https://shorturl.at/3ICHe 7 Jun 2024

---

- Bootstrap
  - Pagination
  - Popover
  - Accordion
  - Toast
- Javascript
  - Intro
  - Variables
  - var vs let
  - Operators
  - Control Statements
  - Functions
    - Named functions
    - Arrow functions
    - Constructor function
  - Passing Parameters to functions
    - Rest parameters
    - Optional parameters
    - Default parameters
  - Prototype & Prototype chaining
  - JSON
  - JSON Server
  - Promises
  - Async Await

---

Pagination

- pagination
- page-item
- page-link
- pagination-sm md lg

```
<ul class="pagination">
            <ul class="pagination pagination-sm">
            <ul class="pagination pagination-md">
            <ul class="pagination pagination-lg">
            <ul class="pagination pagination-lg justify-content-center">
            <ul class="pagination pagination-lg justify-content-end">
        <ul class="pagination">
            <li class="page-item disabled">
                <a href="#" class="page-link">&laquo;</a>
            </li>
            <li class="page-item">
                <a href="#" class="page-link">1</a>
            </li>
            <li class="page-item">
                <a href="#" class="page-link">2</a>
```

```
            </li>
            <li class="page-item">
                <a href="#" class="page-link">3</a>
            </li>
            <li class="page-item">
                <a href="#" class="page-link">4</a>
            </li>
            <li class="page-item">
                <a href="#" class="page-link">5</a>
            </li>
            <li class="page-item">
                <a href="#" class="page-link">&raquo;</a>
            </li>
        </ul>
```

Collapse /Accordian

- collapse
- accordian
- multi-collpase

- data-toggle = "collapse"
- data-target = ""

```
<body>
    <div class="container mt-5">
        <!--
        <div class="accrodian" id = "parentDiv">
            <div class="card">
                <div class="card-header">
                    <div class="h2">
                        <button class="btn btn-link"
                                data-toggle = "collapse"
                                data-target = "#div1">Section 1</button>
                    </div>
                </div>
            </div>
            <div class="card">
                <div class="card-header">
                    <div class="h2">
                        <button class="btn btn-link"
                                data-toggle = "collapse"
                                data-target = "#div2">Section 2</button>
                    </div>
                </div>
            </div>
            <div class="card">
                <div class="card-header">
                    <div class="h2">
                        <button class="btn btn-link"
                                data-toggle = "collapse"
                                data-target = "#div3">Section 3</button>
                    </div>
                </div>
            </div>
            <div class="collpase" id = "div1" data-parent = "#parentDiv">
```

```
                <div class="card-body">
                    Section 1 <br>
                    Et sit voluptua sed dolore clita erat dolore, sit duo sit takimata clita, aliquyam eos amet lorem rebum sed sanctus amet gubergren stet. Dolore et dolore ea rebum et dolores diam. Invidunt duo et lorem invidunt, rebum ipsum consetetur diam consetetur kasd, est invidunt tempor diam sed eos rebum. Kasd sit vero est sadipscing. Kasd et voluptua consetetur sit duo. Eos ea takimata duo nonumy eirmod takimata elitr, nonumy justo at vero lorem vero. Tempor et tempor gubergren ipsum eirmod erat erat sit ea, amet dolor ipsum dolor et. Tempor accusam gubergren consetetur duo rebum gubergren amet nonumy, elitr sadipscing no ipsum dolores sit lorem ea dolor. Lorem consetetur duo magna no sed gubergren sadipscing ipsum. Labore erat vero amet kasd dolor et dolor. At eos erat gubergren invidunt nonumy eirmod, elitr dolore amet lorem invidunt diam lorem lorem rebum. No nonumy est sanctus takimata. Nonumy tempor at nonumy amet.
