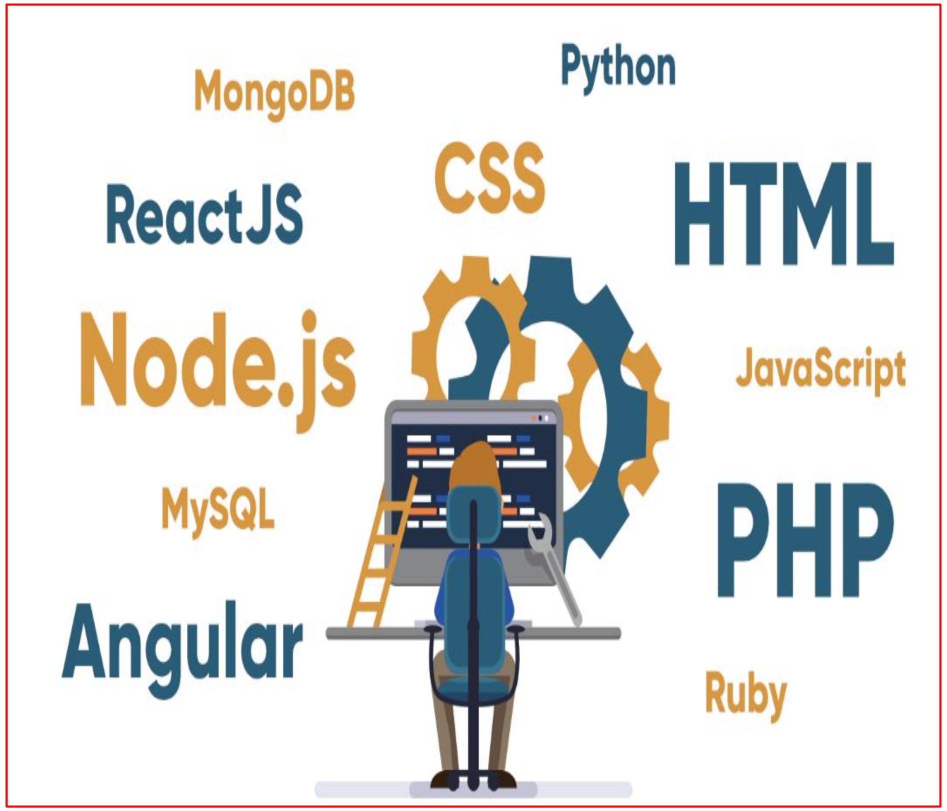
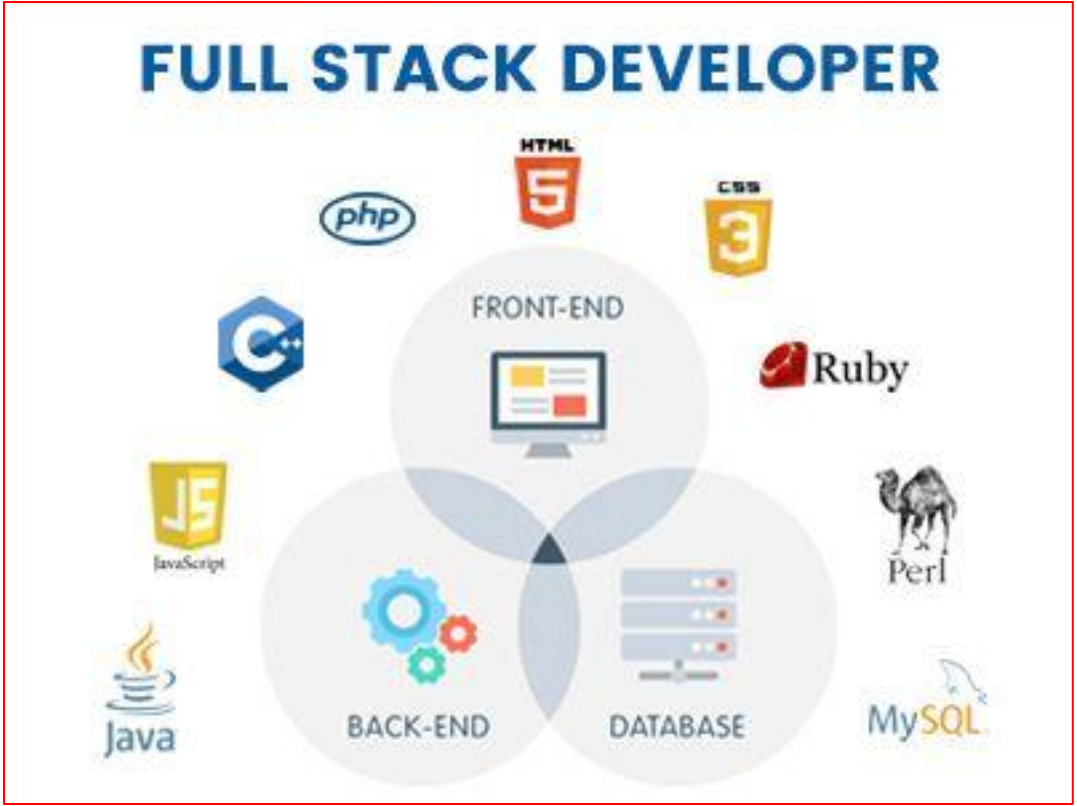
JavaScript

Claritech Solutions LLP

Client Side Vs Server Side Languages





What is Javascript?

