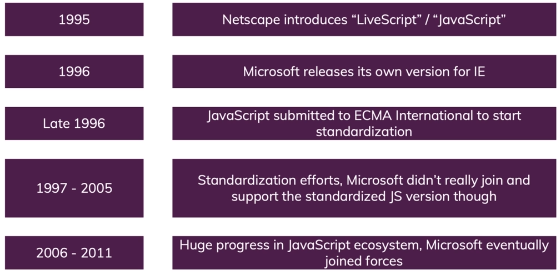
1. **Database:**

* Used to store data.

**How do Web Pages Works?**

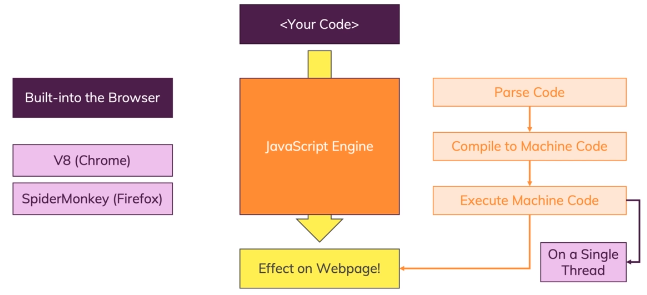
****

**Brief Overview of the JavaScript History**

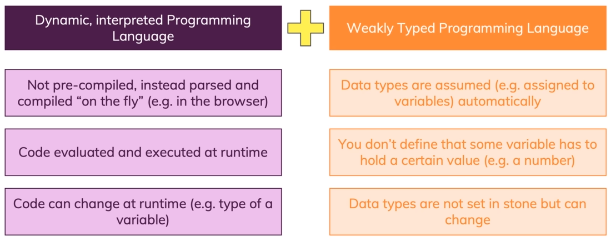
****

**JavaScript Introduction**

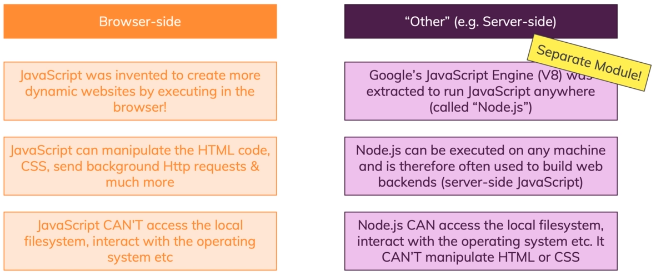
* JavaScript is a client side scripting language (interpreted programming language).
* JavaScript is a dynamic, weakly typed language which compiled at run time.
* Open source and cross-platform.
* Case sensitive.
* Most commonly used as a part of web pages.
* JS was created to make web pages more Dynamic (Change content on a page directly from inside the browser).
* 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.

**How is JavaScript Executing?**

**What is Dynamic & Weakly Typed Language?**

****

**JS in Browser side vs Server side**

****

**What JavaScript can do?**

* Add new HTML to the page, change the existing content, modify styles.

(**DOM Manipulation**)

* React to user actions, Execute on mouse clicks, pointer movements, key presses. (Events)
* 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.

**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 if it has more than one word use camel case.

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:** JS can’t find any value for this. variable can be emptied, by setting the value to undefined.

**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,**

Functions are declared with the following syntax are not executed immediately. They are "saved for later use" and will be executed later, when they are invoked (called upon).

These type of functions work if you call them BEFORE or AFTER where they have been defined. If you call a deceleration function before where it has been defined - Hoisting - works properly.

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

**Difference between function and methods:**

**Function**: A function is a piece of code which is called by Name.

**Method**: A method is a piece of code which is called by Name that is associated with an Object.

**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)

*A regular expression is a sequence of characters that define a pattern.*

**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})$**

The above regular expression can be used for checking if a given set of characters is an email address or not.

**How to write regular expression?**

***The Plus symbol ( + ):***

It tells the computer to repeat the preceding character (or set of characters) for atleast one or more times(upto infinite).

**Example** :

**The regular expression ab+c will give abc, abbc,**

**abbc, … and so on.**

***The curly braces {…}:***

It tells the computer to repeat the preceding character (or set of characters) for as many times as the value inside this bracket.

**Example** :

**{2} means that the preceding character is to be repeated 2**

**times, {min,} means the preceding character is matches min or more**

**times. {min,max} means that the preceding character is repeated at**

**least min & at most max times.**

***Wildcard – ( . )***

The dot symbol can take place of any other symbol, that is why it

is called the wildcard character.

**Example** :

**The Regular expression .\* will tell the computer that any character**

**can be used any number of times.**

***Optional character – ( ? )***

This symbol tells the computer that the preceding character may

or may not be present in the string to be matched.

**Example** :

**We may write the format for document file as – “[doc]?”**

**The ‘?’ tells the computer that x may or may not be**

**present in the name of file format.**

***The caret ( ^ ) symbol:***

Setting position for match :tells the computer that the match must start at the beginning of the string or line.

**Example** :

**^\d{3} will match with patterns like "901" in "901-333-".**

***The dollar ( $ ) symbol:***

It tells the computer that the match must occur at the end of the string or before \n at the end of the line or string.

**Example** :

**-\d{3}$ will match with patterns like "-333" in "-901-333".**

***Character Classes***

A character class matches any one of a set of characters. It is used to match the most basic element of a language like a letter, a digit, space, a symbol etc.

