JavaScript

Brendan Eich

The first ever JavaScript was created by **Brendan Eich** at Netscape

Scripts?

something written : the written text of a stage play, screenplay, or broadcast specifically : the one used in production or performance.

Is “Java” + “Script” = Js

NO!!!, There's often some confusion about the two, but JavaScript and Java have almost nothing in common.

The name JavaScript came from Netscape's support of Java applets within its browser. Many say it was also a marketing tactic to divert some attention from Java, which was the most buzzed-about language at the time.

JavaScript

A lightweight programming language ("scripting language")

Java Script

JavaScript, often abbreviated JS, is a **client-side** scripting / programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.

JavaScript is the fastest growing technology in the world.

Over 97% of websites use JavaScript on the client side for web page behavior, often incorporating third-party libraries.

JavaScript is a high-level, often just-in-time compiled language or can say interpreted.

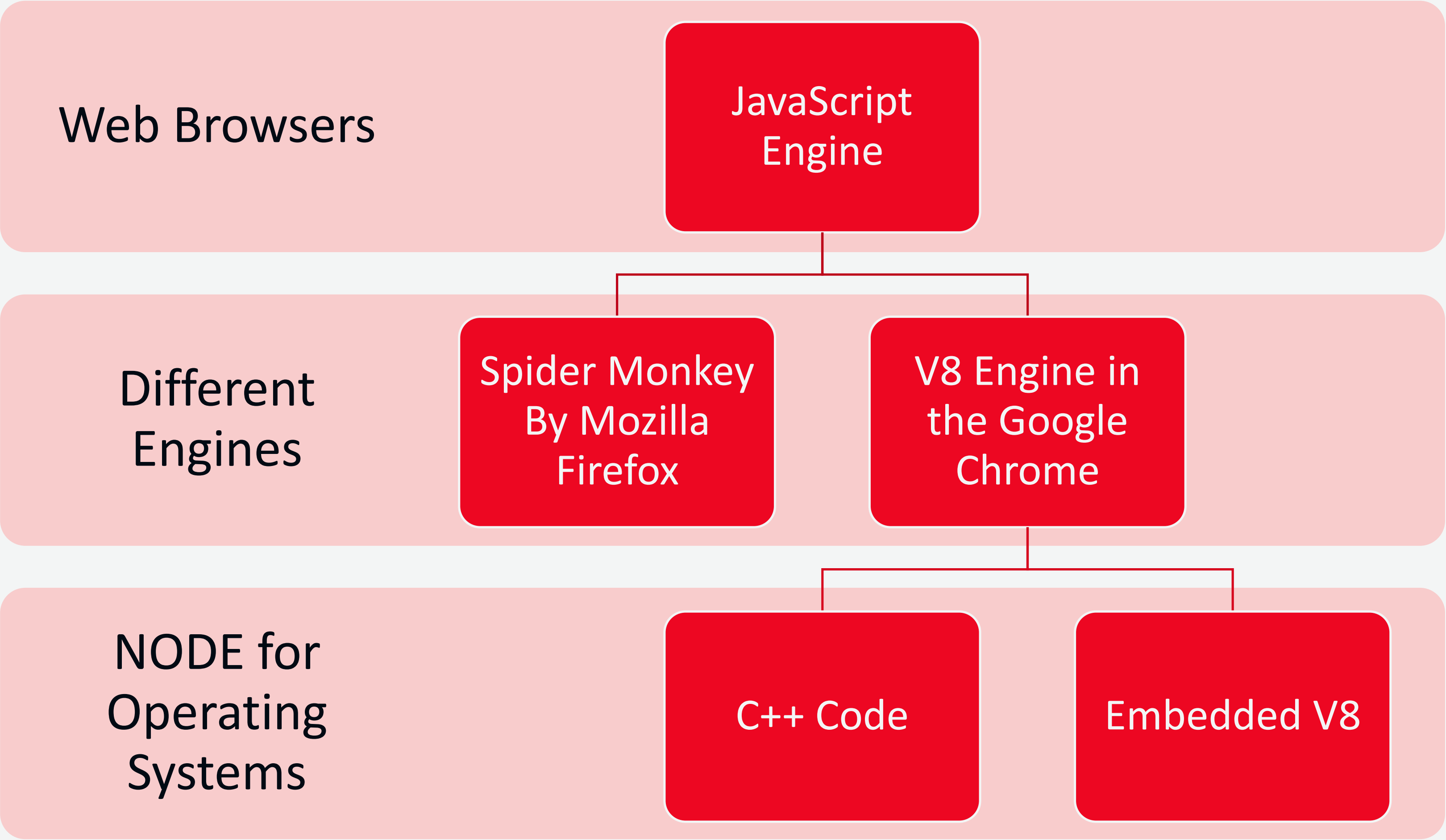
It is a web standard (but not supported identically by all browsers)