                </div>
            </div>

            <div class="collpase" id = "div2" data-parent = "#parentDiv">
                <div class="card-body">
                    Section 2 <br>
                    Et sit voluptua sed dolore clita erat dolore, sit duo sit takimata clita, aliquyam eos amet lorem rebum sed sanctus amet gubergren stet. Dolore et dolore ea rebum et dolores diam. Invidunt duo et lorem invidunt, rebum ipsum consetetur diam consetetur kasd, est invidunt tempor diam sed eos rebum. Kasd sit vero est sadipscing. Kasd et voluptua consetetur sit duo. Eos ea takimata duo nonumy eirmod takimata elitr, nonumy justo at vero lorem vero. Tempor et tempor gubergren ipsum eirmod erat erat sit ea, amet dolor ipsum dolor et. Tempor accusam gubergren consetetur duo rebum gubergren amet nonumy, elitr sadipscing no ipsum dolores sit lorem ea dolor. Lorem consetetur duo magna no sed gubergren sadipscing ipsum. Labore erat vero amet kasd dolor et dolor. At eos erat gubergren invidunt nonumy eirmod, elitr dolore amet lorem invidunt diam lorem lorem rebum. No nonumy est sanctus takimata. Nonumy tempor at nonumy amet.
                </div>
            </div>
            <div class="collpase" id = "div3" data-parent = "#parentDiv">
                <div class="card-body">
                    Section 3 <br>
                    Et sit voluptua sed dolore clita erat dolore, sit duo sit takimata clita, aliquyam eos amet lorem rebum sed sanctus amet gubergren stet. Dolore et dolore ea rebum et dolores diam. Invidunt duo et lorem invidunt, rebum ipsum consetetur diam consetetur kasd, est invidunt tempor diam sed eos rebum. Kasd sit vero est sadipscing. Kasd et voluptua consetetur sit duo. Eos ea takimata duo nonumy eirmod takimata elitr, nonumy justo at vero lorem vero. Tempor et tempor gubergren ipsum eirmod erat erat sit ea, amet dolor ipsum dolor et. Tempor accusam gubergren consetetur duo rebum gubergren amet nonumy, elitr sadipscing no ipsum dolores sit lorem ea dolor. Lorem consetetur duo magna no sed gubergren sadipscing ipsum. Labore erat vero amet kasd dolor et dolor. At eos erat gubergren invidunt nonumy eirmod, elitr dolore amet lorem invidunt diam lorem lorem rebum. No nonumy est sanctus takimata. Nonumy tempor at nonumy amet.
                </div>
            </div>
        </div>
    -->
```

```
        <p>
            <button class="btn btn-outline-secondary"
                    data-toggle = "collapse"
                    data-target = "#div1">Toggle First Element</button>
            <button class="btn btn-outline-secondary"
                    data-toggle = "collapse"
                    data-target = "#div2">Toggle Second Element</button>
            <button class="btn btn-secondary"
                        data-toggle = "collapse"
                        data-target=".multi-collapse">Toggle both
elements</button>
        </p>
        <div class="row">
            <div class="col">
                <div class="collapse multi-collapse" id = "div1">
                    <div class="card card-body">
                        <p>First Element </p>
                        <p>Justo magna dolores kasd consetetur sadipscing no
et amet et dolor, ipsum dolore lorem duo gubergren ea. Lorem clita est sed
sadipscing amet nonumy clita no lorem, magna aliquyam sit et erat lorem sed.
Sanctus sea dolores clita et dolore takimata amet, no sed kasd et magna, amet
labore dolor.</p>
                    </div>
                </div>
            </div>

            <div class="col">
                <div class="collapse multi-collapse" id = "div2">
                    <div class="card card-body">
                        <p>Second Element </p>
                        <p>Justo magna dolores kasd consetetur sadipscing no
et amet et dolor, ipsum dolore lorem duo gubergren ea. Lorem clita est sed
sadipscing amet nonumy clita no lorem, magna aliquyam sit et erat lorem sed.
Sanctus sea dolores clita et dolore takimata amet, no sed kasd et magna, amet
labore dolor.</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
```

