React.js

Software’s required:

* Visual Studio Code Editor
* Node.js & NPM
* Permission to download node modules

Pre-requisites

* HTML
* CSS
* Javascript (ECMA Script new features)

Brief overview on the contents

1. React.js
2. JSX
3. ES6 & ES latest features
4. Virtual DOM
5. Components
6. Controlled Components
7. Props & States
8. HOC
9. Render Props
10. Advanced Concepts
11. Routers
12. Axios
13. Testing
14. Styling libraries
15. Redux

React.js

It is a Javascript library used to develop Single Page Applications, it uses component based approach

Components: These are independent piece of code which are reusable & they are the visible part in the web page.

What React.js uses to create single applications

* HTML
* CSS
* JSX (Superset of Javascript): This makes coding Javascript easier to create UI’s

Why JSX:

JSX stands for Javascript Extended language, it is easier to develop UI’s compare to Javascript, because Javascript is complex when it comes to writing HTML code in it.

Note: HTML, CSS & Javascript are the languages browser understands

HTML is mainly for writing presentation to the web page

CSS is mainly for adding styles to the HTML

Javascript is to add dynamic behaviour to the web page using HTML & CSS as well it provides many programming constructs like variables, functions, operators, conditional statements loops, objects and etc.

Node.js: It is a runtime environment to run the Javascript without browser so that you can use Javascript to write backend programs to perform IO operations, DB operations, Implementing Backend services

Javascript & New Features introduced in Javascript

Javascript & other Javascript run time environment follow the standard provided ECMA (European Computer Manufacturing Association) Script, so those standards are like rules every ECMA specification language must follow

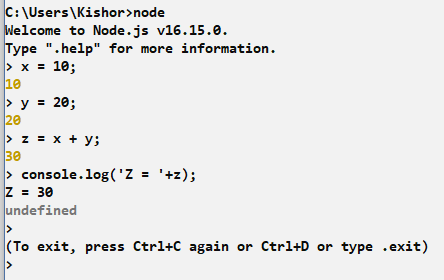
like Jscript, Java Script, Typescript, Browser, Node.js

ECMA Script provided many new features to improve the syntax of Javascript

1. Keywords like let, const, class, super, constructor, static, yield
2. Template Strings
3. Rest & Spread operators
4. Object Destructuring
5. Arrow functions
6. Generators
7. Exponential Operator
8. Trailing Commas
9. String padStart & padEnd
10. Optional Chain
11. Default parameters
12. Array includes
13. Object entries & values.

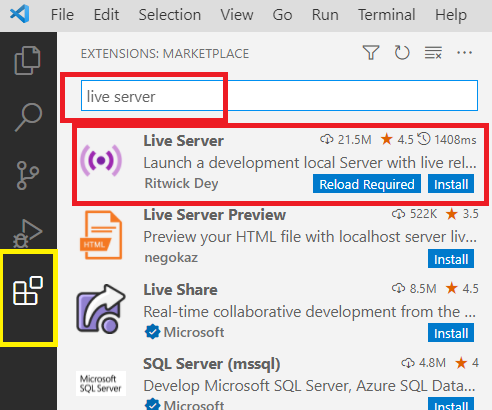
Javascripts are added in the HTML document using <script> tag and run on the browser or you can directly run the Javascript files using Node.js

Simple programs in node terminal

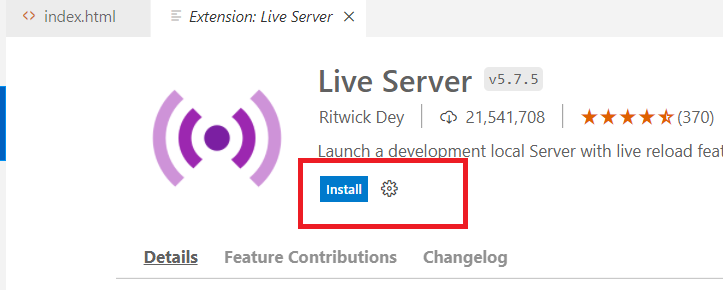


We can use Javascript files and add them in the HTML files and load it on the browser

