Wednesday, February 9, 2022 9:43 AM

http://es6-features.org/

Ecma Specification version 6 ------ ES6

Javascript language is based on this specification!!!

ES6 -----

Spread Operator --- looks like rest parameter === **...varname** (three dots followed by a variable name)

Rest parameter is used for variable number of arguments to be passed to function.(internally the parameter is treated as an array)

Spread operator is used for COPYING

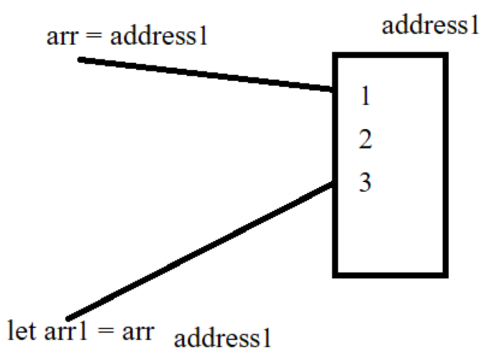
spread operators are used with arrays and objects !!!

____

```
        function  f1( ... arr)    // arr is a formal parameter and a rest parameter
        {
        }

          f1(12,13,14)   //12,13,14 are actual parameter
```

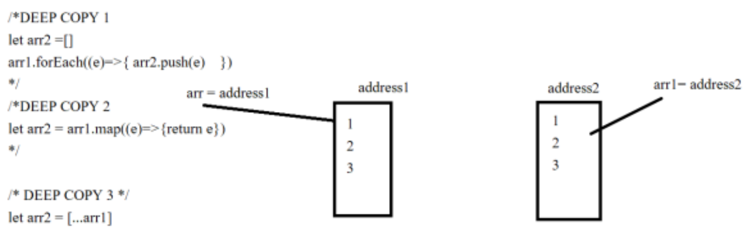
____

```
        console.log( ...arr )    //Actual parameter  is  a spread operator
```



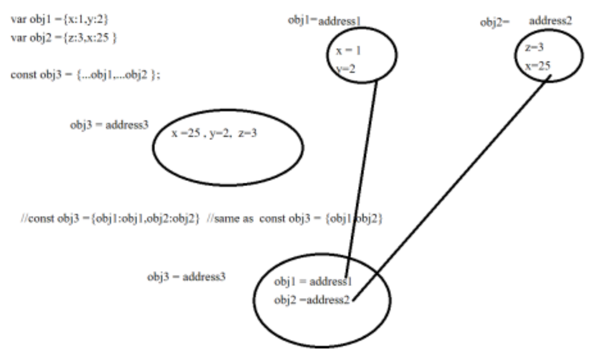
SHALLOW COPY !!!

FOR DEEP COPY



```
/*DEEP COPY 1
let arr2 =[]
arr1.forEach((e)=>{ arr2.push(e)   })
*/
/*DEEP COPY 2
let arr2 = arr1.map((e)=>{return e})
*/

/* DEEP COPY 3 */
let arr2 = [...arr1]
```

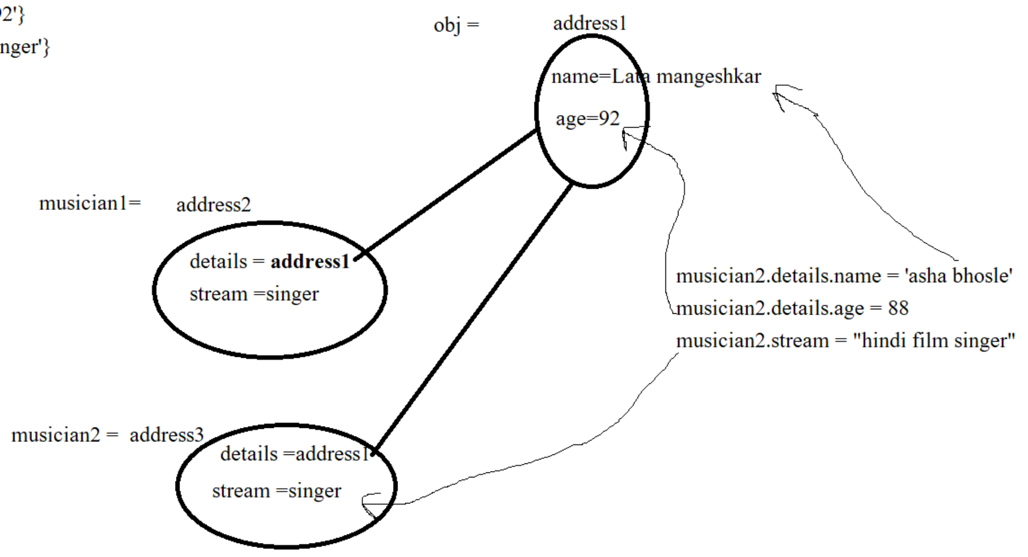
____



```
var obj1 ={x:1,y:2}
var obj2 ={z:3,x:25 }

const obj3 = {...obj1,...obj2 };

//const obj3 ={obj1:obj1,obj2:obj2}  //same as  const obj3 = {obj1,obj2}
```

