Learning JavaScript

What is recommended to learn before diving deep with Node.js?

- Lexical Structure
- Expressions
- Data Types
- Classes
- Variables
- Functions
- `this` operator
- Arrow Functions
- Loops
- Scopes
- Arrays
- Template Literals
- Strict Mode
- ECMAScript 2015 (ES6) and beyond

Introduction to JavaScript

JavaScript is a *lightweight, cross-platform*, *single-threaded,* and *interpreted compiled* programming language.

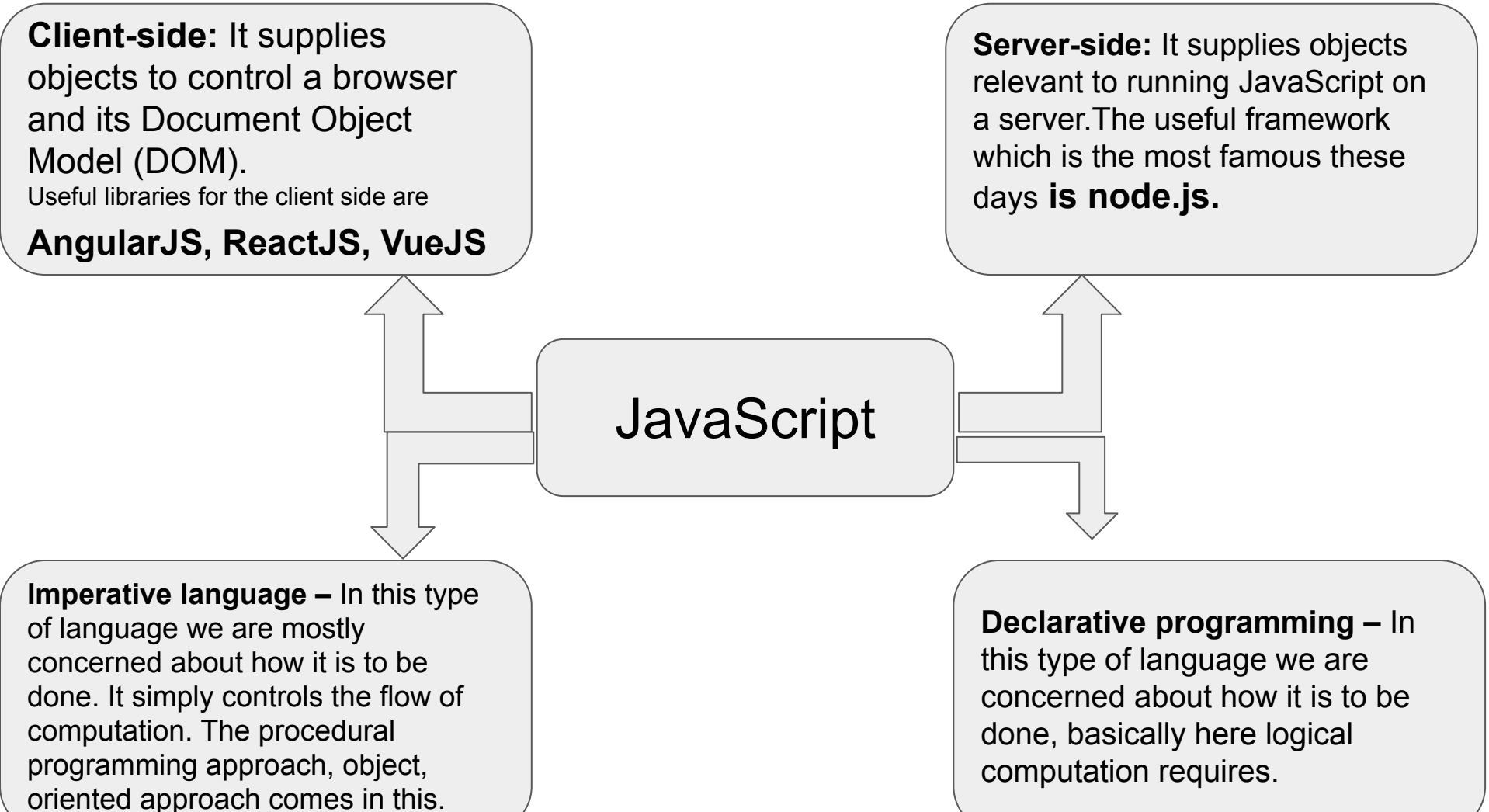
It is also known as the scripting language for web pages.