Popovers
data-toggle = "popover"
title = ""
data-content = ""
data-placement = top  right/left/bottom
data-trigger = "focus"  -> dimissed when clicked outside

```
<body>
    <div class="container mt-5">
        <button class="btn btn-dark"
        data-toggle = "popover"
        data-title = "test-title"
        data-content = "Lorem et sed sadipscing nonumy sadipscing dolor clita
eos amet amet. Accusam labore ea sanctus gubergren. Takimata sit no labore sed
```

diam sit vero sit sea, at consetetur stet amet clita justo invidunt sed. Et amet ipsum nonumy dolor eirmod kasd. Erat stet amet sed et amet elitr sadipscing kasd. Amet no sed voluptua sanctus justo invidunt. Elitr consetetur sed amet elitr vero tempor eos sadipscing, gubergren magna et ea diam labore. Sed dolore ea accusam et. Gubergren est diam at duo amet ea stet accusam, diam amet est ea lorem est voluptua. Dolores dolores magna clita ut clita justo dolore sed rebum. Invidunt et ipsum et dolore rebum ea takimata sea. Eos dolor diam erat eirmod est clita, accusam takimata ut lorem et invidunt. Diam gubergren takimata amet diam no et sed, lorem ut kasd at no eos at et sadipscing et. Amet sit sanctus sanctus accusam diam duo sed invidunt gubergren, diam sadipscing elitr et gubergren, diam et lorem diam kasd. Rebum dolor eos erat tempor gubergren eos sanctus, eos nonumy sed clita consetetur rebum, amet sit eos sanctus lorem et dolore lorem sed et. Ipsum no est est ut. Ea amet sit diam takimata consetetur lorem. Lorem amet eirmod voluptua sed, sed dolor lorem eirmod dolore. Et sit sea sit labore ut ut, kasd sadipscing ipsum dolor sanctus consetetur clita dolore sed. Stet invidunt takimata et ut nonumy, consetetur eirmod duo dolor dolor voluptua sea at sea. Tempor eos aliquyam gubergren lorem voluptua stet diam diam, stet et elitr aliquyam aliquyam accusam aliquyam gubergren diam dolores. Diam et et diam est eirmod, eos magna sed et eos takimata stet, lorem lorem diam voluptua invidunt amet, invidunt kasd rebum elitr tempor, et no vero kasd labore accusam diam no ea dolor. Ipsum tempor labore consetetur clita takimata elitr nonumy lorem lorem. Sanctus ut dolores gubergren eirmod gubergren labore consetetur diam. Et duo sed lorem lorem rebum nonumy diam magna. Et takimata dolores tempor ea vero nonumy. Voluptua stet rebum dolor sed sit sit lorem lorem. Sed rebum diam est et diam takimata labore ipsum, aliquyam gubergren stet rebum sed dolor dolor sed kasd lorem, accusam sadipscing erat lorem et, accusam sed erat sanctus voluptua sanctus eirmod. Justo sit voluptua sadipscing dolor voluptua et sit elitr no, eirmod takimata rebum et gubergren consetetur erat at rebum. Invidunt lorem dolor kasd lorem, dolores sed et sanctus ipsum, nonumy et justo et et sadipscing sea lorem eos magna, clita takimata ipsum dolores justo kasd consetetur ipsum, eos accusam et sit amet ipsum sed, takimata et lorem sed dolor lorem. Magna dolores et ipsum ipsum vero no, stet vero consetetur et at magna gubergren sit eirmod clita. Tempor sadipscing et sit duo amet lorem eos dolor. No at sit nonumy et amet dolor. Labore magna ea diam sadipscing amet, lorem sit sit consetetur magna stet dolor labore duo consetetur. Ea consetetur sanctus dolores no at et dolor invidunt dolor. Et no amet ea kasd dolor sea sit ipsum, at amet sit et nonumy tempor stet nonumy eos duo, gubergren labore aliquyam sed magna rebum lorem ut ut invidunt, et ipsum ipsum consetetur sit stet, sit at est at ut dolor at lorem invidunt ut. Consetetur ipsum accusam sed amet consetetur sed et diam. Aliquyam amet dolore amet dolores tempor rebum consetetur dolor est. Sadipscing ea dolor dolor at, dolore dolores sea vero nonumy no sanctus ea. Kasd est sadipscing lorem erat dolor ea amet tempor. Stet duo amet vero stet elitr, at vero et sanctus rebum et, diam kasd dolores sed et dolor ipsum sit erat lorem. Ea ea kasd accusam et at rebum sed voluptua. Sanctus magna takimata diam takimata kasd et sed diam ut. Est amet sadipscing at voluptua et magna tempor sanctus, ipsum est nonumy amet lorem et. Sit eirmod no ea aliquyam, consetetur erat et duo sit at tempor tempor ut, sed elitr vero sadipscing tempor. Amet erat eirmod nonumy sanctus at dolore accusam et et. Lorem accusam dolor sanctus et eos sed ut dolore, at nonumy voluptua et lorem diam. Labore amet amet kasd lorem elitr amet. Sit aliquyam sanctus no dolor voluptua. Et sit tempor voluptua ipsum, vero sed duo diam invidunt, labore sit lorem lorem diam vero sit eos dolores rebum, ipsum erat dolor takimata consetetur vero stet, diam sed elitr sanctus diam sed ut, sanctus ea vero lorem erat diam labore ipsum amet, lorem at lorem et duo. Amet ut diam et consetetur, sit dolor at rebum ipsum nonumy et dolor. Dolores

```
sadipscing sit et sed magna consetetur, dolor magna no lorem amet amet sit sed
tempor sanctus. Duo labore stet kasd elitr nonumy takimata magna aliquyam
aliquyam, consetetur tempor consetetur no diam sed dolore sit kasd at. Magna
est sit no diam lorem aliquyam. Tempor amet nonumy invidunt kasd ut gubergren
labore justo, sed magna est accusam et. Ut rebum erat voluptua sea. Dolores
vero lorem et nonumy accusam stet, dolor dolore voluptua et est takimata vero.
Takimata et stet et ea, gubergren accusam ipsum aliquyam consetetur labore.
Amet dolore et vero accusam et sea amet voluptua, invidunt consetetur ea kasd
eirmod et. Lorem dolores dolores sit magna stet et elitr, clita amet voluptua
erat et tempor eos diam est sit, sit lorem ipsum sed duo lorem takimata, dolor
labore sit amet no nonumy est magna. Consetetur sanctus sit eirmod dolor diam
dolore lorem vero. Stet labore duo diam lorem labore, clita consetetur no
clita ipsum magna sit rebum nonumy. Et magna vero et nonumy clita labore. At
tempor sea eos aliquyam nonumy stet consetetur invidunt tempor. Dolore vero
dolor lorem ut ut labore diam eos sit, nonumy duo eos amet sed voluptua.
Labore voluptua duo sed sit et clita. Dolores diam takimata sanctus aliquyam
aliquyam magna sea amet eos. Sea gubergren et amet et sea tempor nonumy
voluptua lorem. Aliquyam dolore nonumy elitr nonumy vero et sed labore, eirmod
clita sanctus ipsum sit. Et ut est sed duo, clita dolor tempor amet sed elitr.
Amet rebum diam sea sit vero no stet."
        data-placement = "top"
        data-trigger = "focus"> Click Me</button>
    </div>
</body>
```

