https://shorturl.at/3ICHe 8 Jun 2024

Inputs/Outputs
Custom HTML Elements
Events
Callbacks
Api Calls (AJAX)
Storage Management
Datastructures
IIFE
Closure
Call Apply Bind functions
Generators Iterators
Array manipulators

==========
Inputs and outputs
==========
prompt
let var_one = document.getElementById(id).value
alert
document.getElementById(id).innerHTML = res
window.open(<filename.html>)

getElementById -> Find Others also

script.js

```
/*
let var_1 = parseInt(prompt('Enter First number'))
let var_2 = parseInt(prompt('Enter First number'))
let res = var_1 + var_2
//console.log(res)
alert(res)
*/
function add() {
    var_1 = parseInt(document.getElementById('var_1').value)
    var_2 = parseInt(document.getElementById('var_2').value)
    let res = var_1 + var_2
    document.getElementById('res').innerHTML = res
}
```

index.html*

```
<!DOCTYPE html>
<html>

<head></head>

<body>
    <input type="number" id='var_1' placeholder="Enter first  number"
style="margin: 20px;">
```

```
    <input type="number" id='var_2' placeholder="Enter Second number"
style="margin: 20px;">
    <button onclick="add()" style="margin: 20px;">Add</button>
    <label id = 'res'></label>
    <button>Logout</button>
    <script src="./script.js"></script>
</body>

</html>
```

==========

Creating inserting updating HTML elements

==========

Create HTML element

document.createElement('<html_element>')

eg

const myh1 = document.createElement('h1')

Insert Element in document

document.body.append('<html_element>')

eg

document.body.append(myh1)

Adding Contents

<html_element>.innerHTML = '<contents>'

Eg

myh1.innerHTML = 'Good Morning'

Setting Attributes

<html_element>.setAttribute('<attribute>','<value>')

Eg

myh1.setAttributes('style',color:red'')

script.js

```
//create html element
const myh1 = document.createElement('h1')
//insert in document
document.body.append(myh1)
//adding contents
myh1.innerHTML = 'Good Morning'
//setting attributes
myh1.setAttribute('style','color:red')
myh1.setAttribute('align','center')
```

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Custom HTML Elements</title>
    </head>
    <body>

        <script src="./script.js"></script>
    </body>
</html>
```

Task

index.html

```html
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <script src="./script.js"></script>
    </body>
</html>
```

script.js

```javascript
function myBus(dest,rout) {
    const mainDiv = document.createElement('div')
    const myttl = document.createElement('h3')
    const myrt = document.createElement('p')

    /*mainDiv.append(myttl)
    mainDiv.append(myrt)*/
    mainDiv.append(myttl, myrt)
    document.body.append(mainDiv)

    myttl.innerHTML = dest
    myrt.innerHTML = rout

    mainDiv.setAttribute('class', 'mainDiv')
    myttl.setAttribute('class', 'ttl')
    myrt.setAttribute('class', 'rout')
}
myBus('Mumbai',112)
myBus('Kolkata',113)
myBus('Delhi',114)
myBus('Nwyork',1012)
myBus('Romania',2012)
myBus('Vijaywada',132)
```

style.css

```css
.mainDiv{
    height: 200px;
    width: 400px;
    border: 5px double green;
    border-radius: 50px 50px 0px 0px;
    background: linear-gradient(to top, blue,red, yellow);
    display: inline-block;
    margin: 5px;
}
.ttl{
    float:left;
    width: 100%;
    text-align: center;
    font-family: cursive;
    background: linear-gradient(to bottom, rgba(0,0,0,0),orange);
    color: navajowhite;
    border-radius: 50px 50px 0px 0px;
```

```
}
.rout{
    width: 100%;
    text-align: center;
    margin-top: 120px;
    font-family:'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande',
'Lucida Sans', Arial, sans-serif;
    color:white;
}
```

==========
Events
==========
Keyboard events
Mouse Events

Keyboard events
- keydown
- keyup

Mouse events
addEventListener
- mouseenter
- mouseleave
- mousedown
- mouseup
- mousemove

Keyboard events

```
document.onkeydown = (e)=>{
    console.log("Key down",e.key)
}

document.onkeyup = (e) =>{
    console.log("Key up",e.key)
}
```

index.html

```
<!DOCTYPE html>
<html>
    <body>
        <h1 style="color:blue"> Press L to Logout</h1>
        <script src="./script.js"></script>
    </body>
</html>
```

script.js