Organization ECMA defines standards and offers the Specification and JS follows that thus ECMAScript is the official name of the JavaScript language

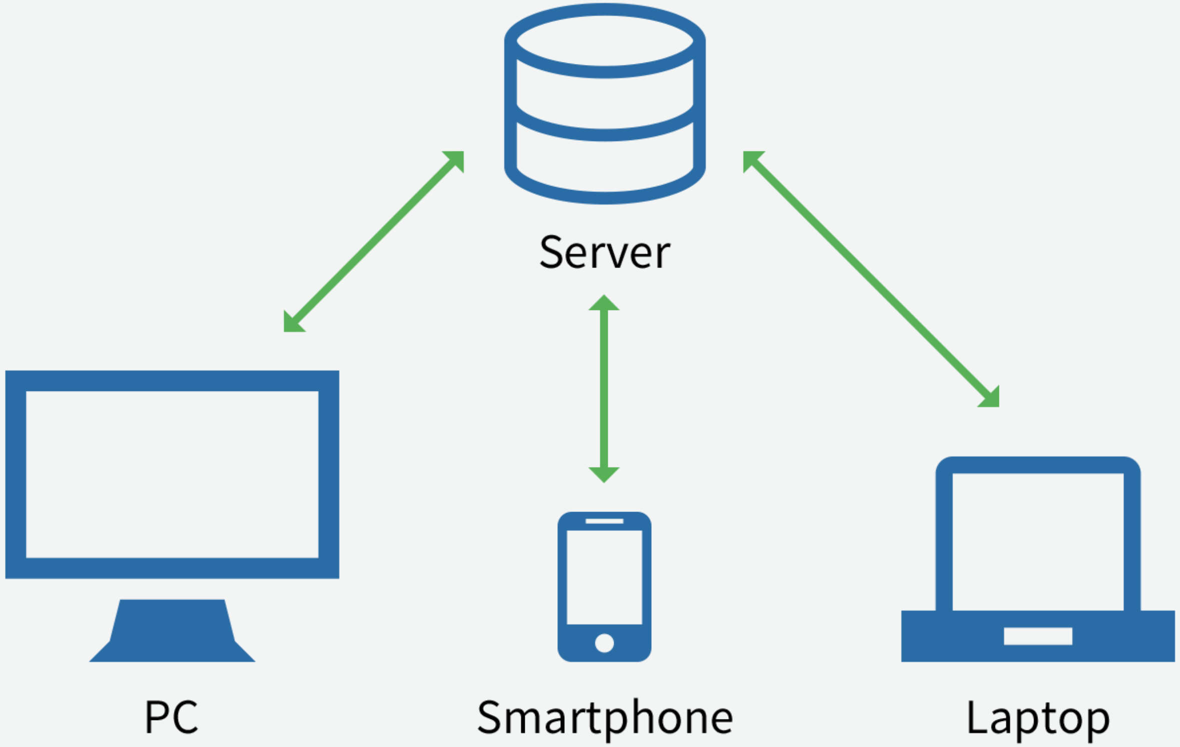
How does JavaScript Code run?