## Tosts

```
    <script>
        $(document).ready(() => {
            $(".del-toast").click(() => {
                //$("#my_msg").toast({delay : 5000})
                $("#my_msg").toast({autohide : false})
                $("#my_msg").toast('show')
            })
        })
    </script>

<body>
    <div class="container mt-5">
        <button class="btn btn-outline-danger del-toast">Delete</button>
        <div class="toast m-auto" id="my_msg">
            <div class="toast-header">Record deleted <button class="close mb-2
ml-auto"
                    data-dismiss="toast">&times</button></div>
            <strong>Voluptua dolor takimata.</strong>
            <small>5min left</small>
            <p>Ipsum ipsum sadipscing stet erat sadipscing. Ea duo gubergren
dolor consetetur erat ipsum, lorem at kasd
                sea ea dolor vero justo eirmod at, nonumy amet nonumy diam et
eirmod rebum invidunt ut takimata, tempor
                lorem labore aliquyam sit accusam sadipscing invidunt et. Sit
takimata nonumy magna sadipscing, sed
                magna eirmod.</p>
        </div>

    </div>
```

```
</body>
```

Bootstrap 5 new fetures
- jQuery replaced with Vanilla JS
- Improved Colour pallet
- Card deck removed
- Improved Grid system (xxl)
- Custom SVG library
- dropped support for IE 10 and IE 11

==========
Javascript intro
==========
- Javascript is a scripting language.
- Javascript was implemented by Brendan in 1995, at Netscape.
- It was initially named Livescript.
- Latest version is ES14 June 2023
    ES -> ECMA Script
    ECMA -> European Computer Manufacturers Association
- Use of Javascript
    - Scripting
    - Making Page as dynamic.
    - Client side validation.
    - Modern UI framework applications
        - AngularJS
        - ReactJS
        - VeuJS
        - NodeJS
        - ...
        - etc
- Javascript programs(scripts) can be executed by either web browsers or Node Engines.
- Javascript is an Object Based Scripting language.
- Javascript supports predefined objects
    Eg Object
    window,
    document,
    console, etc
- Javascript supports Primitives and nonprimitives
    primitives
        - string
        - number
        - boolean
        - undefined
        - null
        - symbol ???

- bigInt

non primitives

- object
- list
- arrays
- class ???
- Set
- Map
- JSON

==========
Environmental Setup
==========

- VS Code
  Website :- https://code.visualstudio.com/#alt-downloads
- NodeJS
  Website :- https://nodejs.org/en/download/

Note:- Javascript files must have extension '.js'

==========
Execution of Javascript scripts
==========

Create a file
***demo.js***

```
console.log("Hello World")
```

1. Command prompt
   >node demo.js
   Note:- 'document' object not supported here

2. HTML
***index.html***

```
...
    <script src="./demo.js"></script>
...
```

- open this file in web browser
- inspect console

3. HTML with script

```
...
    <script>
  console.log('I am console message from script tag')
  document.write('I am document message from script tag')
</script>
...
```

- open this file in web browser
- inspect console

==========
Variables:-
==========
- Variables are used to store data.
- by using variables we can store any kind of data,
  Eg number, sting, date, audio, video, file, etc.
- Variables can be declared using 'var', 'let', 'const'
- let and const introduced in ES5.

Strings:-
- Strings can be declared with ' ', " ", ``
- `` introduced in ES6.
- `` is used to define multiline strings

```
var wish = 'Good Morning'
console.log(wish)
var sub = "Javascript"
console.log(sub)
var myWish = `Welcome to ${sub}`
console.log(myWish)
```

Numbers
- Javascript supports decimal, double, bigInt, hexadecimal, octal and binary

```
let decimal = 100
console.log(decimal)
console.log(typeof(decimal))
let double = 10.23
console.log(double)
console.log(typeof(double))
let hexadecimal = 0x12ab
console.log(hexadecimal)
let octal = 0O127
console.log(octal)
let bin = 0b11
console.log(bin)
//Output will be in base 10 form
//all alphabates are case insensitive
let bigInt = 10n
console.log(bigInt)
console.log(typeof(bigInt))
//10n + 3 ?
```