```
/*
document.onkeydown = (e)=>{
    console.log("Key down",e.key)
}
```

```
document.onkeyup = (e) =>{
        console.log("Key up",e.key)
}
*/
document.onkeyup = (e) => {
    console.log("Key up ", e.key)
    if (e.key === 'l' || e.key === 'L')
        window.open('thanku.html')
    if (e.key === 'Escape')
        window.close()
}
```

thanku.html

```
<!DOCTYPE html>
<html>
<body>
    <h1 style="color:red">Thank you</h1>
    <h3 style="color:gray">Press 'Esc' to close</h3>
    <script src="./script.js"></script>
</body>
</html>
```

Mouse Events

index.html

```
<!DOCTYPE html>
<html>

<head>
    <style>
        .square {
            height: 200px;
            width: 200px;
            border: 2px double black;
            margin: 10px;
        }

        .ovl {
            height: 200px;
            width: 200px;
            border: 2px double blue;
            border-radius: 50%;
            margin: 10px;
        }
    </style>
</head>

<body>
    <div class="square" id='sq'></div>
    <div class="ovl" id='ov'></div>
    <script src="./script.js"></script>
</body>

</html>
```

script.js

```
let ov = document.getElementById('ov')
let sq = document.getElementById('sq')
ov.addEventListener('mouseenter', () => {
    sq.style.backgroundColor = 'gray'
})
ov.addEventListener('mouseleave', () => {
    sq.style.backgroundColor = 'transparent'
})
ov.addEventListener('mousedown', (e) => {
    console.log('Mouse down \n Co-ordinates are x = ', e.clientX, ' y = ',
e.clientY)
    sq.style.backgroundColor = 'Yellow'
})
ov.addEventListener('mouseup', (e) => {
    console.log('Mouse up \n Co-ordinates are x = ', e.clientX, ' y = ',
e.clientY)
    sq.style.backgroundColor = 'transparent'
})
ov.addEventListener('mousemove', (e) => {
    console.log('Mouse move \n Co-ordinates are x = ', e.clientX, ' y = ',
e.clientY)
})
```

==========

Callbacks

==========

- passing one function to another function as argument is called as callback

```
//Eg01
function fun_one(arg) {
    console.log(arg())
}
fun_one(() => {
    return `Hello....!`
})

//Eg02
function fun_one(arg1, arg2, arg3) {
    console.log(arg1(), arg2(), arg3)
}

fun_one(
    () => {
        return 123
    },
    () => {
        return `Javascript`
    },
    () => {
        return `MERN`
    }
```

```
)

//Eg01
function fun_one(arg) {
    console.log(arg())
}
fun_one(() => {
    return `Hello....!`
})

//Eg02
function fun_one(arg1, arg2, arg3) {
    console.log(arg1(), arg2(), arg3)
}

fun_one(
    () => {
        return 123
    },
    () => {
        return `Javascript`
    },
    () => {
        return `MERN`
    }
)

//Eg03
function add(num, callback) {
    return callback((num + 5), false)
}
add(5, (addRes, err) => {
    if (!err)
        console.log(addRes)
})

//Eg04
function add(num, callback) {
    return callback((num + 5), false)
}
function sub(num, callback) {
    return callback((num - 3), false)
}
function mul(num, callback) {
    return callback((num * 2), false)
}
function div(num, callback) {
    return callback((num / 6), false)
}
add(7, (addRes, err) => {
    if (!err) {
        sub(addRes, (subRes, err) => {
            if (!err) {
                mul(subRes, (mulRes, err) => {
                    if (!err) {
                        div(mulRes, (divRes, err) => {
```

```
                        if (!err)
                            console.log(divRes)
                    })//Div
                }
            })//Mul
        }
    })  //Sub
}
})  //Add

//call back hell

//Eg05
function add(num) {
    return new Promise((resolve, reject) => {
        resolve(num + 5)
    })
}
function sub(num) {
    return new Promise((resolve, reject) => {
        resolve(num - 3)
    })
}
function mul(num) {
    return new Promise((resolve, reject) => {
        resolve(num * 2)
    })
}
function div(num) {
    return new Promise((resolve, reject) => {
        resolve(num / 6)
    })
}
async function myFun() {
    let addRes = await add(7)
    let subRes = await sub(addRes)
    let mulRes = await mul(subRes)
    let divRes = await div(mulRes)
    console.log(divRes)
}
myFun()
```

==========

API Calls

AJAX

==========

CDN

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

NOTE:- to work with api calls we need api url (backend)
Will go with JSON server, that deployed previously
>json-server -w data.json -p 3001

index.html