- Lightweight programming language ("scripting language")
- Used to make web pages interactive
- Insert dynamic text into HTML (ex: user name)
- React to events (ex: page load user click)
- Get information about a user's computer (ex: browser type)
- Perform calculations on user's computer (ex: form validation)
- Web standard (but not supported identically by all browsers)
- NOT related to Java other than by name and some syntactic similarities
- Open and cross platform and can integrate with HTML or Java.

Javascript History

- Founded by Netscape
- Initially known as LiveScript
- Netscape changed to JavaScript.
- Became European Computer Manufacturers Association(ECMA) 1997
- ES 1 □ ES 6.

Why Study JavaScript?

1. HTML to define the content of web pages
2. CSS to specify the layout of web pages
3. JavaScript to program the behavior of web pages

Benefits Vs Limitations

Benefits/ Features

- Less server interaction
- Immediate feedback to the visitors
- Increased interactivity
- Richer interfaces
- Browser Support: Chrome, Firefox, IE/ Edge, Safari, Opera.

Limitations

- doesn't have any multithreading capabilities.
- cannot be used for networking applications
- does not allow the reading or writing of files

JS Syntax

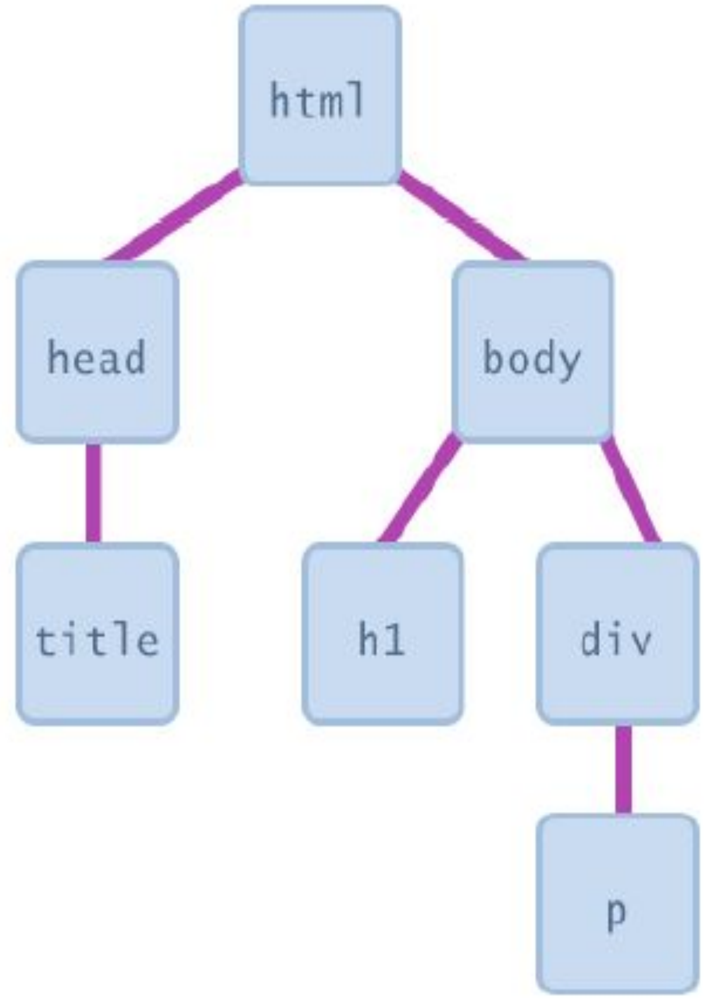
```
<script language = "javascript" type = "text/javascript">
   document.write('Hello JavaScript');
   console.log('Welcome to Scripting Language');
</script>
```

Placement Of JS

- Head Section
- Body Section
- Head and Body Section
- External file

Document Object Model (DOM)