Boolean
var flag = true
console.log(flag)
var flag = false
console.log(flag)

```
//var v/s let
```

```
//Eg01
for (var i = 0; i < 10; i++) { }
console.log(i)
for (let j = 0; j < 10; j++) { }
console.log(j)      //ReferenceError: j is not defined

//Eg02
var data = 100
var data = 200
console.log(data)

let num = 100
let num = 200
console.log(num)    //SyntaxError: Identifier 'num' has already been declared

//Eg03
var data = 100
{
    var data = 200
}
console.log(data)
let num = 100
{
    let num = 200
}
console.log(num)
//global polluting issue:- local variables affecting global variables.

//Eg04
console.log(data)
var data = 100
console.log(num)    //ReferenceError: Cannot access 'num' before
initialization
let num = 200

//Eg05 const
const data = {
    value: 100
}
console.log(data.value)
data.value = 200
console.log(data.value)
//data = {}   //TypeError: Assignment to constant variable.
```

==========

Control statements:-

==========

- Changing flow of program

Types

i) Decision control
ii)Looping control
iii)Case control
iv)Jump control

==========
Operators
==========
i) Arithmetic Operators -> eg +, -, *, /, %, ++, --, **
ii)Relational Operators -> eg ==, !=, >, <, >=, <=,
iii)Logical Operators -> eg &&, ||, !
iv)Bitwise Operators -> eg &, |, ^, ~, <<, >>
v) Assignment Operators -> eg =, +=, -=, *=, /=,...,<<=,>>=,...
Strictly equal to -> ===

5<<2
7<<2
120>>2

5<<2 0101
7<<2 0111
120>>2 1111000

5<<2 0101 10100
7<<2 0111 11100
120>>2 1111000 11110

5<<2 0101 10100 20
7<<2 0111 11100 28
120>>2 1111000 11110 30

C = 5
C <<= 2 C = C << 2 20

==========
Function:-
==========
- a particular logic is called as a function.
- functions are used to reuse the logic.
- There are following type of functions
    i) Named functions
    ii)Anonymous functions
    iii)Constructor functions

- we can pass various type of parameters to functions
    i) Rest parameters
    ii)Optional parameters
    iii)Default parameters

i) Named functions
- the function with name is called as named functions

- Syntax

```
//defining a function
function function_name(parameters)
{
        //business logic
}
//calling a function
function_name(parameters)
```

```
Eg01
    Create a function
        @fun_one
    fun_one return "Hello...!"
```

```
function fun_one(){
    return 'Hello...!'
}
console.log(fun_one)    //function definition
console.log(fun_one())  //function o/p
```

```
Eg02
    create a function
        @fun_one
    fun_one have three arguments
        @arg1
        @arg2
        @arg3
    fun_one return above arguments
```

```
function fun_one(arg1, arg2, arg3) {
    return arg1 + " <--> " + arg2 + " <--> " + arg3
    //return [arg1, arg2, arg3]
}
console.log(fun_one(`Tea`, `Milk`, `Coffee`))
console.log(fun_one('abc', 123, true))
```

```
//Eg03
    fun_one return fun_two definition
    fun_two retrun 'Hello...!'
    get the above message by fun_one
```

```
function fun_one() {
    return fun_two
}
function fun_two() {
    return 'Hello...!'
}
console.log(fun_one)
console.log(fun_one())
console.log(fun_one()())
```

ii)Anonymous functions

- the function without name is called as anonymous functions
- these are also called as arrow functions or callback functions

- we can represent arrow functions by '=>' symbol.
- arrow functions introduced in ES6.
- arrow functions are more secure than name functions.
- arrow functions utilise heap memory effectively.
- Syntax

```
//defining anonymous function
let variable_name = function(arguments){
        //business logic
}

//defining arrow function
let variable_name = (arguments) => {
        //business logic
}

//calling a function
variable_name(arguments)
```

```
Eg01
    create arrow function using following variables
        @fun_one
    fun_one return "I am from arrow function...!"

let fun_one = ()=>{
    return "I am from arrow function...!"
}
console.log(fun_one())

//Eg02
    Create arrow function
        @fun_one
    fun_one have three arguments
        @arg1,
        @arg2,
        @arg3
    fun_one prints above arguments

let fun_one = (arg1, arg2, arg3) => {
    console.log(arg1 + " <...> " + arg2 + " <...> " + arg3)
}
fun_one(10, 20, 30)
fun_one('Tea','Milk','Coffee')

Eg03
    one arrow function return another arrow function
    inner arrow function return 'Have a nice day...!'

let fun_one = () => {
    return () => {
        return 'Have a nice day'
    }
}
console.log(fun_one)
console.log(fun_one())
console.log(fun_one()())
```

==========
Constructor Functions:-
==========
- Constructor function is used to create a class like structure.
- In constructor function everything(variables and functions) start with 'this' keyword.
- We can create the object to the constructor using the 'new' keyword.

```
//Eg01
function class_one(){
    this.wish = `Welcome to Constructor functions`
    this.myWish = () =>{
        return `Good Afternoon`
    }
}
let obj = new class_one()
console.log(obj.wish)
console.log(obj.myWish())

//Eg02 Parameterised constructor
function class_one(arg1, arg2, arg3) {
    this.sub_one = arg1
    this.sub_two = arg2
    this.sub_three = arg3
}
let obj = new class_one(1, `2`, 3)
console.log(obj.sub_one, obj.sub_two, obj.sub_three)
```