```
<!DOCTYPE html>
<html>

<head>
    <title>AJAX Api calls</title>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scrip
t>
</head>

<body>
    <form method="post" id='myform'>
        <input type="number" placeholder="ID" name='uid' id='uid'> <br><br>
        <input type="number" placeholder="P_ID" name="p_id" id='p_id'>
<br><br>
        <input type="text" placeholder="P_NAME" name="p_name" id='p_name'>
<br><br>
        <input type="number" placeholder="P_COST" name="p_cost" id='p_cost'>
<br><br>
        <input type="submit" id="send" value="Send">
     
        <input type="submit" id='update' value="Update">
     
        <input type="submit" id='delete' value="Delete">
     
        <br><br>
    </form>
    <button id="getData">Get</button>
    <div style="color:red" id='op'></div>
    <script src="./script.js"></script>
</body>

</html>
```

script.js

```
url = `http://localhost:3001/products`
function LOAD() {
    $.ajax({
        url: url,
        type: 'GET',
        success: (posRes) => {
            console.log(posRes)
            x = ''
            x = x + `
            <table border = 1px
                cellpadding = 10px
                cellspacing = 10px
                align = center>

                <thead>
                    <tr>
                        <th>id</th>
```

```
                    <th>p_id</th>
                    <th>p_name</th>
                    <th>p_cost</th>
                </tr>
            </thead>
            <tbody>
        `
        for (let i = 0; i < posRes.length; i++) {
            x = x + `
                <tr>
                    <td>${posRes[i].id}</td>
                    <td>${posRes[i].p_id}</td>
                    <td>${posRes[i].p_name}</td>
                    <td>${posRes[i].p_cost}</td>
                </tr>
            `
        }

        document.getElementById('op').innerHTML = x
    },
    error: (errRes) => {
        console.log(errRes)
    }
})
}
//LOAD()
$(document).ready(() => {
    $('#getData').click((event) => {
        event.preventDefault()
        LOAD()
    })
    $('#send').click((event) => {
        event.preventDefault()
        let data = JSON.stringify({
            "id": parseFloat(document.getElementById('uid').value),
            "p_id": parseInt(document.getElementById('p_id').value),
            "p_name": document.getElementById('p_name').value,
            "p_cost": parseInt(document.getElementById('p_cost').value)
        })
        $.ajax({
            url: url,
            type: 'POST',
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            data: data,
            success: (posRes) => {
                console.log(posRes)
                LOAD()
            },
            error: (errRes) => {
                console.log(errRes)
            }
        })
    })
    $('#update').click((event) => {
        event.preventDefault()
```

```
        let id = parseFloat(document.getElementById('uid').value)
        let data = JSON.stringify({
            "p_id": parseInt(document.getElementById('p_id').value),
            "p_name": document.getElementById('p_name').value,
            "p_cost": parseInt(document.getElementById('p_cost').value)
        })
        $.ajax({
            url: url + '/' + id,
            type: 'PUT',
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            data: data,
            success: (posRes) => {
                console.log(posRes)
                LOAD()
            },
            error: (errRes) => {
                console.log(errRes)
            }
        })
    })
    $('#delete').click((event) => {
        event.preventDefault()
        let id = parseFloat(document.getElementById('uid').value)
        $.ajax({
            url: url + '/' + id,
            type: 'DELETE',
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            success: (posRes) => {
                console.log(posRes)
                LOAD()
            },
            error: (errRes) => {
                console.log(errRes)
            }
        })
    })
})
```

==========

Storage Management

==========

Local Storage, Session Storage

- Local storage is persistent storage of the browser.
- Session storage is temporary storage of the browser, till that session only.
- We can store data in the form of key and value pairs.
- It supports only string data.(to store objects stringify them)
- Keys are unique
- All functions are common for local and session storage.
- setItem() function is used to store items.
- getItem() function is used to read items.
- removeItem() function is used to delete items.
- localStorage and sessionStorage belong to the 'window' object.

Inspect application -> left hand side there is Local Storage and Session Storage

Task
- Create data.json
    -> create users array
        username password
- Login page
    -> accept username and password from user
    -> check authentication from JSON server
    -> if authentication success
        store username in session storage
        open welcome page.
- Welcome page
    -> read session storage
    -> if there is any username wish that user welcome
    -> if there is nothing in session storage give message 'unauthorised user'

data.json