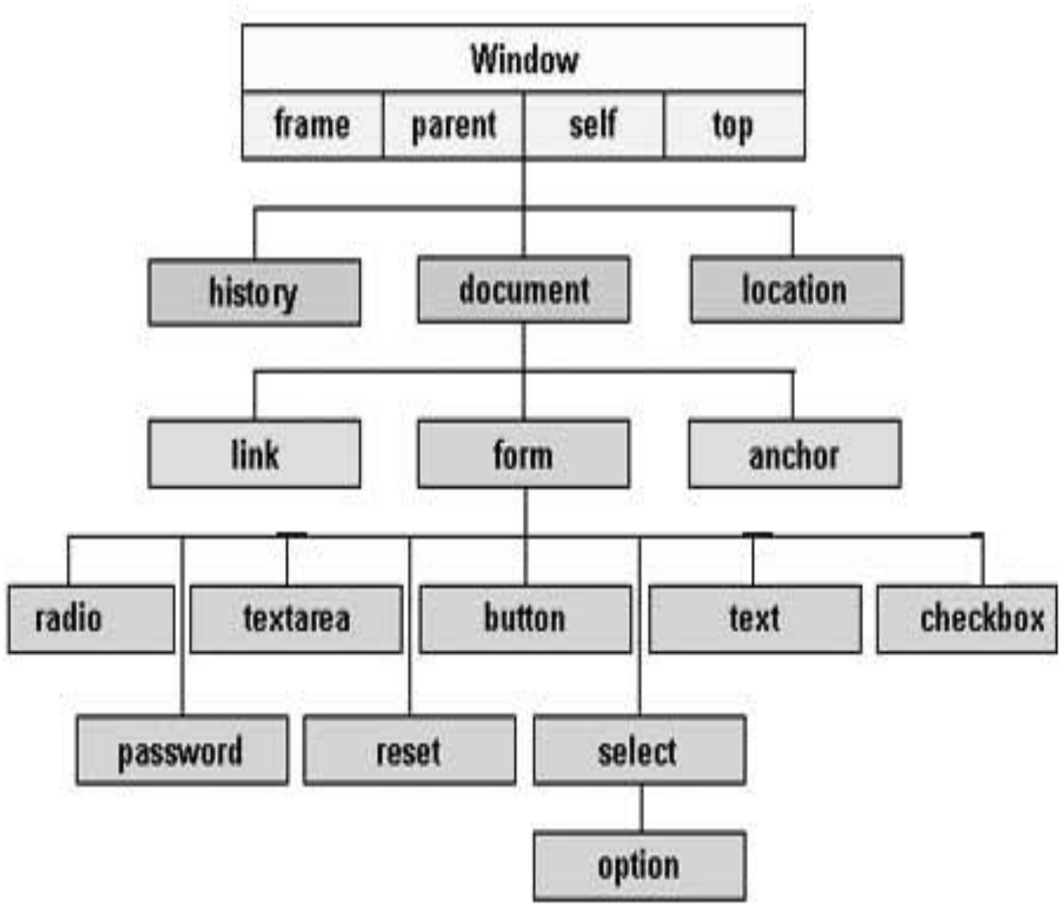
- most JS code manipulates elements on an HTML page
- we can examine elements' state
  - e.g. see whether a box is check
- we can change state
  - e.g. insert some new text into a div
- we can change styles
  - e.g. make a paragraph red