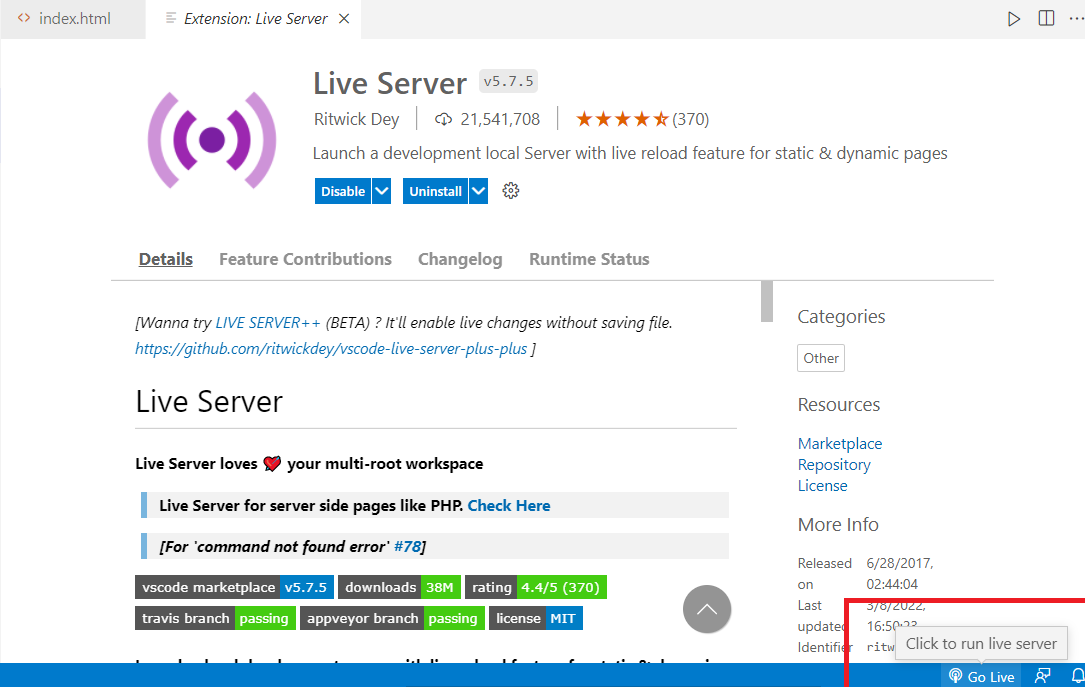
Adding Live Server plugin in the VS Code so that you get live reload feature while you save your code & don’t need to refresh



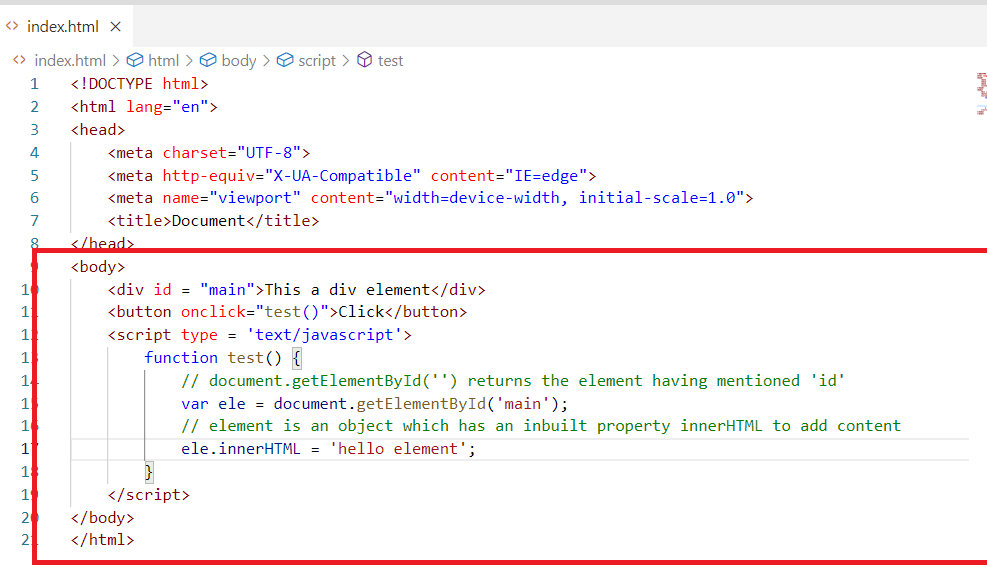
Click on Install



At the bottom right you see Go Live: Don’t click on it, instead you can select your HTML file and open with live server