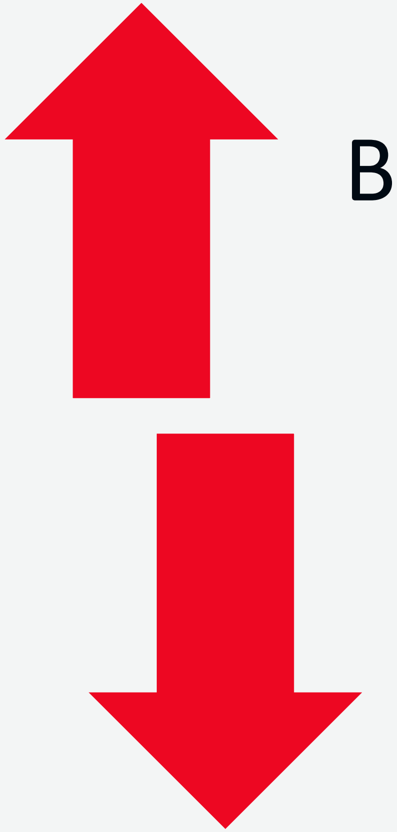
Parse JavaScript using Runtime Environment

In 2009 Ryan Dahl took the open-source engine of chrome and embedded it into C++ to create NODE.


Web Browsers
JavaScript Engine
Different Engines
Spider Monkey By Mozilla Firefox
V8 Engine in the Google Chrome
NODE for Operating Systems
C++ Code
Embedded V8

Client-Server Model





Why use client-side programming?

Usability:

- Can modify a page without having to post back to the server (faster UI)

Efficiency:

- Can make small, quick changes to page without waiting for server

Event-driven:

- Can respond to user actions like clicks and key presses

HTML, CSS & JS

Backbone of any web design

HTML is all about content & flow

CSS is all about layouts & presentation

JavaScript is all about behaviour & functionality

JavaScript Example

```
<!DOCTYPE html>
<html>
        <body>
                <h2>Example</h2>
                <p id=“example">Join Us</p>

                <button type="button" onclick='document.getElementById(“example").innerHTML = “Welcome to
                Careerera"'>Click Me!</button>

                <script>
                document.getElementById("example").innerHTML = “Yes Sure!";
                </script>

        </body>
</html>
```

JavaScript offers

- Variables
- Conditions
- Operations
- Loops
- Logics
- Functions
- Classes

ES6

Arrow functions were introduced in ES6.

Organization ECMA defines standards and offers the Specification and JS follows that thus ECMAScript is the official name of the JavaScript language

Frameworks

Vue.js
React
ember
BACKBONE.JS
ANGULAR

NEXT.JS

Next.js is an open-source web development framework built on top of Node.js enabling React based web applications functionalities such as server-side rendering and generating static websites.

What is TypeScript

TypeScript is a JavaScript "Syntactic Superset" with strongly typed code, developed and maintained by Microsoft offering the same base syntax as JavaScript but with added benefits.

Transpilers

Transpilers (also called transcompilers) in JavaScript are **source-to-source compilers that transform source code in non-JavaScript languages** (CoffeeScript, TypeScript, LiveScript, etc.)

Hands On

Quick Program

```
<!DOCTYPE html>
  <head>
        <title>Demo #1</title>
  </head>
  <body>
        <h1 id="test">Hi</h1>
    <script>
        console.log("Hi");
        alert("Hello");

        document.getElementById("test").innerHTML = "hello";
    </script>
  </body>
</html>
```

IDE & Console

VS Code and Chrome’s Console

- Internal
  - <script> and </script>
  - In Header or in footer
- External
  - <script src=“anyname.js” type="text/javascript"></script>

Where to put JavaScript

Placing scripts at the bottom of the <body> element improves the display speed, because script interpretation slows down the display.

Placement of JavaScript

Two most common reasons are that a page loads in a linear manner from top to bottom so if java script is on the top browser will search for it and will increase the time without even displaying any element resulting in bad user experience.