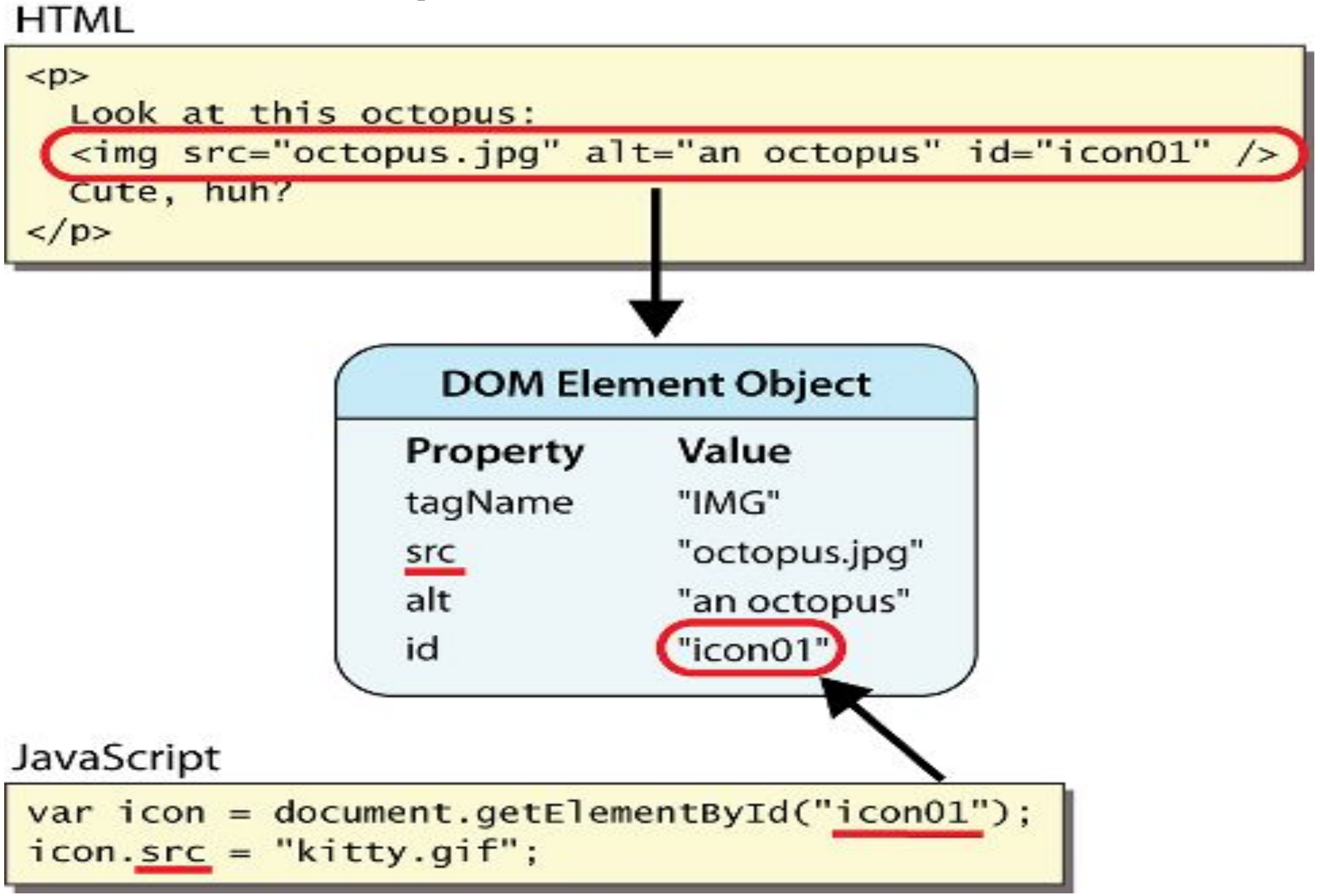
DOM.. continue

- The way a document content is accessed and modified is called the Document Object Model, or DOM.
- Objects are organized in a hierarchy.

I. **Window object**
II. **Document object**
III. **Form object**
IV. **Form control elements**



DOM element objects



HTML

```
<p>
  Look at this octopus:
  <img src="octopus.jpg" alt="an octopus" id="icon01" />
  Cute, huh?
</p>
```

DOM Element Object

Property	**Value**
tagName	"IMG"
src	"octopus.jpg"
alt	"an octopus"
id	"icon01"

JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

Accessing elements: `document.getElementById`

```
var name = document.getElementById("id");
                                             JS
```

```
<button onclick="changeText();">Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />          HTML
```

```
function changeText() {
   var span = document.getElementById("output");
   var textBox = document.getElementById("textbox");

    textbox.style.color = "red";

}                                            JS
```

Accessing elements: `document.getElementById`

- document.getElementById returns the DOM object for an element with a given id
- can change the text inside most elements by setting the innerHTML property
- can change the text in form controls by setting the value property

Comments (same as Java)

```
// single-line comment
/* multi-line comment */
                JS
```

- identical to Java's comment syntax
- recall: 4 comment syntaxes
  - HTML: <!-- comment -->
  - CSS/JS/PHP: /* comment */
  - Java/JS/PHP: // comment
  - PHP: # comment

Variables

```
var name = expression;                    JS
```

```
var clientName = "Connie Client";
var age = 32;
var weight = 127.4;                           JS
```

- variables are declared with the var keyword (case sensitive)
- types are not specified, but JS does have types ("loosely typed")
  - `Number, Boolean, String, Array, Object, Function, Null, Undefined`
  - can find out a variable's type by calling `typeof`

JS Variables Rule set

- Local Vs Global variables.

```
var myVar = "global"; // Declare a global variable

function checkscope( ){
 var myVar = "local"; // Declare a local variable
 document.write(myVar);
 }
```

Rules:

I. Can not use keyword as name of variable. E.g. if, for, switch, var, let, return, goto, public, void, function, with, int, export, class, catch.