index.html



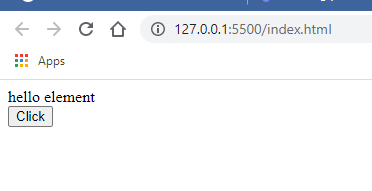
document is an inbuilt Javascript object, it provides many inbuilt properties & functions, some of the functions are getElementById(), getElementsByTagName(), write()

getElementById(): it returns the element object from which you can access many inbuilt properties of element object like innerHTML, style,

innerHTML: To access the content of the element

style: To access the style attribute of the element so that you can add CSS properties

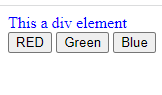
Output:



Using element object we can also add style attributes at runtime



Output:



New Features added in Javascripts

let & const keywords to declare variables: These are used to create block scoped variables,

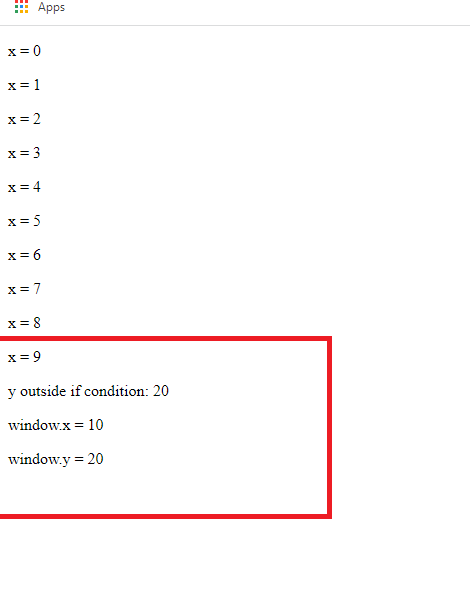
var keyword is not a block scoped variable, it creates in a global scope called window

window: It is a global object in Javascript, it is the top level object, every object is inside window object

When you declare any variable using var, then the variable will become part of window object, it wouldn’t be having any scope.



x & y are declared inside for loop, but you can access them outside the for loop as var creates the variable inside window object hence you can also access using window.x , window.y

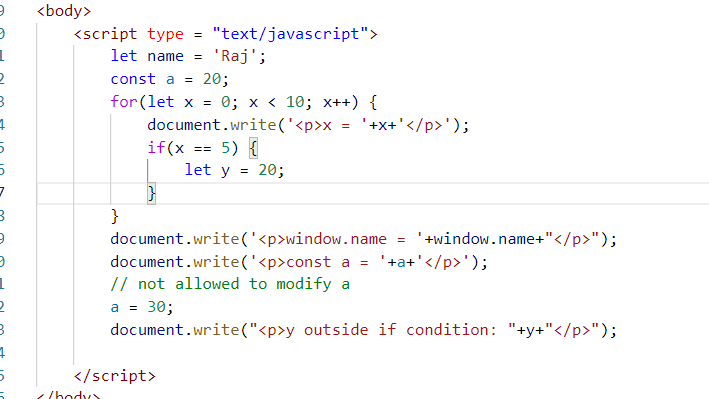


To address this problem ECMA script created introduced keywords to create block scoped variables i.e., let & const

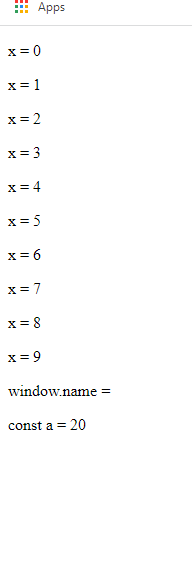
let: It is to create block scoped variables & it can be modified

const: It is to create block scoped variables & it is constant, can’t be modified

Note: let & const doesn’t add their variables to the global object, window



Output:



Template String Literals:

It is used to avoid string concatenations with many + character, so that you can create the strings with dynamic data without concatenation.