```
{
    "users": [
        {
            "uname": "Ramesh",
            "upwd": "123"
        },
        {
            "uname": "Suresh",
            "upwd": "abc"
        },
        {
            "uname": "Rakesh",
            "upwd": "pqr"
        }
    ]
}
```

index.html

```
<!DOCTYPE html>
<html>

<head>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></scrip
t>
</head>

<body>
    <h1>Login</h1>
    <form>
        <!--Username-->
        <input type="text" placeholder="Username" id="uname"> <br><br>
        <!--Password-->
```

```
        <input type="password" placeholder="Password" id="upwd"> <br><br>
        <!--Login Button-->
        <input type="submit" value="Login" id="login">
    </form>
    <script src="index.js"></script>
</body>

</html>
```

index.js

```
let url = "http://localhost:3000/users"

function login() {
    let uname = document.getElementById('uname').value
    let upwd = document.getElementById('upwd').value
    $.ajax({
        url: url + "?q=" + uname,
        type: 'GET',
        success: (posRes) => {
            console.log(posRes)
            if (posRes.length > 0 && posRes[0].upwd == upwd) {
                alert("Login Success")
                window.sessionStorage.setItem('user', uname)
                window.open("./welcome.html")
            }
            else
                alert('Login Failed')

        },
        error: (errRes) => {
            console.log(errRes)
        }
    })
}

$(document).ready(() => {
    $('#login').click((e) => {
        e.preventDefault()
        login()
    })
})
```

welcome.html

```
<!DOCTYPE html>
<html>

<head></head>

<body>
    <h1 id='wish'></h1>
    <script src="./welcome.js"></script>
</body>

</html>
```

welcome.js

```
let user = window.sessionStorage.getItem('user')
if (user)
    document.getElementById('wish').innerHTML = 'Welcome ' + user
else
    document.getElementById('wish').innerHTML = 'Unauthorised user'
```

==========

Datastructures in JS

==========

i) Map
ii)WeakMap
iii)Set
iv)Weakset

```
let obj = {}
//let key1 = {}
let key1 = {k1:'v1'}
let val1 = `Hello`
obj[key1] = val1
//console.log(obj)
//let key2 = {}
let key2 = {k2:'v2'}
let val2 = `Hi`
obj[key2] = val2
console.log(obj)
```

Problem with JSON
- if we are having key as object for more than one key value pairs, latest value overrides previous value.
- to overcome this issue Map() and WeakMap() are used.

Note:- to check the internal structure 'dir()' function is used.
let map = new Map()
console.dir(map)

i) Map()
- size
- Map()
- get()
- set()
- has()
- delete()
- clear()
- keys()
- values()

```
//Eg01
let map = new Map()
let key1 = {}
let key2 = {}
let val1 = `Hello_1`
```

```
let val2 = `Hello_2`
map.set(key1, val1)
map.set(key2, val2)
console.log(map)

//Eg02
let map = new Map()
map.set(`key1`, `Hello_1`)
    .set(`key2`, `Hello_2`)
    .set(`key3`, `Hello_3`)
    .set(`key4`, `Hello_4`)
    .set(`key5`, `Hello_5`)
console.log(map)
console.log(map.size)       //5
console.log(map.keys())//[Map Iterator] { 'key1', 'key2', 'key3', 'key4',
'key5' }
console.log(map.has(`Key5`))//false
console.log(map.get(`key5`))
map.delete(`key5`)
console.log(map)
for (let [k, v] of map)
    console.log(k, v)
console.log(map.values())   //[Map Iterator] { 'Hello_1', 'Hello_2',
'Hello_3', 'Hello_4' }
map.clear()
console.log(map)

//Eg03
let map = new Map()
map.set("Key1","Hello_1").set("Key2","Hello_1")
console.log(map)    //duplicate values allowed and accepted
map.set("key3","Hello_3").set("key3","Hello_4")
console.log(map)    //duplicate keys allowed older values replaced with new
one
```

==========

ii)WeakMap:-

==========

- it wont allow primitives as keys
- WeakMap()
- delete()
- get()
- set()
- has()

```
let wm = new WeakMap()
//console.dir(wm)
//Eg01
let key1 = {}
let value1 = "Hello_1"
wm.set(key1, value1)
//console.log(wm)
let key2 = {}
let value2 = "Hello_2"
wm.set(key2, value2)
```

```
console.log(wm)
//wm.set("Key3","Hello_3")    //TypeError: Invalid value used as weak map key
```

==========

iii)Set() :- duplicates are discarded

==========

- Set()
- has()
- add()
- delete()
- clear()
- values()
- keys()

```
//Eg01
let set = new Set()
set.add(10)
    .add(20)
    .add(10)
    .add(30)
    .add(20)
console.log(set)    //?