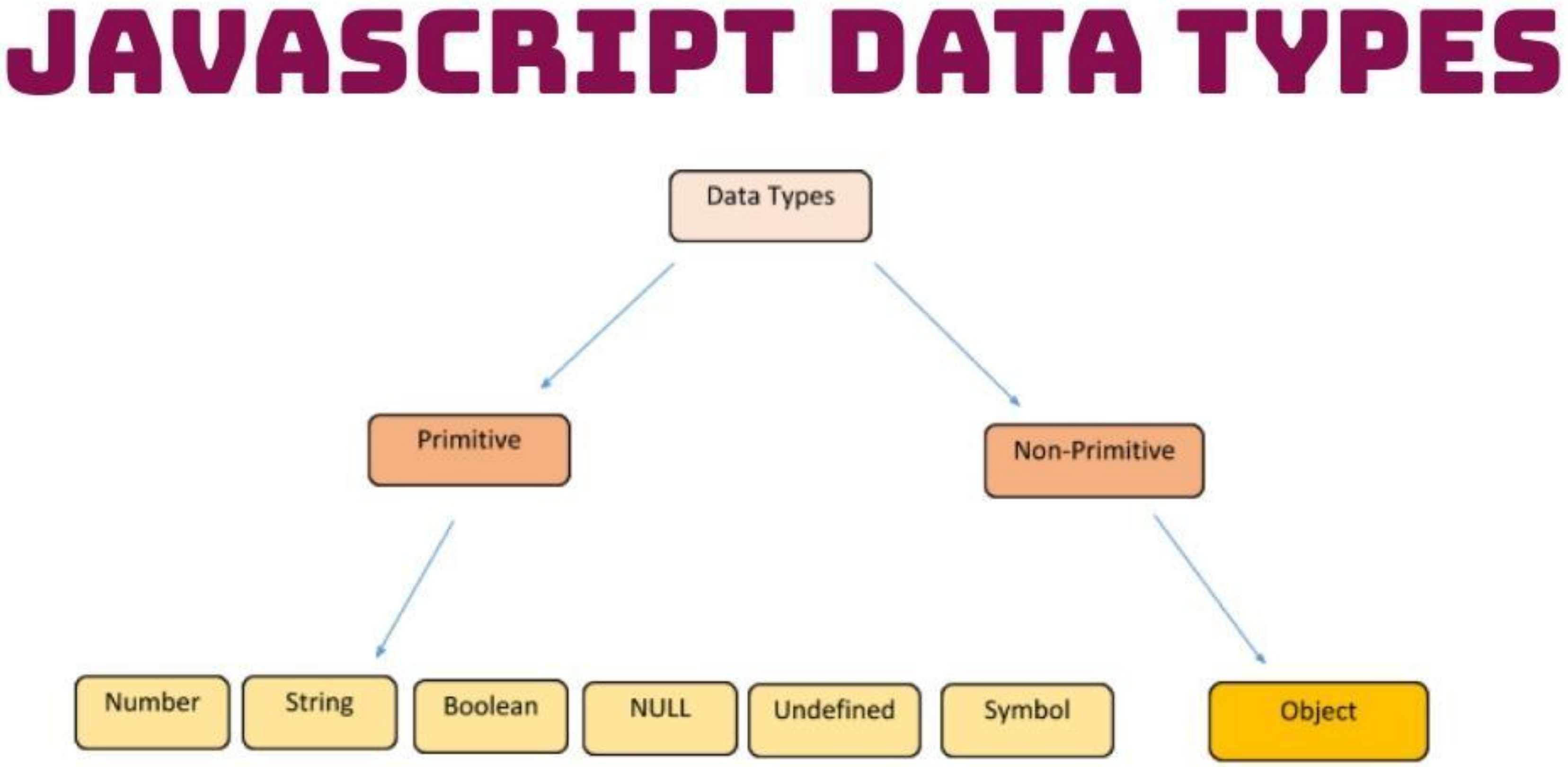
II. Should not start with numbers

III. Can start with '_'.

IV. Variable names are case-sensitive

Datatypes in JS

Most fundamental characteristics of any programming language is data types.



JS Operators

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

1. Arithmetic Operators

- Addition
- Subtraction
- Multiplication
- Division
- Modulus : Outputs the remainder of an integer division.
- Increment: X++
- Decrement: Y--

2. Comparison Operators

- Equal : ==
- Not Equal: !=
- Greater than: >
- Less than: <
- Greater than or Equal to: >=
- Less than or Equal to: <=

3. Logical Operators

Logical Operators returns Boolean output.

- Logical AND: &&
- Logical OR: ||
- Logical NOT: !

5. Assignment Operators

- Simple Assignment : =
- Add and Assignment: +=
- Subtract and Assignment: -=
- Multiply and Assignment: *=
- Divide and Assignment: /=
- Modules and Assignment: %=

6. Miscellaneous Operator

- Conditional operator:

  If Condition is true? Then value X : Otherwise value Y

- typeof operator: unary operator

```
var abc= ‘ClariTech’;
console.log(typeof(abc);
```

Conditional Statements in JS

- If.. Else
- Switch
- While
- For loop
- For.. In
- Loop Control i.e. Break continue

1. If… Else

- if statement
- if...else statement
- if...else if... statement.

```
if (expression) { Statement(s) to be executed if expression
is true } else { Statement(s) to be executed if expression
is false }
```

if/else statement (same as Java)

```
if (condition) {
    statements;
} else if (condition) {
    statements;
} else {
    statements;
}
```
JS

- identical structure to Java's if/else statement
- JavaScript allows almost anything as a condition

2. Switch Case

The interpreter checks each **case** against the value of the expression until a match is found. If nothing matches, a **default** condition will be used.

```
 switch (expression) {
   case condition 1: statement(s)
   break;
   case condition 2: statement(s)
   break;
   ...
   case condition n: statement(s)
   break;

   default: statement(s)
}
```

3. While… do…While

while loop is to execute a code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

Syntax: while

```
while (expression) {
   Statement(s) to be executed if expression is true
}
```

Syntax: do…while

```
do {
   Statement(s) to be executed;
} while (expression);
```

while loops (same as Java)

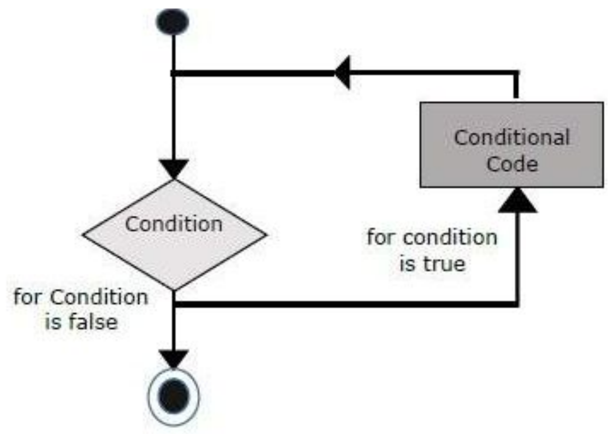
```
while (condition) {
    statements;
}                                        JS
```

```
do {
    statements;
} while (condition);
                        JS
```