You can use `string ${expression} string`: Here the back tick can be used to create strings & the string can be added as it is however the dynamic data is wrapped in a ${data}

id = 100

qty = 4

url = “http://123.15.11.0/product/”+id+”/qty/”+qty

The above url is having many + character to concatenate string with dynamic value and also you can’t break the string inbetween without + operator

url = “http://123.15.11.0/

product/”+id+”/qty/”+qty

The above line is an error as string is not closed, it must be closed before bringing the same line to next line as below:

url = “http://123.15.11.0/” +

“product/”+id+”/qty/”+qty

To avoid this we can use a new syntax template string literals which doesn’t need any + to break the string

back tick (`) and single quote(‘): Theirs is a slight variation in back tick & single quote

url = `http://123.15.11.0/product/${id}/qty/${qty}`

Since id = 100 & qty is 4 the above URL will be

<http://123.15.11.0/product/110/qty/4>

The above URL can have a line break without any concatenation

url = `http://123.15.11.0/

product/${id}/qty/${qty}`

The above URL is still valid as back tick preserves the line break & doesn’t need any + to concatenate

Since id = 100 & qty is 4 the above URL will be

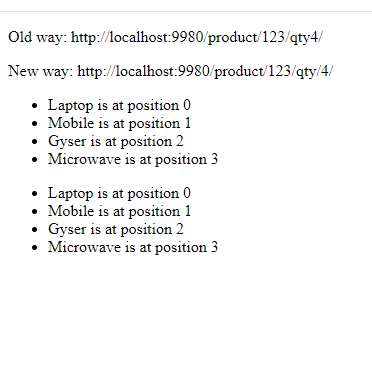
<http://123.15.11.0/product/110/qty/4>

These template strings are very much helpful in creating html elements when they have many dynamic data to be embedded.



You can observe the template strings reduces lot of concatenation.

Output:

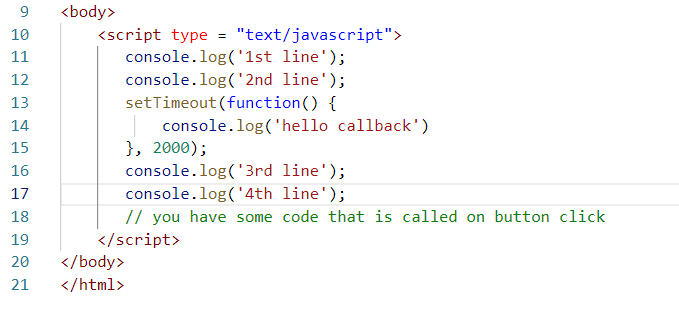


Callback:

These are special functions in JavaScript which are executed after some time but not immediately , they are usually asynchronous in nature, i.e., the next line or subsequent lines doesn’t wait for callback response

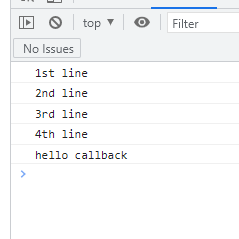
stmt1;  
stmt2;  
callbackfunction  
stmt3;  
stmt4

Here the stmt3, stmt4 is not going to wait for callback to execute, callback can be executed when it gets the opportunity



Here the setTimeout() takes callback as an argument, but the callback doesn’t block other operations for users, hence the 3rd & 4th line are executed without waiting for callback to complete, here the Front-end will not hang waiting for callback to complete, it still let user to access the UI’s.

Output:



In JavaScript you have XMLHttpRequest object which performs asynchronous operations by connecting to the backend.

XMLHttpRequest object provides some functions & properties to interact with the server side programs

open(http-method, url, boolean): Initialize the request

send(): it is to send the request

readyState: It will have state numbers of the XMLHttpRequest object from 1 to 4

1: request is initialized

2: request is sent

3: request is under process

4: response is ready to use

onreadystatechange: It is property that generates the event each time the XMLHttRequest object changes its state.

responseText: Stores the response from the backend

We have a fake online REST api to use XMLHttpRequest object to update our front end, we need to use callback here to update as we don’t have idea when the response will be ready.

