<hr>

<br>leaves line

<ul><ol><li>

<p>

An image that is a link:

<a href="https://www.w3schools.com">

<img src="smiley.gif" alt="Go to W3Schools!" width="78" height="78" border="78">

</a>

flexbox

CCS TRICKS

UNSPLASH

PALETTON.COM

google fonts

codepen.io(css)

</p>

JAVASCRIPT

alert gives the popup

<script type="text/javascript" src="name of file">

a)functions

i)function function\_name()

{

-----}

function\_name();//calling

ii)var sayBye=function()

{

console.log("bye");

}

sayBye();

b)array

var array\_name=[];

array can also store functions ,array inside array

array\_name.shift():deletes 1st element

array\_name.pop():deletes last element

array\_name.push(): adds element at the last

array\_name.concat():adds two array creating a new array addition to original one

array\_name.sort():

c)objects

var ob\_name={

name:"",

age:,

};

ob\_name.name

to add new thing--ob\_name.property="";

DOM SELECTORS

i) var x=document.querySelector("li")

ii) document.querySelectorAll("li")(it is a form of array)

document.createElement("li");

iii)getElementByTagName("button");

iv)getAttribute,setAttribute

v)document.querySelector("h1").style;//gives the styles of first heading

vi)document.querySelector("h1").style.background="yellow";//can change style of preexisting

vii)x.className="coolTitle"(style of classtitle defined in css)

viii)x.classList.add("coolTitle");

ix)x.classList.remove("coolTitle");

x)x.classList.toggle("coolTitle");(toggle:on off)

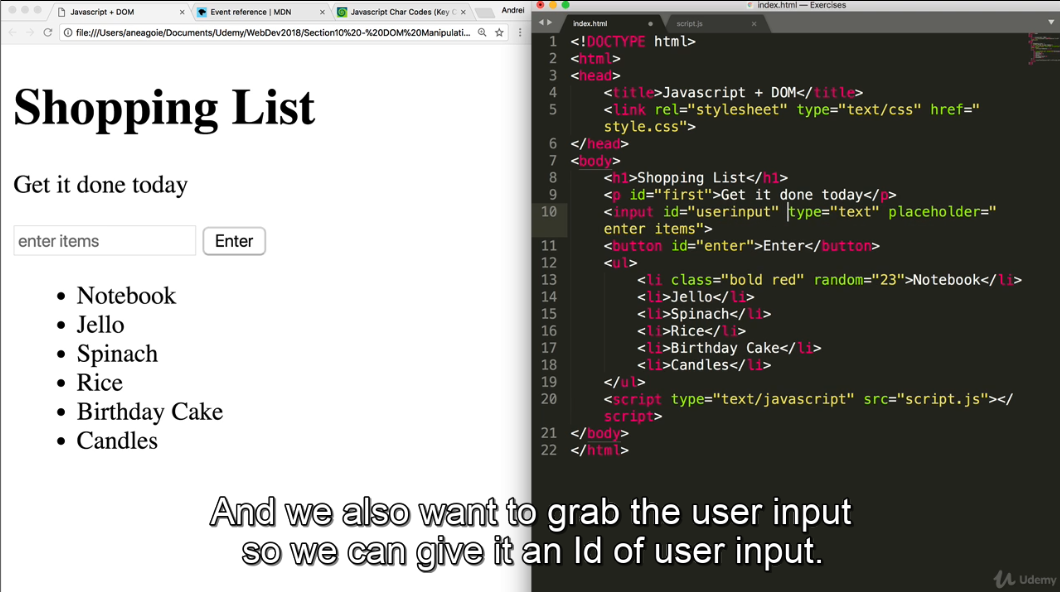
EVENT LISTNER

i)var button=document.getElementByTagName("button")[0] ;

botton.addEventdListner("click",function()