- break and continue keywords also behave as in Java

4. For loop

- Loop initialization
- Test statement
- Iteration statement



```
for (initialization; test condition; iteration statement) {
Statement(s) to be executed if test condition is true
}
```

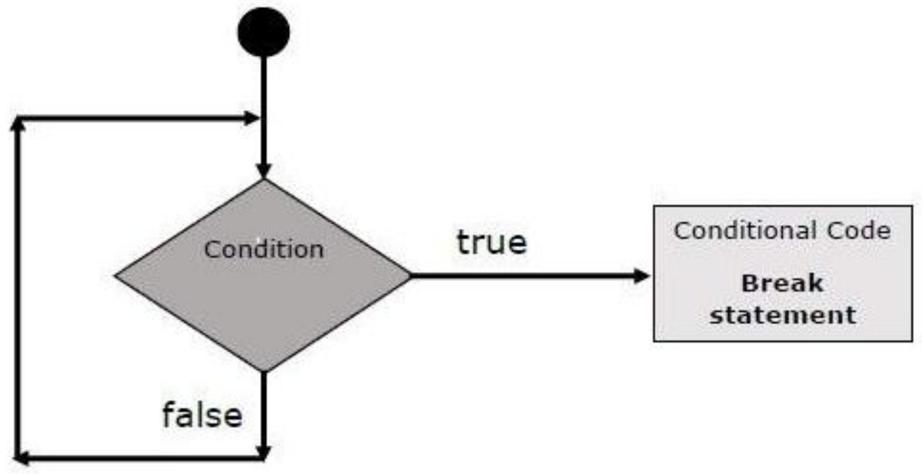
for loop (same as Java)

```
var sum = 0;
for (var i = 0; i < 100; i++) {
    sum = sum + i;
}                                    JS
```

```
var s1 = "hello";
var s2 = "";
for (var i = 0; i < s.length; i++) {
    s2 += s1.charAt(i) + s1.charAt(i);
}
// s2 stores "hheelllloo"                         JS
```

5. Loop controls

- **Break:**
  used to exit a loop early.



Continue:
The **continue** statement tells the interpreter to immediately start the next iteration of the loop and skip the remaining code block.

JS Functions

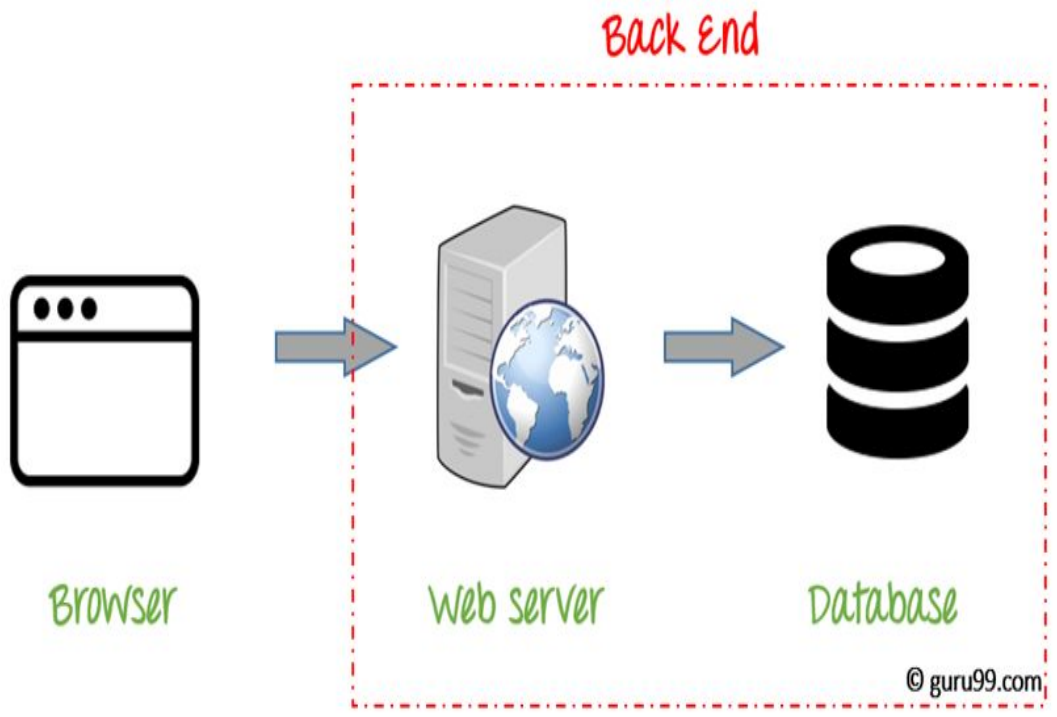
- reusable code which can be called anywhere in your program.
- Function Declaration
- Function Definition
- Function Argument
- Function return statement
- Constructor of Function
- Built in Function and Custom Function

