Q1. Given this array: `[3,62,234,7,23,74,23,76,92]`, Using arrow function, create an array of the numbers greater than `70`.

Ans.

An arrow function expression is a syntactically compact alternative to a regular function expression, although without its own bindings to the this, arguments, super, or new.target keywords.

Arrow functions serve two main purposes:
-> more concise syntax (a shorthand way of declaring functions)
-> sharing lexical this with the parent scope. (shares the scope with the parent)

```
Filter output                                                    Persist Logs
>> var getno = () => {
    for(let i=0;i<arr.length;i++){
       if(arr[i]>70)
        console.log(arr[i]);
     }
   }

   let arr=[3,62,234,7,23,74,23,76,92]
   getno(...arr); // 11
   234                                                  debugger eval code:4:6
   74                                                   debugger eval code:4:6
   76                                                   debugger eval code:4:6
   92                                                   debugger eval code:4:6
```

Q2.

```
<ul>

        <li data-time="5:17">Flexbox Video</li>

        <li data-time="8:22">Flexbox Video</li>

        <li data-time="3:34">Redux Video</li>

        <li data-time="5:23">Flexbox Video</li>

        <li data-time="7:12">Flexbox Video</li>

        <li data-time="7:24">Redux Video</li>

        <li data-time="6:46">Flexbox Video</li>

        <li data-time="4:45">Flexbox Video</li>

        <li data-time="4:40">Flexbox Video</li>
```

```
<li data-time="7:58">Redux Video</li>

<li data-time="11:51">Flexbox Video</li>

<li data-time="9:13">Flexbox Video</li>

<li data-time="5:50">Flexbox Video</li>

<li data-time="5:52">Redux Video</li>

<li data-time="5:49">Flexbox Video</li>

<li data-time="8:57">Flexbox Video</li>

<li data-time="11:29">Flexbox Video</li>

<li data-time="3:07">Flexbox Video</li>

<li data-time="5:59">Redux Video</li>

<li data-time="3:31">Flexbox Video</li>
```

</ul>

1. Select all the list items on the page and convert to array.

**Ans**

**i)**

JavaScript ▾

```
var x=document.querySelectorAll("li");
  var arr=[];
  for (var i = 0; i < x.length; i++) {
  arr.push(x[i].innerText);

}
console.log(arr);
```

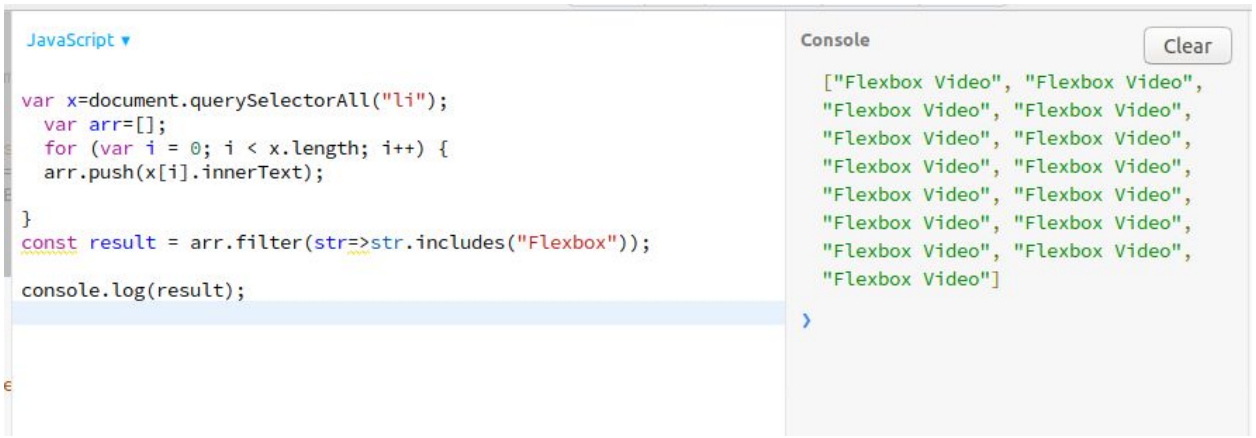
Console

Clear

```
["Flexbox Video", "Flexbox Video",
"Redux Video", "Flexbox Video",
"Flexbox Video", "Redux Video",
"Flexbox Video", "Flexbox Video",
"Flexbox Video", "Redux Video",
"Flexbox Video", "Flexbox Video",
"Flexbox Video", "Redux Video",
"Flexbox Video", "Flexbox Video",
"Flexbox Video", "Flexbox Video",
"Redux Video", "Flexbox Video"]
```