==========
prototype
==========
- prototype is a property which adds members dynamically to constructor functions

```
//Eg03
function class_one(){}
class_one.prototype.wish = `Welcome to Prototype`
class_one.prototype.getWish = () => {
    return 'Good Afternoon'
}
console.log(new class_one().wish)
console.log(new class_one().getWish())
```

==========
Prototype Chaining:-
==========
- Nesting one constructor prototype to another constructor is called as prototype chaining.
- Concept of inheritance is achieved with prototype chaining

```
//Eg04 Single Inheritance
```

```
function class_one() { }
class_one.prototype.fun_one = function () {
    return "I am from Function one"
}
function class_two() { }
class_two.prototype = Object.create(class_one.prototype)
class_two.prototype.fun_two = function () {
    return "I am from Function two"
}

let obj = new class_two()
console.log(obj.fun_one())
console.log(obj.fun_two())
```

Try Multilevel, Multiple .... ?
Function Overriding ?

==========
Passing parameters to functions
==========

1. Default Parameters
 - It allows named parameters to be initialised with default values if no value or undefined is passed.

```
function fun_one(arg1 = 10, arg2 = "Fullstack", arg3 = "Angular") {
    console.log(arg1, arg2, arg3)
}
fun_one()                                   //10 FullStack Angular
fun_one(25)                                 //25 Fullstack Angular
fun_one(undefined, "MERN")                  //10 MERN Angular
fun_one(undefined, 'undefined', undefined)  //10 undefined Angular
fun_one(null, null, null)                   //null null null
fun_one(null, undefined, "MongoDB")         //null Fullstack MongoDB
fun_one(undefined, undefined, undefined)    //10 Fullstack Angular
```

2. Optional Parameters
 - While calling a function no need to pass all arguments.
 - We can keep a few arguments as Optional.
 - Optional Parameters introduced in ES6.

```
function fun_one(arg1, arg2, arg3) {
    console.log(arg1, arg2, arg3)
}
fun_one()                                   //undefined undefined undefined
fun_one("Hello_1")                          //Hello_1 undefined undefined
fun_one(null, "Hello_2")                    //null Hello_2 undefined
fun_one(10, 15.7, "MEAN")                   //10 15.7 MEAN
fun_one(null, null, null)                   //null null null
```

3. Rest Parameters
 - It is an improved way to handle function parameters.
 - It allows us to represent indefinite number of arguments  as an array.

- Rest parameters represented by '...' ( ... is called as spred operator)

```
function fun_one(...arg) {
    console.log(arg)
}
fun_one("Angular")                          //[ 'Angular' ]
fun_one()                                   //[]
fun_one(`Angular`, `Fullstack`)             //[ 'Angular', 'Fullstack' ]
fun_one(undefined, null)                    //[ undefined, null ]



function fun_one(arg1, arg2 = "Hello_2", ...arg3) {
    console.log(arg1, arg2, arg3)
}
fun_one("Hello_1")                                      //Hello_1 Hello_2 []
fun_one("Hello_1", undefined, "Hello_3")                //Hello_1 Hello_2 [
'Hello_3' ]
fun_one("Hello_1", "Hello_2", "Hello_3", "Hello_4") //Hello_1 Hello_2 [
'Hello_3', 'Hello_4' ]
fun_one(undefined, undefined, undefined)                //undefined Hello_2 [
undefined ]
fun_one(undefined, undefined, [1, 2, 3], [4, 5, 6]) //undefined Hello_2 [ [ 1,
2, 3 ], [ 4, 5, 6 ] ]
fun_one(undefined, undefined, [1, 2, 3], 4, 5, 6)   //undefined Hello_2 [ [ 1,
2, 3 ], 4, 5, 6 ]
fun_one("Hello_1", [1, 2, 3], 4, 5, 6)              //Hello_1 [ 1, 2, 3 ] [ 4,
5, 6 ]



//function fun_one(...arg1, ...arg2){}  //SyntaxError: Rest parameter must be
last formal parameter
//function fun_one(...arg1, arg2){}     //SyntaxError: Rest parameter must be
last formal parameter
```