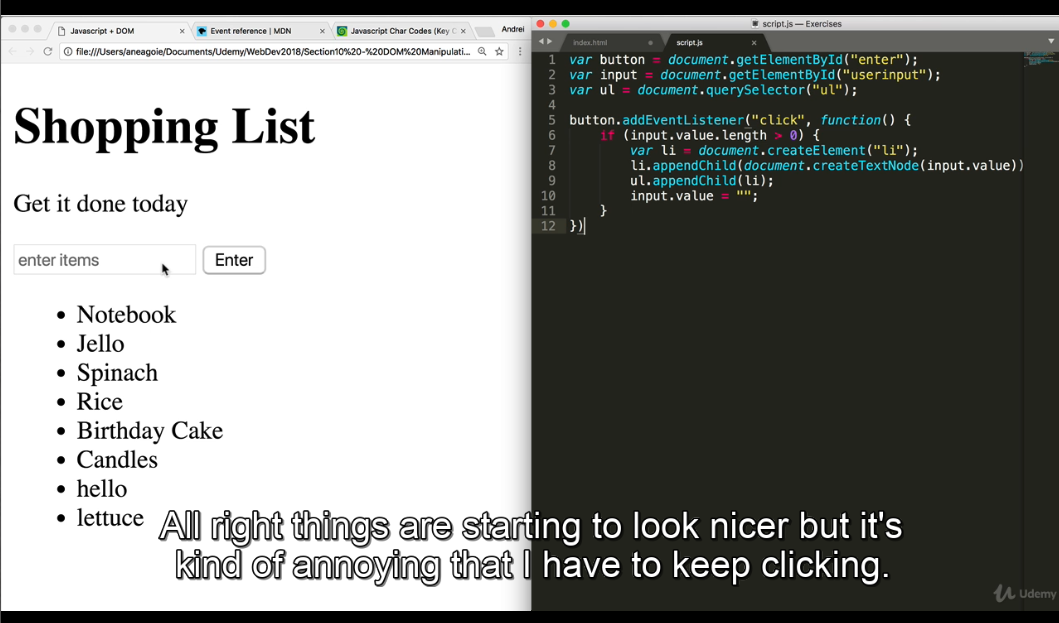
{console.log("click!!");}

);



Ids are more helpful in extracting elements therefore used them

Now the task is to allow the user to enter the thing which gets added to the list



Code:

var button=document.getElementById(“enter”);//variable button can be used to access things of button

var input=document.getElementById(“userinput”);

button.addEventListner(“click”,function(){

if(input.value.length>0)//to access value entered input.value and to find its length input.value.length

{

var li=document.createElement(“li”);//creating new element since we need to add new thing

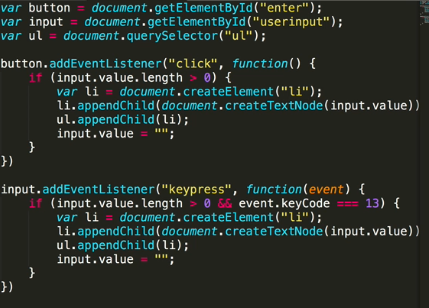
li.appendChild(document.createTextNode(input.value));//each li has something written which is its child node

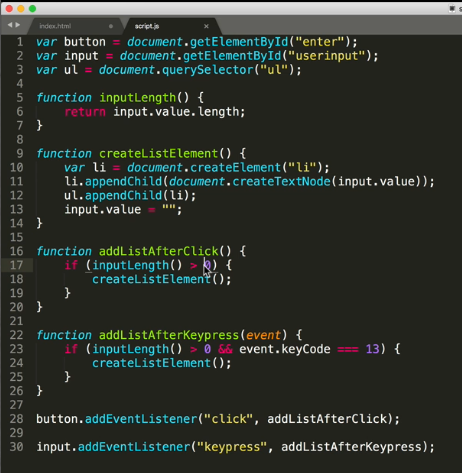
ul.appendChild(li);//li is child of ul

input.value=” “;

}}

If you want the above to work even if enter key is pressed





**ECMA6**

1. **declaring variables**

**i)let**

**scope of variable is within the function itself**

**eg.**

**const player=’bobby’;**

**let experience=100;**

**let wizardlevel= false;**

**if(experience>90)**

**{**

**let wizardlevel=true;**

**console.log(‘inside’,wizardlevel);**

**}**

**console.log(‘outside’,wizardlevel);**

**output:**

**inside true**

**outside false**

**-🡪if we use var**

**const player=’bobby’;**

**let experience=100;**

**var wizardlevel= false;**

**if(experience>90)**

**{**

**var wizardlevel=true;**

**console.log(‘inside’,wizardlevel);**

**}**

**console.log(‘outside’,wizardlevel);**

**output:**

**inside true**

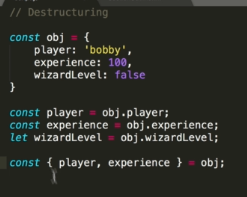
**outside true**

**ii)const**

**you are not able to reassign ,so it cannot be updated**

**we can change properties of an object**

**b)destructuring**

****

**c)template strings**

way of getting output:

var grreting=`Hello ${name} how are you and ${pet}`;

**d)arrow functions**

function add(a,b)

{return a+b;

}

can be written as:

const add=(a,b)=>a+b;

or

const add=(a,b)=>{return a+b;}

**ADVANCED FUNCTIONS**

Childern always have access to their parent scope but parent do not have access to thie child.(Closures)