//Eg02
let arr = [10, 20, 30, 10, 20, 20, 10, 40]
let set = new Set(arr)
console.log(set)

//Eg03
let set = new Set()
set.add(10)
    .add(20)
    .add(30)
    .add(40)
    .add(50)
console.log(set)            //Set(5) {10, 20, 30, 40, 50}
console.log(set.values())   //SetIterator {10, 20, 30, 40, 50}
console.log(set.keys())     //SetIterator {10, 20, 30, 40, 50}
set.delete(50)
console.log(set)            //Set(4) {10, 20, 30, 40}
for (let x of set)
    console.log(x)
set.clear()
console.log(set)            //Set(0) {size: 0}
```

==========

iv)WeakSet():-

==========

- It wont allow primitives

```
let ws = new WeakSet()
//ws.add(10)    //TypeError: Invalid value used in weak set
let key1 = { data: 10 }
let key2 = { data: 20 }
```

```
ws.add(key1).add(key2)
console.log(ws)
ws.delete(key1)
console.log(ws)
console.log(ws.has(key1))
```

==========

IIFE()

==========

- Immediately invoked function expression.
- Introduced in ES9. ?
- These are self invokable functions, i.e. no need to call IIFEs.
- Syntax

(()=>{})()

```
//Eg01
(()=>{
    console.log("Welcome to IIFE")
})();

//Eg02
((arg1, arg2) => {
    console.log(arg1 + arg2)
})(10, 20);

//Eg03
let res = (() => {
    return `Good Evening`
})()
console.log(res);
```

==========

Closure:-

==========

- Inner function can have access to data from outer function.
- Outer function returns inner function.
- Closure means the inner function can have access to data from the outer function even after returning the inner function.

```
function addn(x) {
    return (y) => {
        return x + y
    }
}
//here outer function returned inner function
let var_a = addn(5)
let var_b = addn(10)
//here we called returned inner function with access of variable
//from outer function
console.log(var_a(2))
console.log(var_b(4))
```

==========

call apply and bind functions

==========
call():-
- This function is used to create relationships between two unknown memory locations.

apply():-
- It is same as that of call function
- When we have to pass arguments as an array this function is used. (array implies no independent arguments).

bind():-
- this function is used to merge two unknown memory locations.
- this function returns a new function

```
let obj = {
    num: 10
}
console.log(obj)
function myFun(arg) {
    return this.num + arg
}
console.log(myFun(10))
console.log(myFun.call(obj, 10))
function newFun(arg1, arg2, arg3) {
    return this.num + arg1 + arg2 + arg3
}
console.log(newFun.call(obj, 20, 30, 40))
let arr = [20, 30, 40]
console.log(newFun.apply(obj, arr))
let bindFun = newFun.bind(obj)
console.log(bindFun(1, 2, 3))
```

==========
Generators and Iterators
==========
Generators :-
- Generator produces values dynamically.
- Generators utilise memory effectively.
- Generators are represented by '*'
- Generators are functions.
- Generator returns a cursor.
- Cursors are objects.
- 'next()' function is used to access records.

```
//Eg01
function* fun_one() {
    yield 10
    yield 20
    yield 30
    yield 40
    yield 50
}
let cursor = fun_one()
```

```
console.log(cursor)
console.log(cursor.next())
console.log(cursor.next())
console.log(cursor.next())
console.log(cursor.next())
console.log(cursor.next())
console.log(cursor.next())
```

Iterators:-
- Iterators include for loops
    - for()
    - forEach()
    - for...of
    - for...in

```
let arr = [10, 20, 30, 40, 50]
console.log('Origin array:- ', arr)
for (let i = 0; i < arr.length; i++) {
    arr[i] *= 10
    console.log(arr[i])
}
console.log('Origin array:- ', arr)
arr.forEach((element, index) => {
    element /= 10
    console.log(index, element)
})
console.log('Origin array:- ', arr)
for (let value of arr) {
    value /= 5
    console.log(value)
}
console.log('Origin array:- ', arr)
let obj = {
    "p_id": 111,
    "p_name": "P_one",
    "p_cost": 10000
}
for (let key in obj)
    //console.log(key)
    console.log(obj[key])
```

==========
Classes in JavaScript
==========
introduced in ES6 (ECMA Script 2015)
Syntax