Second important reason is that most of the time JavaScript interacts with the DOM Components and thus placing it at bottom allows all the components to render first and then JavaScript can apply all the operations like show/hide or popup or validate and so.

Exceptions are always there from third party like Google Analytics or so.

So, Putting the scripts at the end of your html’s body can help browser can parse it at the end to enhance user experience

Comment

- Single line comments start with //
- Multi-line comments start with /* and end with */
- In Tag comments start with <!-- and end with //-->

Example

```
<script>
        <!--
        document.write("Hello World!")
        //-->
</script>
<noscript>Sorry, your browser does not support JavaScript!</noscript>
```

Dialog box

Alert is a JS command that pops up a dialog box with a message

```
<script>

    alert(“Hello I am alert”); - Just to display
    confirm(“Are you sure”); - Returns True or False
    prompt(“What is your Name?”); - Returns Null or
    User Input String
    prompt(“Who are you?”, “Jack”);

</script>
```

Output Display

```
document.getElementById("demo").innerHTML
```

```
HTML element, using .innerhtml.
Html output using document.Write().
Alert box, using window.Alert().
Browser console, using console.Log().
```

Variables

- Variables are containers for storing data so should be meaningful not like a = x or a22 = y or so.
- keywords Using var, let, const and nothing (undeclared) to declare variables.
  - Let can change or can be reasign but const can’t as it is constant
- All JavaScript identifiers (name of unique variable) are case sensitive
- Hyphens are not allowed in JavaScript
- Underscore, Upper Camel Case (Pascal Case) and Lower Camel Case allowed
- A JavaScript name must begin with:
  - A letter (A-Z or a-z) not with a number
  - A dollar sign ($)
  - Or an underscore (_) but can't contain space

#IMPORTANT – In jQuery (JavaScript library) $("p"); means "select all p elements

Variables Declaration

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`

```
var name = expression;
```

```
   var clientName = “The Client";
   var age = 41;
   var weight = 127.4;
```

```
var clientName = “The Client";
alert(clientName);
```

Primitive or Value Type

- String
- Number
- Boolean
- Undefined
- Null

Reference Type

- Object
- Array
- Function

integers and real numbers are the same type (no int vs. double)

String type

methods: charAt, charCodeAt, fromCharCode, indexOf, lastIndexOf, replace, split, substring, toLowerCase, toUpperCase

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

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
```

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);
```

Splitting strings: split and join

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

```
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"
```

Special values: null and undefined

- `undefined` : has not been declared, does not exist
- `null` : exists, but was specifically assigned an empty or null value

Operators

- operators: + - * / % ++ -- = += -= *= /= %=
- arithmetic operators (+ - * /) to compute values
- assignment operator (=) to assign values to variables
- Equal to operator (==) to compare values of variables
- Equal to operator (===) to compare strict values of variables

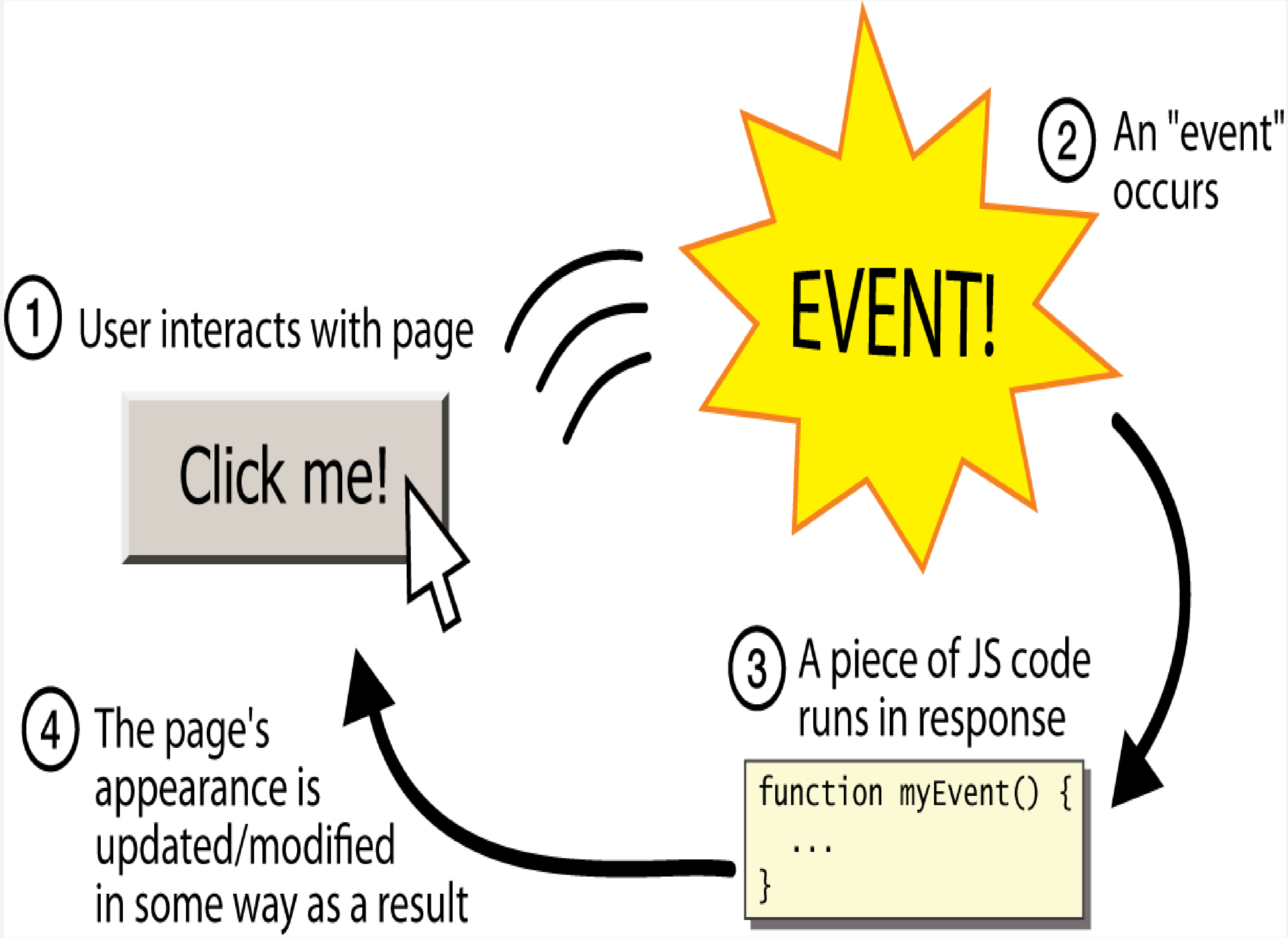
Logical operators

- > < >= <= && || ! == != === !==
- most logical operators automatically convert types:
  - 5 < "7" is true
  - 42 == 42.0 is true
  - "5.0" == 5 is true
- === and !== are strict equality tests; checks both type and value
  - "5.0" === 5 is false

Events

JavaScript programs wait for user actions called *events* and respond to them

event-driven programming: writing programs driven by user events



1 User interacts with page
Click me!
EVENT!
2 An "event" occurs
3 A piece of JS code runs in response
4 The page's appearance is updated/modified in some way as a result

```
function myEvent() {
  ...
}
```

Event handlers

- JavaScript functions can be set as event handlers
  - when you interact with the element, the function will execute
- onclick is just one of many event HTML attributes we'll use
- but popping up an alert window is disruptive and annoying
  - A better user experience would be to have the message appear on the page...

onClick Event

```
<input type=button onClick=myMessage() Value=Click >

     <script/>
        function myMessage()
            {
                alert("Hello!");
                document.write("<strong>Hello World</strong>");
            }
    </script>
```

JavaScript functions

```
function name() {
statement ;
statement ;
...
statement ;
}
```

```
function myFunction() {
   alert("Hello!");
   alert("How are you?");
}
```