/s : matches any whitespace characters such as space and tab

/S : matches any non-whitespace characters

/d : matches any digit character

/D : matches any non-digit characters

/w : matches any word character (basically alpha-numeric)

/W : matches any non-word character

/b : matches any word boundary (this would include spaces, dashes, commas, semi-colons, etc)

**repeat()**

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

**Syntax**:

string.repeat(count)

**toString()**

Returns the value of a String object

**Syntax**:

string.toString()

**split()**

Splits a string into an array of substrings

**Syntax**:

string.split(separator, limit)

**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)**

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 which is 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 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

**Conditional Statements**

Based on condition we can execute a particular task.

1. if .. else

i. **If** : If you have used multiple if statements then ***if*** the condition is ***true*** all will be executed.

ii. **if else if** : If you have used ***if and else if*** combination only one will be executed where first comes the true value.

1. switch

**Iteration Statements**

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

1. For
2. While
3. Do while

*do*

*{*

*wash\_hands();*

*}*

*while (hands\_are\_dirty());*

*do*

*{*

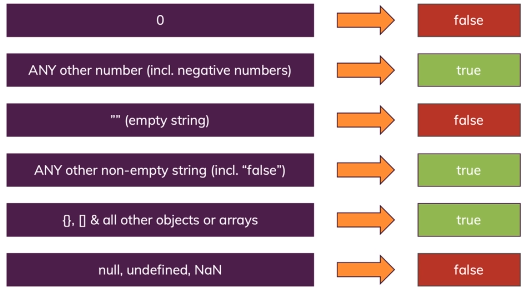
*eat();*

*}*

*while (still\_hungry());*

1. forEach
2. For in
3. For of

**Truthy and Falsy Values**



**Conditional (ternary) operator**

**Syntax**

*condition ? exprIfTrue : exprIfFalse*

**condition**

An expression whose value is used as a condition.

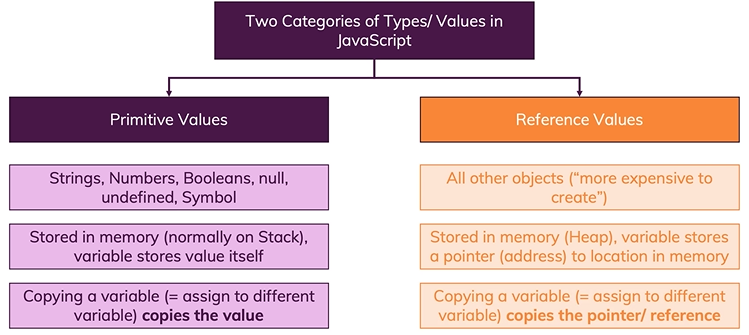
**exprIfTrue**

An expression which is evaluated if the condition evaluates to a truthy value (one which equals or can be converted to true).

**exprIfFalse**

An expression which is executed if the condition is falsy (that is, has a value which can be converted to false).

**Primitive vs Reference Values**

****

**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 the 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(Starting Index position, 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 that is called **callback** function.

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

function functionName(value, index, arr)

{

}

**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();

**every()**

The every() method checks if **all** the elements in an array pass a test which is done inside the callback function.

**Syntax**: ArrayName.every(theCallBackFnName)

If it returns any false, then does not check the remaining values

**some()**

The some() method checks if **any** of the elements in an array pass a test which is done inside the callback function.

**Syntax**: ArrayName.some(theCallBackFnName)

**map()**

The map() method creates a **new array** with the results of calling a function for every array element.

**Syntax**: ArrayName.map(theCallBackFnName)

The map() method calls the provided function once for each element in an array, in order.

**filter()**

The filter() method creates an **array** filled with all array elements that pass a test which is done inside the function.

**Syntax**: ArrayName.filter(theCallBackFnName)

**find()**

The find() method returns the value of the first element in an array that pass a test which is done inside the function.

**Syntax**: ArrayName.find(theCallBackFnName)

**findIndex()**

The findIndex() method returns the index of the first element in an array that pass a test which is done inside the function.

**Syntax**: ArrayName.findIndex(theCallBackFnName)

**Practicals:**

1. ToDo list with Add, Edit, Delete, Search and List

2. Username check Availability with case insensitive. With for and without for

3. Separate numbers and strings in a separate Array.

4. Call a function mirror() with some string and it should give you mirror effect.

Ex: function mirror()

{

return \*\*\*\*\*; // olleH

}

mirror(“Hello”);

**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);

1. setInterval()

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

**Syntax :**

setInterval(*function, milliseconds*);

**Obtrusive and Unobtrusive in Js:**

Separation of HTML and JavaScript (define your JavaScript in external JavaScript files)

**Obtrusive :**

|  |
| --- |
| <input type="button" value="obtrusive" onclick="alert('obtrusive')"> |

**Unobtrusive:**

|  |
| --- |
| <**input** type="button" id="unob" value="unObtrusive">  <**script**>  window.addEventListener("DOMContentLoaded", function () {  document.getElementById("unob").addEventListener("click", function () {  alert("unObtrusive");  });  })  </**script**> |

**JavaScript Hoisting**

Basically, when Javascript compiles all of your code, all variable declarations using **var** are hoisted/lifted to the top of their functional/local scope (if declared inside a function) or to the top of their global scope (if declared outside of a function) regardless of where the actual declaration has been made. This is what we mean by “**hoisting**”.

**Functions** declarations are also hoisted, but these go to the very top, so will sit above all of the variable declarations.

Hoisting is JavaScript's default behavior of moving all declarations to the top of the current scope (to the top of the current script or the current function).

**Note: Variables and constants declared with let or const are not hoisted!**