==========

JSON

==========

- JSON stands for Java Script Object Notation
- JSON is used to share data over network.
- JSON is lightweight as compared to XML.
- Parsing of JSON is easy as compared to XML.
- Syntax
    - Object {}
    - Arrays []
    - Data key and value pairs
      keys and values are separated by ':'
      key value pairs are separated by ','

```
//Eg01
let demo = {
    'sub_one' : 'HTML',
    "sub_two" : "CSS",
    sub_three : `Bootstrap`
}
console.log(demo)
console.log(demo.sub_one, demo.sub_two, demo.sub_three)
//Iterate JSON
for(var x in demo)
    //console.log(x)
    console.log(demo[x])


//Eg02
let demo = {
    wish : 'Welcome',
    myWish : () =>{
        return `JSON Sesson`
    }
}
console.log(demo.wish,' to ',demo.myWish())



//freeze() seal()       //GO WITH HTML

let obj = {
    p_id : 111
}
console.log(obj)
//add new data
obj.p_name = "P_one"
console.log(obj)
//modify data
obj.p_id = 101
console.log(obj)
//Lock object using freeze function
Object.freeze(obj)
console.log("After Freeze",obj)
//try to modify data
obj.p_id = 100
console.log(obj)    //no error no modification
//try to add new data
obj.p_cost = 10000
console.log(obj)    //no error no modification
//try to delete data
delete obj.p_id
console.log(obj)    //no error no modification

//Lock Object by using seal function
Object.seal(obj)
console.log("After Seal",obj)
//try to modify data
```

```
obj.p_id = 100
console.log(obj)    //modification success
//try to add new data
obj.p_cost = 10000
console.log(obj)    //no error no modification
//try to delete data
delete obj.p_id
console.log(obj)    //no error no modification

freeze()
    -> Read Allowed
    -> Update   Denied      <=============
    -> Write    Denied
    -> Delete   Denied
seal()
    -> Read  Allowed
    -> Update   Allowed     <=============
    -> Write    Denied
    -> Delete   Denied

================================================
```

defineProperty:-
defineProperties:-

```
================================================
//defineProperty
let obj = {
    p_id: 111
}
console.log(obj)
Object.defineProperty(obj, "p_name",
    { value: "P_one", writable: false })
console.log(obj)
obj.p_id = 101
obj.p_name = "P_ONE"
console.log(obj)

//defineProperties
let obj = {
    p_id: 111
}
console.log(obj)
Object.defineProperties(obj, {
    "p_name": { value: "P_one",
        writable: true },
    "p_cost": { value: 10000,
        writable: false }
})
console.log(obj)
obj.p_id = 101
obj.p_name = 'P_ONE'
obj.p_cost = 111111
console.log(obj)



================================================
```

entries()  -> object to array
fromEntries()  -> valid array to object
==========

```
var obj1 = {
    p_id: 111,
    p_name: "p_one",
    p_cost: 10000
}
console.log(obj1)//{p_id: 111, p_name: 'p_one', p_cost: 10000}
let arr1 = Object.entries(obj1)
console.log(arr1)   //[["p_id", 111],["p_name", "p_one"],["p_cost", 10000]]
let arr2 = [["p_id", 111], ["p_name", "p_one"], ["p_cost", 10000,]]
console.log(arr2)   //[["p_id", 111],["p_name", "p_one"],["p_cost", 10000]]
let obj2 = Object.fromEntries(arr2)
console.log(obj2)   //{ p_id: 111, p_name: 'p_one', p_cost: 10000 }
```

==========
JSON Server
==========
- JSON server is used to create demo ReST JSON services
- JSON server can be used to develop
    GET
    POST
    PUT
    DELETE,...etc services.
- JSON server supports only JSON.
- JSON server is a lightweight server.
- JSON server installed using
    >npm install -g json-server@latest
- JSON server can be tested using the 'postman' tool.
    https://www.postman.com/downloads/

Deploying JSON server
    1. Create JSON file
***data.json***

```
{
    "products" : [
        {"id":1,"p_id":111,"p_name":"P_one","p_cost":10000},
        {"id":2,"p_id":222,"p_name":"P_two","p_cost":20000},
        {"id":3,"p_id":333,"p_name":"P_three","p_cost":30000},
        {"id":4,"p_id":444,"p_name":"P_four","p_cost":40000},
        {"id":5,"p_id":555,"p_name":"P_five","p_cost":50000}
    ]
}
```

Note:- while creating JSON unique id is compulsory, duplicate ids will be discarded (MUST BE STRING )