<https://jsonplaceholder.typicode.com/users>

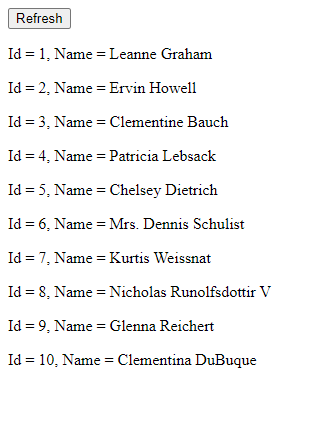


Here the callbacks are used to ensure that it will be executed when it gets time

i.e., the callback at 18th line is executed when the XMLHttpRequest objects changes its state from 1 to 4,

the another callback at 20th line is executed after 2seconds, by that time response will be ready as its used inside readyState == 4.

Output:



Note: Callbacks play an important role at the front end as they don’t block users to wait for the response.

These callbacks doesn’t have name which are called as anonymous functions, you use anonymous functions when you feel their logics are not reusable at other places.

Arrow Function:

This is a new feature in Javascript to simply writing anonymous function, it reduces writing function signature & also return statements if the anonymous function has only one line.

Usually callbacks are passed as arguments, but you will write function keyword, braces and return statement if necessary, but with arrow functions you can avoid them latelytely

Callback/anonymous function with one line having return statement:

function() {   
 return value;  
}

Arrow Functions with one line having return statement:

() => value;

Callback/anonymous function with more than one line having return statement

function() {   
 statement1;  
 return value;  
}

Arrow function with more than one line having return statement

() => {  
 statement1;  
 return value;  
}

Callback/Anonymous function of one line logic that doesn’t have return statement

function() {  
 statement1;  
}

Arrow function having one line logic that doesn’t have return statement

() => statement1;

Callback/Anonymous function of more than one line logic that doesn’t have return statement

function() {  
 statement1  
 statement2;  
}

Arrow function having more than one line logic that doesn’t have return statement

() => {  
statement1;  
statement2;  
}

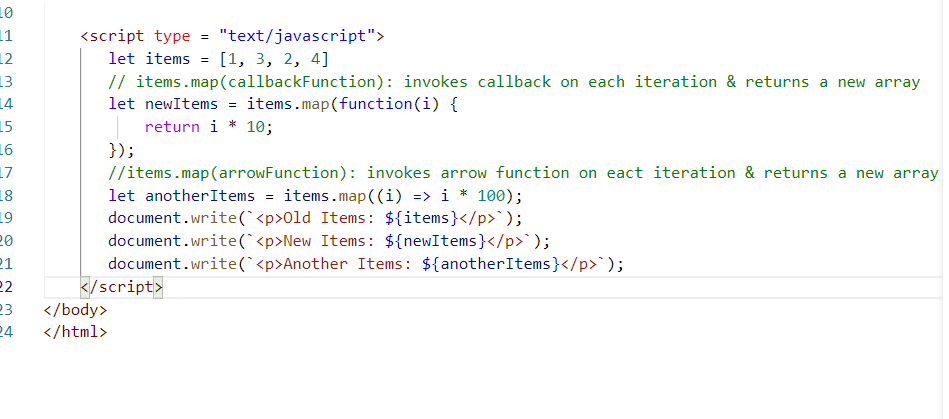
Callbacks/Anonymous function that takes argument

function(x, y) {   
  
}

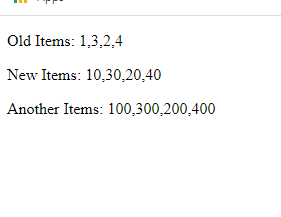
Arrow function that takes argument

(x, y) => { … }

Suppose you want to iterate an array and generate a new array in Javascript you have a map function



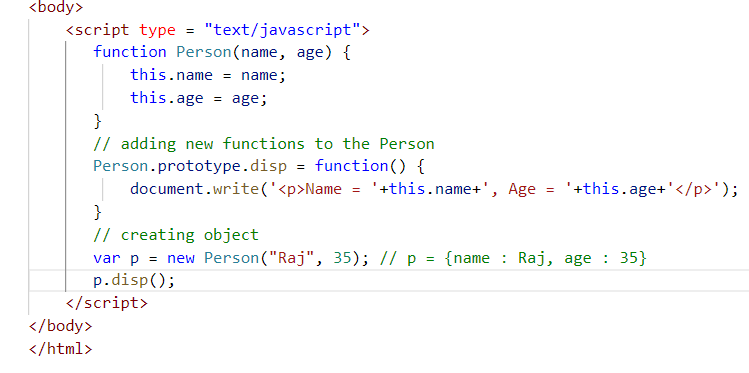
Output:



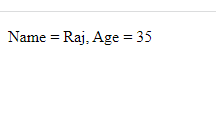
class, constructor, extends, super keyword

Earlier Javascript Functional based approach to create objects & achieve inheritance, which had a quite complex syntax:

Old approach



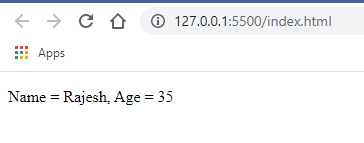
Output:



New feature of Javascript uses class & constructor to create objects & initialize objects, so that all the functions & constructors are modularized in a single unit.

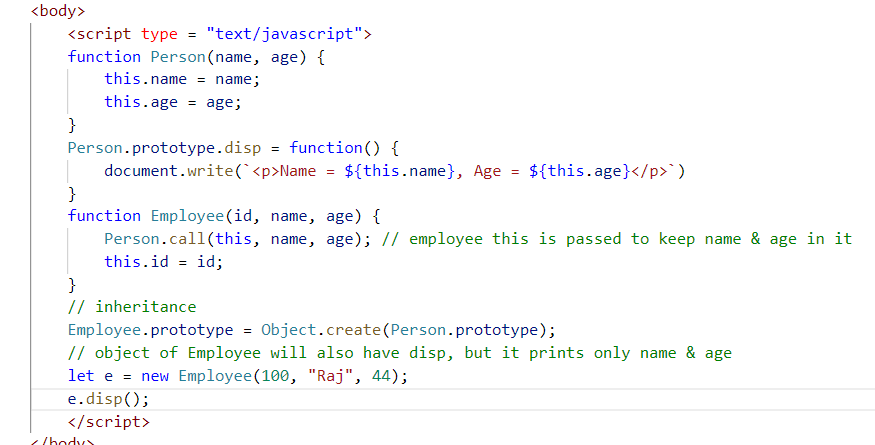


Output:

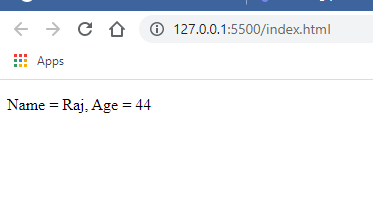


Another new feature is using extends & super to achieve inheritance

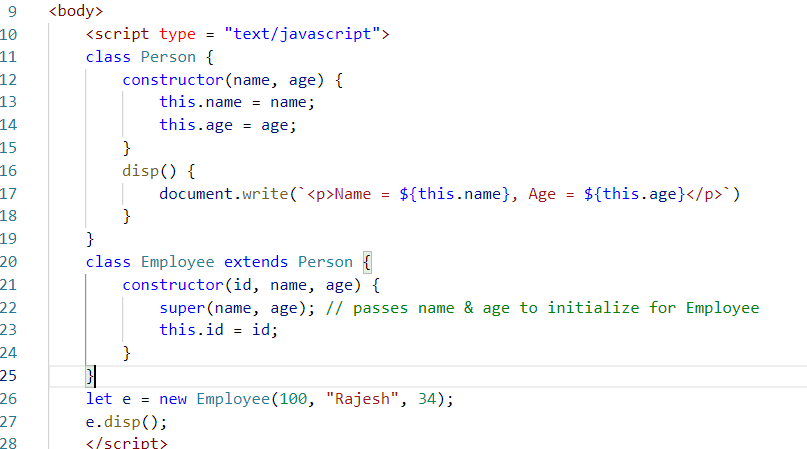
Older approach uses prototype to achieve inheritance and call() function to reuse the parent object function.



Here the properties of Employee is initialized by using Person.call() & Object.create() helps to inherit Person to Employee, hence you can access e.disp();



New syntax uses class, extends & super keyword to simplify achieving inheritance



Output:

