



I see a bunch of text with no idea where most of it came from or why it is there and, in some cases, don't even know what it's supposed to be doing...

<p>Remember what we are doing is typing instructions (code/scripts) for the machine to execute...</p> <p>--It isn't the "language" I write with but what I need to tell the machine to do or connect to – and what is needed to make that code run—</p>	<p>It's All Just</p> <div style="border: 1px solid black; padding: 10px; display: inline-block;"> <p>● → TEXT ●</p> </div> <p>(Universal·Programming·Language) DO·C·RU·D·with·Content·on·the·Web</p>
---	--

- For JavaScript it the Browser for the "Client" that runs code.
- Node.js runs JavaScript on the "Server" side without a browser.
- HTTP lets them "talk" to each other over the web.

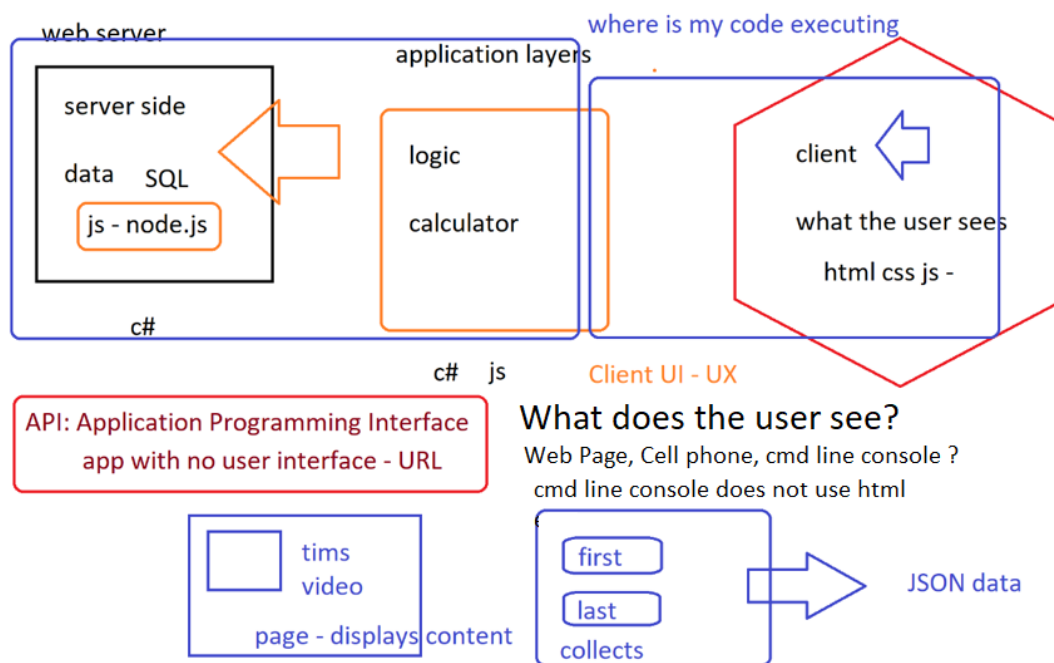
We build structures like Classes/Objects – Arrays to hold data so we can use it.

We write or use built in functions/methods/loops/ to do things (The [JavaScript Array Sort \(w3schools.com\)](https://www.w3schools.com/js/array_sort.asp) sorts an array alphabetically for example)

Events are the things computers or users do – we can "listen" for one and use it to run other code.

(mouseover for example - [W3Schools Tryit Editor](https://www.w3schools.com/js/tryit.asp))

What part am I working on, is the code running in the browser (client) or Node.js (server) – and what do I need to code to tell the computer to do things, run TSQL or hold data while doing things...



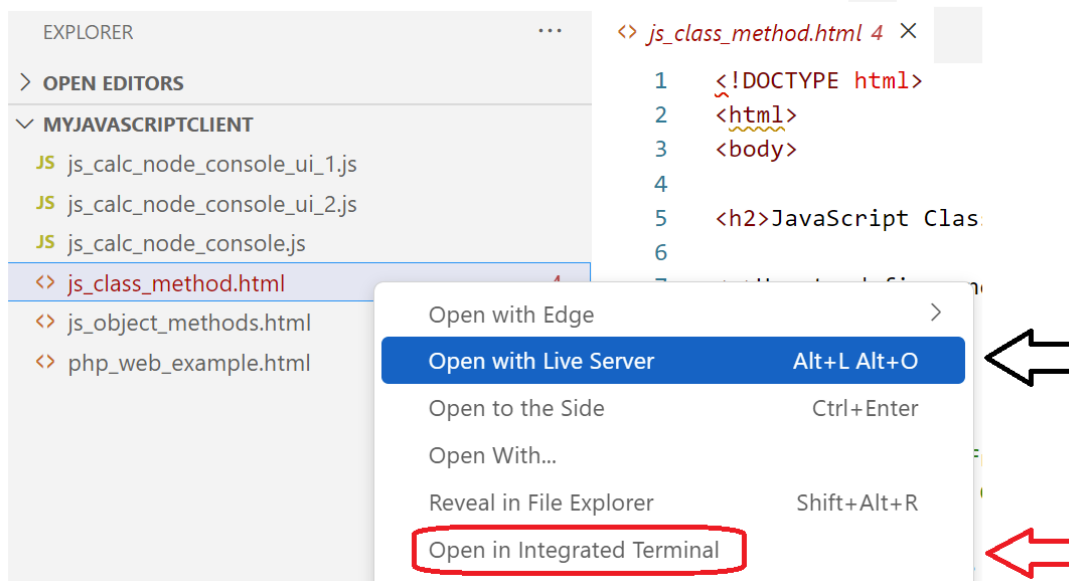
Build the calculator example as a Node.js console app and as a web app running in a web page. Code is here: [mrtime/html CSS JavaScript: General Notes for UI and UX \(github.com\)](https://github.com/mrtime/html-css-javascript-general-notes-for-ui-and-ux) unless you create your own.

Easiest first:

1. Create a folder to put each project inside with the command line terminal;
 - a. Finding files from a terminal or creating new projects.
 - b. [CMD: 11 basic commands you should know \(cd, dir, mkdir, etc.\) \(digitalcitizen.life\)](https://www.digitalcitizen.life/cmd-11-basic-commands-you-should-know-cd-dir-mkdir-etc/)
 - c. When in that folder – typing – code . – opens VS code with the files in that folder.
2. Node.js Console app
 - a. [basic-calculator-js - npm \(npmjs.com\)](https://www.npmjs.com/package/basic-calculator-js)
 - b. [How to create a CLI based calculator app using Node.js ? - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-create-a-cli-based-calculator-app-using-node-js/)
 - i. Discusses each step – finished code file at the end
 - c. [calculator.js - NodeJS - OneCompiler](https://www.onecompiler.com/tutorial/nodejs/calculator.js)
 - i. Console with choices for user input.
 - d. [Very Basic NodeJS Calculator \(github.com\)](https://github.com/VeryBasicNodeJS/Calculator)
 - i. Console with choices for user input.
 - e. [How to Create a Node.js Module \(tutorialkart.com\)](https://www.tutorialkart.com/tutorial/how-to-create-a-node-js-module/)
 - i. Create a Calculator Node.js Module with functions add, subtract and multiply. And use the Calculator module in another Node.js file.
3. Web page with styles: [JavaScript Calculator \(codepen.io\)](https://codepen.io/javascript-calculator/)
 - a. Must create your own files – you can copy n paste to start.
4. [Node.js - RESTful API \(tutorialspoint.com\)](https://www.tutorialspoint.com/nodejs/nodejs_restful_api.htm)

Try your app....

In VS Code - Rt. Mouse the file - Open with Live Server for html
Integrated Terminal for .js files



What makes anything work....

In JavaScript, almost "everything" is an object.