JS Events

- Events are a part of the Document Object Model (DOM) Level 3
- When JavaScript is used in HTML pages, JavaScript can **"react"** on these events.
- every HTML element contains a set of events which can trigger JavaScript Code.
- Onclick, onchange, onmouseover, onmouseout, onkeydown
- Ondblclick, onfocus, onkeypress, onkeyup,

Storage Options in JS

- Cookies
- Local Storage
- Sessions



1. Cookies

e.g. Facebook

Syntax:

document.cookie = "cookiename=cookievalue; expires= Thu, 21 Aug 2014 20:00:00 UTC“;

- Limitations:

I. Security
II. Size(4kb and 20 cookies per site)
III. User can disable cookies
IV. Allow only plain text data.

2. Local storage

Type of Web storage API in HTML5

- More Secure.
- More data can store
- Simpler way to store the data.

Syntax: localStorage.setItem('user', JSON.stringify(user)); // set data

```
localStorage.getItem(‘user’); //get the data
localStorage.removeItem('user'); // delete data.
```

- Limitations:
  Can store only string data.

3. Session Storage

- Web API Option
- Same as local storage only difference is, Once the user closes that browser tab, the data is cleared.

  Syntax:

  ```
  sessionStorage.setItem('user', JSON.stringify(user));// set data
  sessionStorage.getItem('user') // get data
  sessionStorage.removeItem('user'); // remove data
  ```

Navigation in JS

- If user want to redirect to some another page then we can use this options.
  E.g.:
  window.location = "https://angular.io/";

Dialog Boxes in JS

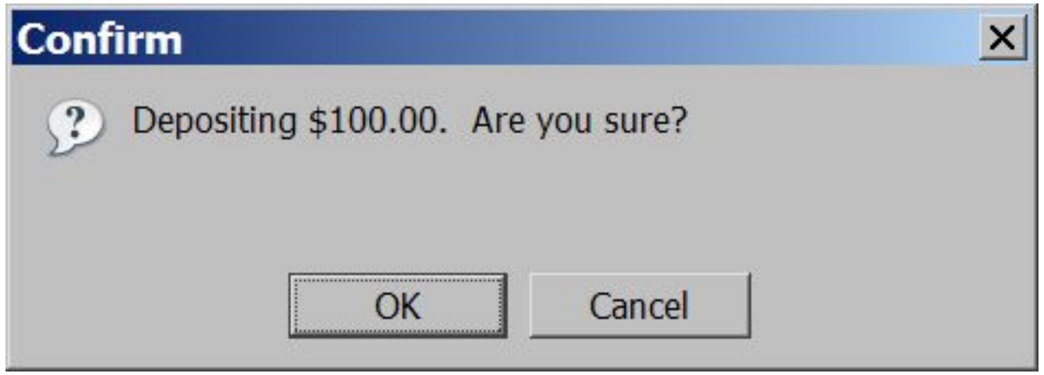
1. Alert: Provide only one option.
   Used for warning message.
   Gives only one button "OK" to select and proceed.

2. Confirm: Provide two options- OK and Cancel.
   If the user clicks on the OK button, the window method **confirm()** will return true. If the user clicks on the Cancel button, then **confirm()** returns false.

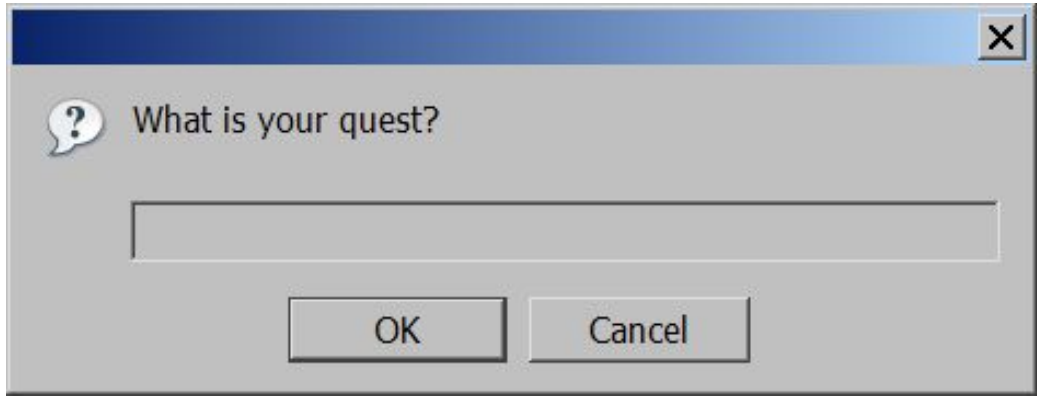
3. Prompt: Provide two options- OK and Cancel.
   If the user clicks the OK button, the window method **prompt()** will return the entered value from the text box. If the user clicks the Cancel button, the window method **prompt()** returns **null**.

Popup boxes

```
alert("message"); // message
confirm("message"); // returns true or false
prompt("message"); // returns user input string
                                        JS
```



Depositing $100.00. Are you sure?