It is well-known for the development of web pages.

JavaScript is a **weakly typed language (dynamically typed)**.

JavaScript can be used for **Client-side** developments as well as **Server-side** developments.

JavaScript is both an imperative and declarative type of language.


Client-side: It supplies objects to control a browser and its Document Object Model (DOM).
Useful libraries for the client side are
AngularJS, ReactJS, VueJS
Server-side: It supplies objects relevant to running JavaScript on a server.The useful framework which is the most famous these days **is node.js.**
JavaScript
Imperative language – In this type of language we are mostly concerned about how it is to be done. It simply controls the flow of computation. The procedural programming approach, object, oriented approach comes in this.
Declarative programming – In this type of language we are concerned about how it is to be done, basically here logical computation requires.

The V8 JavaScript Engine

V8 is the name of the JavaScript engine that powers Google Chrome. It's the thing that takes our JavaScript and executes it while browsing with Chrome.

V8 is the JavaScript engine i.e. it parses and executes JavaScript code. The DOM, and the other Web Platform APIs (they all makeup runtime environment) are provided by the browser.

Compilation

JavaScript is generally considered an interpreted language, but modern JavaScript engines no longer just interpret JavaScript, they compile it.

This has been happening since 2009, when the SpiderMonkey JavaScript compiler was added to Firefox 3.5, and everyone followed this idea.

JavaScript is internally compiled by V8 with **just-in-time** (JIT) **compilation** to speed up the execution.

Declaring variables and their scope

Let : Variables declared with let have **block** scope,must be **Declared** before use cannot be **Redeclared** in the same scope.

Var : Variables declared with the `var` always have **Global Scope**.can NOT have block scope.

JavaScript Functions

Syntax : function *name*(*parameter1, parameter2, parameter3*) {

// code to be executed

}

Example :

```
let x = Add_nos(4, 3);
```

function call

```
function Add_nos(no1,no2) {
```


function declaration

```
// Function returns the product of a and b
  return no1+no2;
```


returning values

```
}
```

JavaScript Can Change HTML Content

The example below "finds" an HTML element (with id="demo"), and changes the element content (innerHTML) to "Hello JavaScript":

```
document.getElementById("demo").innerHTML = "Hello JavaScript";

document.getElementById("demo").style.fontSize = "35px";

document.getElementById("demo").style.display = "none";

document.getElementById("demo").style.display = "block";
```

The <script> Tag

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

```
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script> </head>
<body>
      <h2>Demo JavaScript in Head</h2>
      <p id="demo">A Paragraph.</p>
      <button type="button" onclick="myFunction()">Try it</button>
</body> </html>
```

The <script> Tag in body section

```
<html> <body>

<h2>Demo JavaScript in Body</h2>
<p id="demo">A Paragraph.</p>
<button type="button" onclick="myFunction()">Try it</button>
<script>
      function myFunction() {
            document.getElementById("demo").innerHTML = "Paragraph
      changed.";
      }
</script>
</body>    </html>
```

The script tag : external files

```
<html><body>

<h2>Demo External JavaScript</h2>

      <p id="demo">A Paragraph.</p>

      <button type="button" onclick="myFunction()">Try it</button>

      <p>This example links to "myScript.js".</p>
      <p>(myFunction is stored in "myScript.js")</p>

      <script src="myScript.js"></script>

</body></html>
```

External scripts cannot contain `<script>` tags.

External file: myScript.js

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed."; }
```

The source can be URL Link

JavaScript Output

JavaScript can "display" data in different ways:

- Writing into an HTML element, using `innerHTML`.
- Writing into the HTML output using `document.write()`.
- Writing into an alert box, using `window.alert()`.
- Writing into the browser console, using `console.log()`.

```
<script>

document.getElementById("demo").innerHTML = 5 + 6;

</script>
```

Client-side:
It supplies objects to control a browser and its Document Object Model (DOM).
Useful libraries for the client side are **AngularJS**, **ReactJS**, **VueJS**

JavaScript