**a)**CURRYING(a type of function inside function)

const curriedmultipy=(a)=>(b)=>a\*b;

if we enter in the console:curriedMultiply(3);

we get :

(b)=>a\*b

if we enter : curriedmultiply(3)(5);

we get:

15

if we do: const mul5=curriedmultiply(5);

mul5(5);

we get :

25 (Always multiply things by 5)

**b)compose**

COMPOSING: function inside a function.(output of one =input of other)

const compose=(f,g)=>(a)=>f(g(a));

const sum=(num)=>num+1;

compose (sum,sum)(5);

OUTPUT WILL BE : 7

AVOID SIDE EFFECTS ,USE FUNCTIONAL PURITY

FUNTIONAL PURITY IS THAT A FUNCTION ALWAYS RETURNS SOMETHING.

**c)ADVANCED ARRAY**:

i)MAP :

USE MAP INSTEAD OF FOREACH.

const array=[1,2,6];

const maparray=array.map(num=> num\*2);

console.log(maparray);

OUTPUT:2 4 12

ii)FILTER://can filter an array with condition

const filterarray=array.filter(num=>num>3)ssssssss

console.log(filterarray);

OUTPUT: 6

It filters out elements as per the condition

iii )REDUCE :

const reducearray=array.reduce((accumulator,num)=>{

return accumuator+num;

},0); //initial value of accumulator

console.log(reducearray);

OUTPUT:9 //since 1+2=3 adn 3+6=9

**d)ADVANCED OBJECTS :**

i)REFERENCING : object referencing (values inside objects are unique)

ii)CONTEXT: (Different from scope)

this keyword used.

This tells inside which object we are in

Eg

Const object4={

A:function(){

Console.log(this);}} output---🡪 a:f

INSTANTIATION:

It is when you make a copy of an object and reuse the code.

It's just the same as creating an object of a class.(here we use this keyword in constructor (may use)).

eg:

class Player{

constructor(name,type){

this.name=name;

this.type=type;

}

introduce(){

console.log(`HI ${this.name}`);

}

}

class wizard extends Player{ //anytime we use extend we have to call the constructor of parent as well

using super keyword

constructor(name,type){

super(name,type)

}

play(){

console.log(`yoo I am ${this.type});

}

}

const wizard1=new wizard('Ak','Healer');

const wizard2=new wizard('aj','Dark');

**ES7**

pets.includes('1234'); //checks whether it is present in the array or not.Boolean return type.(pets is name of array)

y\*\*3 is eqivalent to y\*y\*y.(\*\* is exponential operator)

**ES8**

.padStart(3) //adds spaces at the end inculding the string .

.padEnd(10)

Object.values(OBJECT NAME)

Object.entries(OBJECT NAME)

Object.keys(OBJECT NAME)

eg:

let obj={

user1:'santa',

user2:'Rud',

user3:'got'

}

Object.values(obj).forEach(value=>{

console.log(value);

})

//This will print the value of each element ..

OUTPUT: santa

Rud

got

Object.entries(obj).forEach(value=>{

console.log(value);

})

//prints username as well as name

OUTPUT: ["user1","santa"]..and so on

Object.entries(obj).map(value=>{

return value[1]+value[0].replace('user',' ');

})

OUTPUT: ["santa1","Rud2","got3"]

**ADVANCED LOOPING:**

const basket=['apple','oranges','grapes'];

const detbasket={

apples:5,

oranges:10;

grapes:100

}

**i)FOR OF**

for(item of basket){

console.log(item);

}

//prints each item in the array.

**ii)FOR IN:it works with objects.**

for(item in detbasket){

console.log(item);

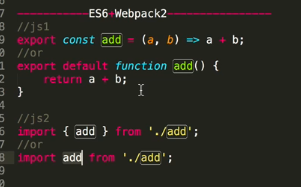
}

OUTPUT: apples

oranges

grapes //here we are not iterating but ENUMERATING.

Enumerating is for objects.



**Using the terminal**

dir - list files

cd {directory name} - change directory

cd / - go to root (top) directory

cd .. - go up a level

mkdir {file name} - make a directory

echo > {filename} - create an empty file

del {filename} - remove a file

rmdir {directory name} - remove a directory and all files within

rename {filename} {new filename} - rename a file or folder

start {filename} - open file in default program

start . - open current directory

cls - clear the terminal screen

**github**

git clone

git status

git add index.html

git add style.css

git commit -m “ANY MSG”

git push

git pull

**branches**

git branch

git branch name\_of\_new\_br

git checkout name\_of\_br\_to\_switch

git merge master

git push origin conflict

**REACT**