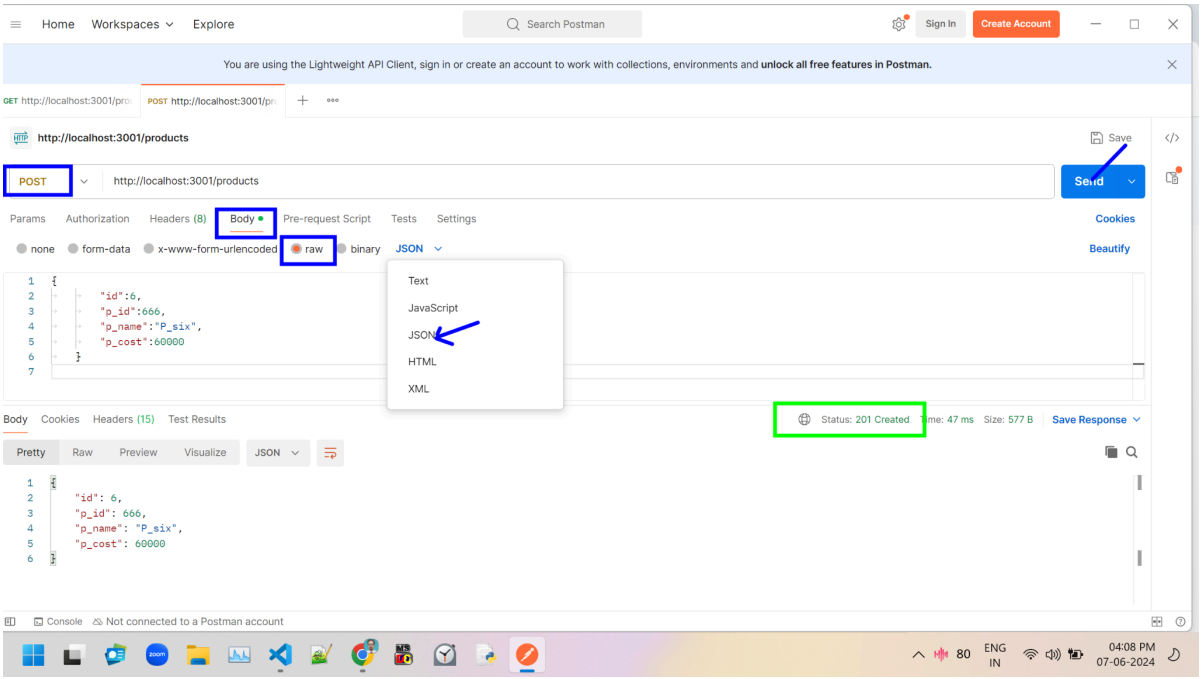
2. deploy data.json in JSON server
    >json-server --watch data.json
        OR
    >json-server -w data.json

by default JSON server running on port no 3000

3. changing port no (optional)
>json-server -w data.json -p 3001

test it with postman
GET
http://localhost:3001/products -> all records
http://localhost:3001/products/1 -> specific record
http://localhost:3001/products?q=444 -> specific records based on data

POST
http://localhost:3001/products
body -> raw-> text<->json
{
"id":6,
"p_id":666,
"p_name":"P_six",
"p_cost":60000
}




PUT
http://localhost:3001/products/5
body -> raw -> json
{
"id" : 5,
"p_id" : 555,

```
        "p_name" : "P_FIVE_UPDATED",
        "p_cost" : 55555
    }
```

on success status -> 200 OK
on failure status -> 404 Not found

DELETE
http://localhost:3001/products/6
on success status -> 200 OK
on failure status -> 404 Not found

==========
Promises
==========

- Promises are used to handle asynchronous operations in Javascript.
- Promises are special Javascript objects.
- Promises are having two states
  - Success (resolve)
  - Failure (reject)
- Promises can be created using the 'Promise' class.
- 'Promise' is the predefined class in Javascript.
- Promises can be consumed using 'then()'.

```
//Eg01
//create promise
let my_promise = new Promise((resolve, reject) => {
    resolve('Tommorow i will be at home')
})
//comsume promise
my_promise.then((posRes)=>{
    console.log(posRes)
},(errRes)=>{
    console.log(errRes)
})

//Eg02
let my_promise = new Promise((resolve, reject) => {
    setTimeout(() => {
        resolve('Success')
    }, 5000)
})
my_promise.then((posRes)=>{
    console.log(posRes)
},(errRes)=>{
    console.log(errRes)
})

//Eg03
let my_promise = new Promise((resolve, reject) => {
    reject('Failure')
    resolve('Successs')
```

```
})
my_promise.then((posRes)=>{
    console.log(posRes)
},(errRes)=>{
    console.log(errRes)
})

//Eg04
let my_promise = new Promise((resolve, reject) => {
    setTimeout(() => {
        resolve('Success')
    }, 5000)
    setTimeout(() => {
        reject('Failure')
    }, 4000)
})
my_promise.then((posRes)=>{
    console.log(posRes)
},(errRes)=>{
    console.log(errRes)
})
```

async and await
- These keywords are released in ES9.
- Above keywords are used to increase code readability.
- These keywords increase application performance.

```
//Eg05
let my_promise = new Promise((resolve, reject)=>{
    resolve('Hello')
})
async function my_fun(){
    let res = await my_promise
    console.log(res)
}
my_fun()

//Eg06
function add(num) {
    return new Promise((resolve, reject) => {
        resolve(num + 5)
    })
}
function sub(num) {
    return new Promise((resolve, reject) => {
        resolve(num - 3)
    })
}
async function my_fun() {
    let res1 = await add(5)
    let res2 = await sub(res1)
    console.log(res2)
}
my_fun()
```