- Booleans can be objects (if defined with the `new` keyword)
- Numbers can be objects (if defined with the `new` keyword)
- Strings can be objects (if defined with the `new` keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

All JavaScript values, except primitives, are objects.

Typical "Client side": In a web browser

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page
- How to change the content of HTML elements
- How to change the style (CSS) of HTML elements
- How to react to HTML DOM events
- How to add and delete HTML elements

1. The use and function of (in Node.js or any other programming text):

- Classes & Objects** are very similar when looking at the code.
 - Object describe something to the computer in name:value pairs between curly braces {}
 - Writing code as an object allows us to get to a property or method with "dot notation" – meaning - person.name - for example.
 - Can be made as "re-usable" blueprints (what a "class" is) with internal variables.
 - [JavaScript Objects \(w3schools.com\)](https://www.w3schools.com/js/objects.asp) – watch for how code is written.
- ECMAScript 2015, also known as ES6, introduced JavaScript Classes
 - Classes are a template for creating objects.
 - When a class is defined (we write the code)
 - No memory is allocated
 - Memory is allocated when it is instantiated (created) with the "new" keyword.

<p>How to define and use a Class method.</p>

<p id="demo"></p>

<script>

//Create the class Car - note code is similar to an object but does not create anything in ram memory about the object yet.

```
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
}
```

//Age function - Creates a new Date object and getFullYear method gets Full Year from the Date object properties and subtracts this.year to get cars age.

```
age() {
  let date = new Date();
  return date.getFullYear() - this.year;
}
```

//Create a new myCar object from Car class - now myCar exists to the computer as an object - Ford passed to Car; 2014 is passed to age().

```
let myCar = new Car("Ford", 2014);
```

//Output to web page

```
document.getElementById("demo").innerHTML =
  "My car is " + myCar.age() + " years old.";
```

</script>

</body>

</html>

iv.

c. Methods

- i. A function I write or get from a module or package that does something – like make uppercase for example.
- ii. Where is it written?
 1. Inside the curly braces of an object?
 2. Inside a module that is imported?
- iii. By itself in code?

```
<!DOCTYPE html>
<html>
<body>
```

```
<h2>JavaScript Objects</h2>
<p id="demo1"></p>
```

```
<script>
const person = {
  firstName: "John",
  lastName: "Doe",
  id: 5566,
};
```

```
person.name = function() {
  return (this.firstName + " " +
    this.lastName).toUpperCase();
};
```

```
document.getElementById("demo1").innerHTML =
  "My father is " + person.name();
</script>
```

```
<p>Search a string for "W3Schools", and display the
position of the match:</p>
```

```
<p id="demo2"></p>
```

built in method .search

```
<script>
let text = "Visit W3Schools!";
let n = text.search("W3Schools");
document.getElementById("demo2").innerHTML = n;
</script>
</body>
</html>
```

JavaScript Objects

My father is JOHN DOE

Search a string for "W3Schools", and display the position of the match:

6

adds "name" function to person
object - called as method in code

- d. **2 common ways to create objects** (there are more) by how we write the code and how or what I code later that can change its values. Can I re-use it or have to write it more than once?

- i. Literal: Define and create an object in one statement – assigned to variable

```
const person = {firstName:"John", lastName:"Doe"};
```
- ii. The **new** keyword.
 1. **new** Car ("Ford", 2014);
- iii. [JavaScript Objects \(w3schools.com\)](https://www.w3schools.com/js/objects.asp) – ways to create JS object.

Break that down a bit more - Multiple things at the same time is right!

All apps are hybrid code apps today – while the focus is JavaScript; we will see other code or may need it (HTML for example) for a User Interface to show results. This list is for info only to put a name to what is common for understanding app building terms and a code hint.

2. **JavaScript:** A programming language that runs in a browser or Node.js on a server and does not need anything extra to run in a browser. Node.js, we installed to use.