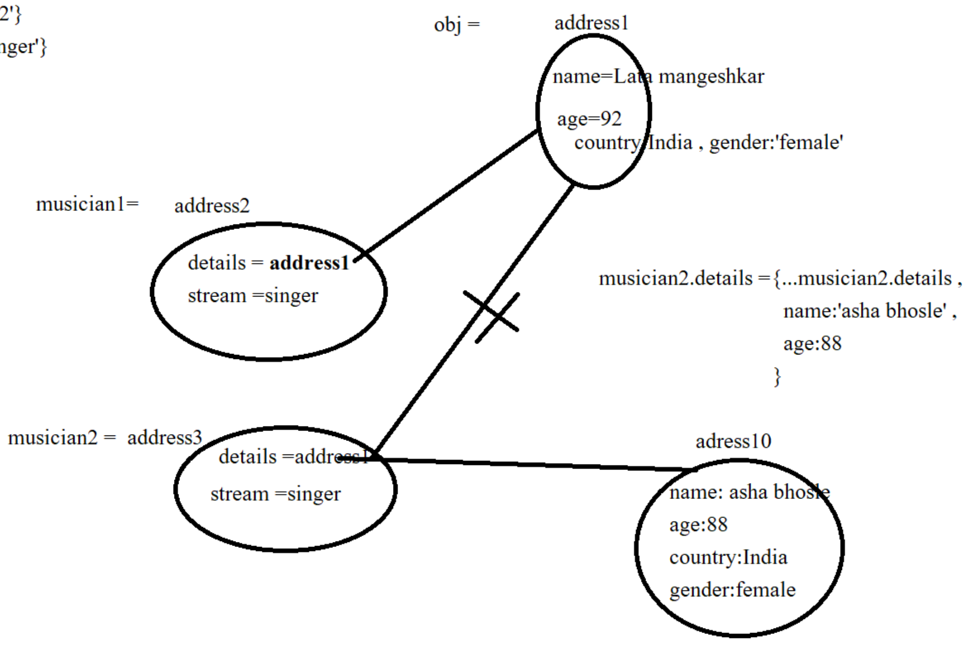
USING SPREAD OPERATOR DEEP COPY OF CONTAINED OBJECT /HAS A property DOES NOT HAPPEN

```
let obj = {name:'lata mangeshar' , age:'92'}
let musician1 ={ details : obj, stream:'singer'}

let musician2={...musician1}
```



```
musician2.details.name = 'asha bhosle'
musician2.details.age = 88
musician2.stream = "hindi film singer"
```

```
let obj = {name:'lata mangeshar' , age:'92'}
let musician1 ={ details : obj, stream:'singer'}

let musician2={...musician1}
```



```
musician2.details ={...musician2.details ,
                   name:'asha bhosle' ,
                   age:88
                  }
```

---

async and await KEYWORDS in backend java-script ====NODE

async defines **function that returns a promise by default**

await = block the code flow **(wait until)** till the promise is resolved or rejected !!!

3 data structures for holding the function calls in javascript

Stack = normal function calls are pushed and popped when returned
Callback Queue = the callback functions that occur while executing code is added to this queue
Job Queue = the promises that occur while executing the code are added to job queue

Job queue has higher priority than callback queue
Stack has highest priority

____

```
new Promise( (resolve,reject )= > {
        ….
        ….
        ….
        Depending on results the function either resolves or rejects
})
```

____

If a named , anonymous , arrow function is preceded by async in its definition ----IT ALWAYS returns a promise

Await keyword waits for a promise to resolve !!!! --- blocking code
Await is always written inside the async function

____

Express Server = BACK END WEB SERVER for EXPOSING REST API written in NODE.js

Tomcat Server = BACK END WEBSERVER FOR EXPOSING REST API written in JAVA

Check the node module folder that you installed on day one i.e in the set path of your env variables !!!! Check for express if not go to that install folder (outside node_modules) and then type the following
npm install express --save

____