What is your quest?

Data Types

- Primitive Data Types

I. Number
II. Boolean
III. String
IV. Null
V. Undefined

- Non Primitive Data types

I. Objects.

Number type

```
var enrollment = 99;
var medianGrade = 2.8;
var credits = 5 + 4 + (2 * 3);
                          JS
```

- integers and real numbers are the same type (no int vs. double)
- same operators: + - * / % ++ -- = += -= *= /= %=
- similar precedence to Java
- many operators auto-convert types: "2" * 3 is 6

Boolean type

```
var iLike190M = true;
var ieIsGood = "IE6" > 0; // false
if ("web devevelopment is great") { /* true */ }
if (0) { /* false */ }
                    JS
```

- any value can be used as a Boolean
  - "falsey" values: 0, 0.0, NaN, "", null, and undefined
  - "truthy" values: anything else
- converting a value into a Boolean explicitly:
  - `var boolValue = Boolean(otherValue);`
  - `var boolValue = !!(otherValue);`

Special values: null and undefined

```
var ned = null;
var benson = 9;
// at this point in the code,
// ned is null
// benson's 9
// caroline is undefined
                   JS
```

- `undefined` : has not been declared, does not exist
- `null` : exists, but was specifically assigned an empty or null value
- Why does JavaScript have both of these?

`String` type

```
var s = "Connie Client";
var fName = s.substring(0, s.indexOf(" ")); // "Connie"
var len = s.length; // 13
var s2 = 'Melvin Merchant';
                  JS
```

- methods: `charAt, charCodeAt, fromCharCode, indexOf, lastIndexOf, replace, split, substring, toLowerCase, toUpperCase, search, slice, splice, split, substring,`
    - charAt returns a one-letter String (there is no char type)
- length property (not a method as in Java)
- Strings can be specified with "" or ''
- concatenation with + :
    - 1 + 1 is 2, but "1" + 1 is "11"

More about `String`

- escape sequences behave as in Java: \' \" \& \n \t \\
- converting between numbers and Strings:

```
var count = 10;
var s1 = "" + count; // "10"
var s2 = count + " bananas, ah ah ah!"; // "10 bananas, ah
ah ah!"
var n1 = parseInt("42 is the answer"); // 42
var n2 = parseFloat("booyah"); // NaN                  JS
```

- accessing the letters of a String:

```
var firstLetter = s[0]; // fails in IE
var firstLetter = s.charAt(0); // does work in IE
var lastLetter = s.charAt(s.length - 1);                  JS
```

Splitting strings: split and join

```
var s = "the quick brown fox";
var a = s.split(" "); // ["the", "quick", "brown", "fox"]
a.reverse(); // ["fox", "brown", "quick", "the"]
s = a.join("!"); // "fox!brown!quick!the"
```
JS

- split breaks apart a string into an array using a delimiter
  - can also be used with regular expressions (seen later)
- join merges an array into a single string, placing a delimiter between them

Arrays

```
var name = []; // empty array
var name = [value, value, ..., value]; // pre-filled
name[index] = value; // store element
                     JS
```

```
var ducks = ["Huey", "Dewey", "Louie"];
var stooges = []; // stooges.length is 0
stooges[0] = "Larry"; // stooges.length is 1
stooges[1] = "Moe"; // stooges.length is 2
stooges[4] = "Curly"; // stooges.length is 5
stooges[4] = "Shemp"; // stooges.length is 5
                     JS
```

Array methods

```
var a = ["Stef", "Jason"]; // Stef, Jason
a.push("Brian"); // Stef, Jason, Brian
a.unshift("Kelly"); // Kelly, Stef, Jason, Brian
a.pop(); // Kelly, Stef, Jason
a.shift(); // Stef, Jason
a.sort(); // Jason, Stef
                   JS
```

- array serves as many data structures: list, queue, stack, ...
- methods: `concat, join, pop, push, reverse, shift, slice, sort, splice, toString, unshift, forEach, join,`
  - push and pop add / remove from back
  - unshift and shift add / remove from front
  - shift and pop return the element that is removed

Math object

```
var rand1to10 = Math.floor(Math.random() * 10 + 1);
var three = Math.floor(Math.PI);
```
JS

- methods: `abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan`
- properties: `E, PI`

JS Objects

- JS is partially OOPs based language.

I. Encapsulation: Can store property or methods together.
II. Inheritance: can access property or methods of other class.
III. Abstraction: can store one object inside another object.
IV. Polymorphism: write once use multiple time.

- Objects composed of attributes i.e. methods or property.

Object.. Continue

- **Object Properties:**

I. Properties are like variables which contain primitive datatypes.

II. Properties are internally used in the methods.
Syntax: objectName.objectProperty = propertyValue;
e.g. : student.name=‘Alex’;

- **Object Methods:**

I. This is a function to do some operation.

II. Same as function but difference is function is a standalone unit of statements and a method is attached to an object.

III. can be referenced by the **this** keyword.

Object.. Continue

- **Object() Constructor:**

I. Mainly constructor used to create object and initializes an object.

-