Has many “Flavors” or “Framework” of the JavaScript ice cream – each of these **are JavaScript** but use external libraries or modules to make the code run – some help build html pages/apps; type less code to do the work.

- a. **JSON:** JavaScript Object Notation – data exchange; config files like app.json; Azure Resource Manager templates – etc..
 - i. Uses curly braces – name: value pairs
- b. **Angular:** Adds a way to pass content/data to html pages – just note this part.

```
<!DOCTYPE html>
<html>
  <script
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
  <body>
    <div ng-app="">
      <p>Name: <input type="text" ng-model="name"></p>
      <p>You wrote: {{ name }}</p>
    </div>
  </body>
</html>
```

needs external support

Name:

You wrote: Its me

What Angular adds to HTML:

A way to create an electronic box called inside <div ng-app> </div>

{{ holding spot for the content }}

- c. **JQuery:** A JavaScript Library that greatly simplifies JavaScript programming. Note the dollar sign - \$ - in front of each line.

//The dollar sign is an alias for the word JQuery

```
<!DOCTYPE html>
<html>
<head>
  <script
    src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
  <script>
    $(document).ready(function(){
      $("p").click(function(){
        $(this).hide();
      });
    });
  </script>
</head>
<body>
  <p>If you click on me, I will disappear.</p>
  <p>Click me away!</p>
  <p>Click me too!</p>
</body>
</html>
```

JQuery

needs external support

"traverses the DOM" to find <p>

click - hides the <p> and contents

i.

- d. **AJAX:** Asynchronous JavaScript and XML
- Read data from a web server - after the page has loaded
 - Update a web page without reloading the page
 - Send data to a web server - in the background
 - XMLHttpRequest() is what makes it work
 - [AJAX Introduction \(w3schools.com\)](https://www.w3schools.com/ajax/)
 - XHR is an old way make requests to the server. **Fetch API is a modern way to make requests to the server.** Chaining is difficult in XHR, but in fetch you can easily chain the promises. In XHR we need to check the status property for the errors, but in fetch the catch method is there to handle errors.

Async requests from browser to server

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Fetch a file to change this text.</p>

<script>
  getText("fetch_info.txt");
  async function getText(file) {
    let myObject = await fetch(file);
    let myText = await myObject.text();
    document.getElementById("demo").innerHTML = myText;
  }
</script>
</body>
</html>
```

Fetch - newest

both open files and move
content async

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
      this.responseText;
  }
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
</script>
</body>
</html>
```

XMLHttpRequest - older

- e. **React:** JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called "components".
- [Tutorial: Intro to React – React \(reactjs.org\)](https://reactjs.org/)

Not JavaScript but need good knowledge when you work with web pages.

- HTML:** Hypertext Markup Language: Builds web pages – each pair of tags/elements like <div></div> can be located (traversing the DOM – Document Object Model) and then inside those tags we put things – pictures, data etc.
 - No HTML tags – nothing shows that is meaningful with JavaScript.
 - [JavaScript HTML DOM \(w3schools.com\)](https://www.w3schools.com/html/html_dom.asp)
- CSS:** Cascade Style Sheets: Positions HTML elements and adds colors, fonts – everything that make a web page look good.
 - NOT REQUIRED for JavaScript but often see it in a web page.
 - [CSS Borders \(w3schools.com\)](https://www.w3schools.com/css/)

```

<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  border: 2px solid red;
  padding: 5px;
}

p.round {
  border: 2px solid red;
  border-radius: 12px;
  padding: 5px;
}
</style>
</head>
<body>
<p class="normal">Normal
border</p>

<p class="round">Round
border</p>
</body>
</html>

```

CSS: Cascade Style Sheets

Note this one inside <style></style>

This example adds the border -
Can do more like round corners - style
one side only..