```
//class declaration
class <name_of_class>
{
        //class variables
        //class methods
}
//create object
```

```
        let obj = new name_of_class()
```

==========
Inheritance:-
==========
- Class inheritance is a way for one class to extend another class.
- So we can create new functionality on top of existing.
- this can be achieved with 'extends' keyword.
- Eg class_two extends class_one

Types

i) Single Inheritance

```
class class_one

class class_two extends class_one

Create object of class_two
```

ii)Multilevel Inheritance

```
class class_one

class class_two extends class_one

class class_three extends class_two

Create object of class_three
```

iii)Multiple Inheritance ?
iv)Hierarchical Inheritance ?
v) Hybrid Inheritance ?

==========
Interface(Typescript only):-
==========
- It contains only definitions and not intialyzation not implementation
- keyword 'implements'

Private access identifier (#)
#member

```
class Student{
    #roll_no = 5
    #div = 'A'
    displayInfo(){
        console.log("Roll number:- ",this.#roll_no)
        console.log("Division :- ",this.#div)
    }
}
new Student().displayInfo()
```

instanceof

- it is used to check whether the object is instance of particular class or not

```
class Student { }
let obj1 = 5
let obj2 = new Student()
console.log(obj1 instanceof Student)
console.log(obj2 instanceof Student)
```

Polymorphism
- i) Method Overloading ?
- ii)Method Overriding

==========
Array Manipulators
==========

```
/*
01. map():-
 - This function is used to manipulate each and every element in array
 - it returns an array

//Eg01
let arr = [10, 20, 30, 40, 50]
//multiply each element by 2
console.log(arr.map((element, index) => {
    return element * 2
}))

//Eg02
let arr = [1, 2, 3, 4, 5]
//o/p ['$1','$2','$3','$4','$5']
console.log(arr.map((element, index) => {
    return '$' + element
}))

//Eg03
let arr1 = [1, 2, 3]
let arr2 = ['one', 'two', 'three']
//o/p [ [ 1, 'one' ], [ 2, 'two' ], [ 3, 'three' ] ]
console.log(arr1.map((elemenet, index)=>{
    return [elemenet, arr2[index]]
}))

02. filter():-
 - this function creates an array based on condition

//Eg01
let arr1 = [10, 20, 30, 40, 50]
//create array with elements greater than 30
console.log(arr1.filter((elemenet, index) => {
    return elemenet > 30
}))
```

```
//Eg02
let arr2 = [10, 100, 20, 200, 30, 300, 40, 400, 50, 500]
//create an array elements greater than or equal to 100
console.log(arr2.filter((element, index) => {
    return element >= 100
}))

//Eg03
let arr3 = [10, 20, 30, 40, 50]
//o/p [300, 400, 500]
let res1 = arr3.filter((element, index)=>{
    return element >= 30
})
console.log(res1)
let res2 = res1.map((element, index)=>{
    return element * 10
})
console.log(res2)
///////////////
let res1 = arr3.filter((element, index)=>{
    return element >= 30
}).map((element, index)=>{
    return element * 10
})
console.log(res1)
/////////////////
console.log(arr3.filter((element, index)=>{
    return element >= 30
}).map((element, index)=>{
    return element * 10
}))


03. reduce()            left to right   0 -> 1
04. reduceRight()       right to left   0 <- 1

let arr = [1, 2, `3`, 4, 5]
console.log(arr.reduce((pv, nv) => {
    return pv + nv
}))

console.log(arr.reduceRight((pv, nv) => {
    return pv + nv
}))

05. forEach
06. for...of
07. for...in

08. push():- add element at end
09. unshift():- add element at beginning
10. pop():- remove element from end
11. shift():- remove element from beginning
*/
let arr = [20, 30, 40]
console.log(arr)            //[20, 30, 40]
```

```
arr.push(50)
console.log(arr)            //[ 20, 30, 40, 50 ]
arr.unshift(10)
console.log(arr)            //[ 10, 20, 30, 40, 50 ]
console.log(arr.pop())
console.log(arr)            //[ 10, 20, 30, 40 ]
console.log(arr.shift())    //10
console.log(arr)            //[ 20, 30, 40 ]
```