Functions

```
function function_Name(parameters) {
  // your code here
      //action, display or return
}
<p id=“ex”></p>


<script>
function functionName() {
  document.getElementById(“ex").innerHTML = “Updated Text";
}
</script>


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

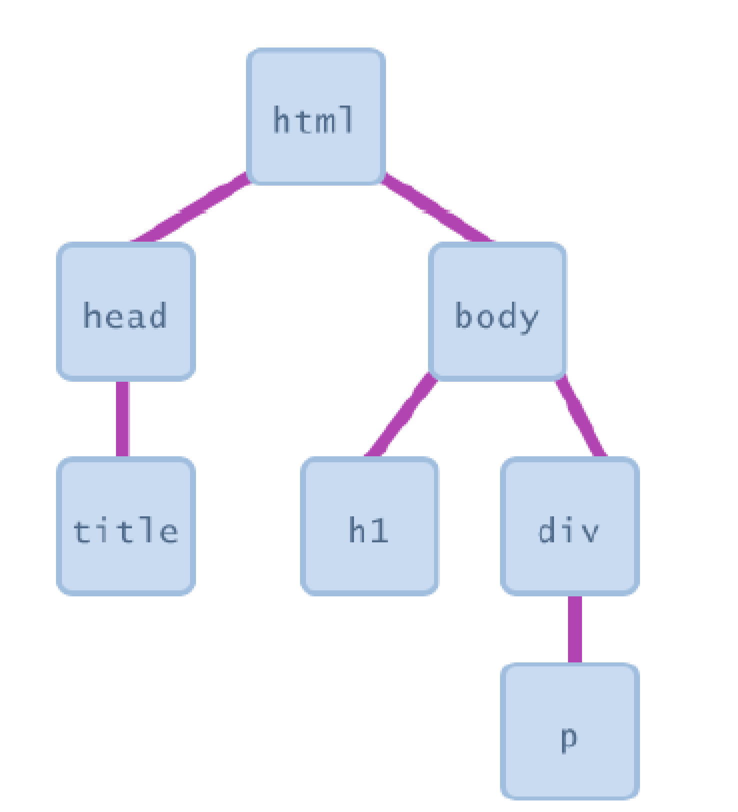
DOM

Document Object Model

Example DOM

```
 <script>
      var clientName = "The Client";
      alert(clientName);
      document.write('<p id="myId">Your Name will appear
here</p>');
      var clientName = prompt("What is your Name?");
      alert("Hello " + clientName);
      document.getElementById("myId").innerHTML = "Hello " +
clientName;
</script>
```

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



DOM element objects

```
<img src= "buddy.jpg" alt= "buddy pic" id= "unique_buddy" title = "buddy" />

<script>
  var bropic = document.getElementById("unique_buddy");
  bropic.alt = "New buddy pic";
  bropic.src = "newbuddy.jpg";
  bropic.title = "New buddy pic";
</script>
```

Property	Value
tagname	“img”
src	"buddy.jpg"
alt	"buddy pic"
title	"buddy"
id	"unique_buddy"

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 like

```
.innerHTML
.style
```

Changing element style:

`element.style`

Attribute	**Property or style object**
color	color
padding	padding
background-color	backgroundColor
border-top-width	borderTopWidth
Font size	fontSize
Font famiy	fontFamily

Example: DOM Element Style

```
<button onclick="changeText();">Click me!</button>
    <span id="output">replace me</span>
    <input id="textbox" type="text" />
    <script>
      function changeText() {
        var color = prompt("type any color");
        var textBox = document.getElementById("textbox");
        textbox.style.color = color;
        textbox.style.fontSize = "16pt";
      }
    </script>
```

Array

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

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
```

Array methods

array serves as many data structures as possible: list, queue, stack, ...

```
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
```

if/else statement

JavaScript allows almost anything as a condition

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

for loop

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

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

while loops

```
while (condition)
      {
      statements;
      }


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

Math object

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

Thanks

In case of any assistance,

please feel free to contact us at

info@careerera.com