Looks like JSON - name : value pairs

Normal border

Roundest border

p.normal - "Traversed the DOM" to
find <p class="normal">

3. **SQL**: Standard for C.R.U.D. Operations on MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.
 - a. **SELECT * FROM** Customers; //for example

Frameworks include jQuery, Angular, React – we only use them if needed –



















4. **ASP**: Active Server Pages - a framework for building applications from Microsoft.
 - a. Classic ASP: typically, VB script – the rest are usually C/open source based.
 - b. ASP.NET Web Forms
 - c. ASP.NET MVC
 - d. ASP.NET Web Pages
 - e. ASP.NET API
 - f. ASP.NET Core
5. There are many Frameworks – designed to support the development of web applications including web services, web resources, and web APIs. This is just for info – working with any would be what – Parts & TEXT? Yep !

Web application framework Software

From sources across the web

Just a few... 2023

Not all

 Django BSD licenses	 Flask BSD licenses	 Angular MIT License
 Ruby on Rails MIT License	 Express.js MIT License	 Laravel MIT License
 Vue.js MIT License	 ASP.NET Apache License	 Meteor MIT License
 Play Framework Apache License	 ASP.NET Core MIT License	 Next.js MIT License
 Rocket MIT License	 jQuery GNU General Public License	 Svelte MIT License
 web2py GNU Lesser General Public License	 AngularJS MIT License	 CakePHP MIT License

6. **PHP**: a scripting language – doing same job as Node.js – we see it in use as a “server-side form handler” sometimes – so only to note similarities in its job and using variables/query string.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

Requires PHP to be installed

```
<?php
```

```
echo "My first PHP script!";
```

```
?>
```

```
</body>
```

```
</html>
```

7. Methods, classes, functions specifically - **JavaScript functions are executed in the sequence they are called. Not in the sequence they are defined.**
- We must “control” each of the parts an app might use or consume or display.
 - Because code executes top down – we must tell the computer to stop or pause or test if something is true or false - to make it do something else -then resume or whatever is supposed to happen.
 - “Listening” for events is how we can monitor to catch or manipulate something that happens either from or for the user – or the machines.

Events

Every action on a computer is an event. We build event-driven applications.

- [Node.js Events \(w3schools.com\)](#):
 - Listen on the server side: more for watching what requests/responses might need.
 - I could use to create empty pages and maybe [JavaScript DOM Nodes \(w3schools.com\)](#) - document.createElement – to add html
- User events in a browser
 - [JavaScript Events \(w3schools.com\)](#)
 - [JavaScript DOM Events \(w3schools.com\)](#)
 - [JavaScript Events Examples \(w3schools.com\)](#)
 - [JavaScript DOM EventListener \(w3schools.com\)](#)


```

<!DOCTYPE html>
<html>
<body>

<h2>JavaScript addEventListener()</h2>

<p>How to pass parameter values using addEventListener()
method.</p>

<p>Click the button to perform a calculation.</p>

<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>
let p1 = 5;
let p2 = 7;
document.getElementById("myBtn").addEventListener("click",
function() {
  myFunction(p1, p2);
});

function myFunction(a, b) {
  document.getElementById("demo").innerHTML = a * b;
}
</script>

</body>
</html>

```

web page shows

JavaScript addEventListener()

How to pass parameter values using
addEventListener() method.

Click the button to perform a calculation.

35

8. Why in the first example (Node.js) were we assigning events; **what was being accomplished by the code?**

We want to be able to call a function when something happens – **this example shows us the steps to “Wrap code with a function” in Node.js – that’s it – not fancy.**

- The [W3Schools Tryit Editor](#) – Node.js EventEmitter - example shows how to tie together the event and the code we want to run when the event happens on the “server side” – not in a browser for the “client”.
- Always needs these steps:
 - “Create event handler” puts the “event” in a function call code wise so we can use it later.
 - Assign event handler to event – glue them together.
 - Fire (execute – run) event.

```

var events = require('events');
var EventEmitter = new events.EventEmitter();

```

```

//Create an event handler:
var myEventHandler = function () {
  console.log('I hear a scream!');
}

```

I hear a scream!
console is the
cmd line
terminal

```

//Assign the eventhandler to an event:
eventEmitter.on('scream', myEventHandler);

```

```

//Fire the 'scream' event:
eventEmitter.emit('scream');

```

By "wrapping" `console.log('I hear a scream!')` with a function allows me to call it to run when I want.

c.

9. It seemed like the console would have displayed the text regardless without assigning an event in the first place.
- True** – without the function code wrapped around `console.log('I hear a scream!')` statement it would just print – sort of like loading a web page can run a script - but we want to tell it when to print instead.
 - [W3Schools Tryit Editor](#) – example with no function – prints immediately.

REPL: A quick way to test JavaScript code without having to create a file.

- [Node.js - REPL Terminal \(tutorialspoint.com\)](#)
- Read Eval Print Loop: a command is entered and the system responds with an output.
REPL can be started by simply running **node** on shell/console without any arguments as follows.

```
$ node
```

You will see the REPL Command prompt > where you can type any Node.js command –

```
$ node
>
```

Let's try a simple mathematics at the Node.js REPL command prompt –

```
$ node
> 1 + 3
4
```

e.

General discussions about apps

The App: MyAppDoesThisBetter	
On the “Server side” I can have:	On the “Client” side:
Web servers File servers Storage SQL servers API services Virtual Machines (VM) Network/Security/User Accounts Messaging IoT Devices Connections to Content Delivery Networks (CDN)	Console: Web Pages: Cell phones: Tablets: Gestures: BIO/Face Recognition: IoT Devices: GPS, robots, smart mobiles, smart refrigerators, smartwatches, smart fire alarms, smart door locks, smart bicycles, medical sensors, fitness trackers, smart security system, etc...
Whiteboard/Design: Server-side events	Client-side events
A web request runs code that opens a file, writes to it, then closes it – triggers message alert.	User clicks on “Show details: button; JS function toggles hide element to show element

Lots of connections – lots of data – example links

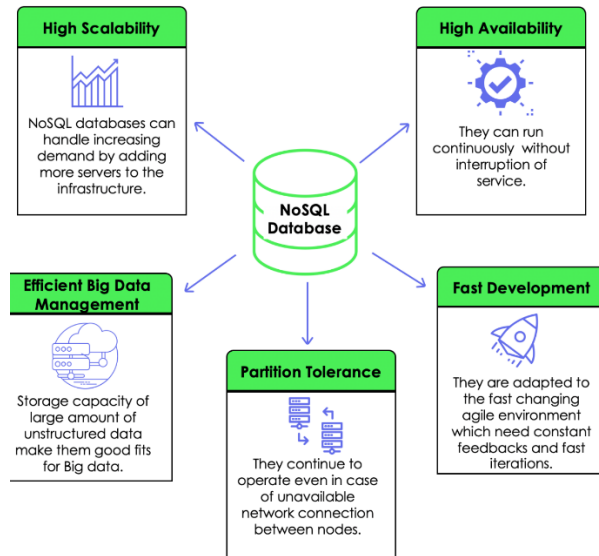
[MarineTraffic: Global Ship Tracking Intelligence | AIS Marine Traffic](#)

[18 Most Popular IoT Devices in 2023 \(Only Noteworthy IoT Products\) \(softwaretestinghelp.com\)](#)

Being a data scientist is not only about building machine learning models but also about being able to **process, analyze and better communicate your results from data in different formats.**

[Why the Hell Would I Use Node.js? A Case-by-case Tutorial | Toptal®](#)

- Real-time web applications employing push technology (chat; - wouldn't use Node.js to perform CPU-intensive operations
- Exposing data from NoSQL database (e.g., MongoDB). JSON-stored data allows Node.js to function without impedance mismatch and data conversion.
- [NoSQL - Wikipedia](#)
- [NoSQL Databases - Types of NoSQL Databases and How to Use Them | DataCamp](#)



- [Express.js And MongoDB REST API Tutorial | MongoDB](#)