›

2. Filter for only the elements that contain the word 'flexbox'




3. map down to a list of time strings

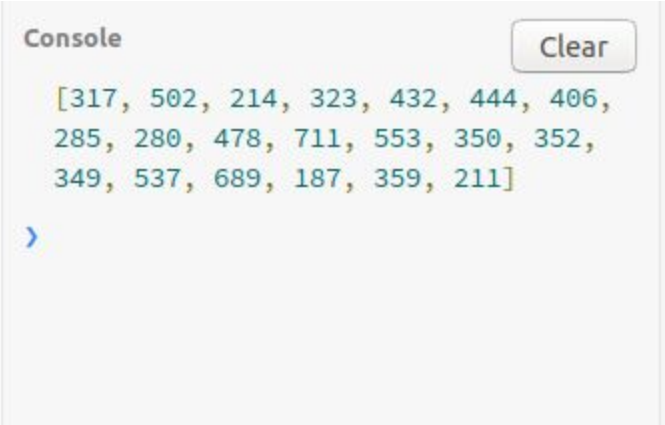
```
var x=document.querySelectorAll("li");
  var arr=[];
  for (var i = 0; i < x.length; i++) {
  arr.push(x[i].innerText);

}
var temp=Array.prototype.slice.call(x);

var arr2=temp.map(XYZ=>{
                    return XYZ.dataset.time;});
```

console.log(arr2);


4. map to an array of seconds

```
var x=document.querySelectorAll("li");

  var arr=[];

  for (var i = 0; i < x.length; i++) {

  arr.push(x[i].innerText);


}
```

```
var temp=Array.prototype.slice.call(x);

var arr2=temp.map(XYZ=>{

        var time= XYZ.dataset.time;
        var convertToSec=time.split(':');

        return (convertToSec[0])*60 +parseInt(convertToSec[1]);

});

console.log(arr2);
```



5. reduce to get total using .filter and .map

```
  var x=document.querySelectorAll("li");
 var arr=[];
 for (var i = 0; i < x.length; i++) {
 arr.push(x[i].innerText);

}
```

```
function lengthofArray(){
let result=arr.filter(str=>str.includes("Flexbox"))
return result.length;
}
```

```
var temp=Array.prototype.slice.call(x);

var arr2=temp.map(XYZ=>{
```

```
                var time= XYZ.dataset.time;
                var convertToSec=time.split(':');

                return (convertToSec[0])*60 +parseInt(convertToSec[1]);

});

var reduceTime=arr2.reduce ((total,value)=>{
  return total+value;

});


var lenofArray=lengthofArray();

var obj={Flexbox_Video:lenofArray,Timeset:reduceTime }

console.log(obj)
```

```
Console                                  Clear
  [object Object] {
    Flexbox_Video: 15,
    Timeset: 7979
  }
›
```

**Q3.** Create a markup template using string literal

```
const song = {
 name: 'Dying to live',
 artist: 'Tupac',
 featuring: 'Biggie Smalls'
};
```

Result:

```
"<div class="song">
   <p>
     Dying to live — Tupac
     (Featuring Biggie Smalls)
   </p>
 </div>
“
```

**Ans.**

Template literals are string literals allowing embedded expressions. You can use multi-line strings and string interpolation features with them.

Template literals are enclosed by the back-tick (` `) character instead of double or single quotes. Template literals can contain placeholders. These are indicated by the dollar sign and curly braces (${expression}).

**Code:**

```
console.log(`<div class="song">

 <p>

        ${song.name} - ${song.artist}

        (Featuring ${song.featuring})

</p>

</div>`);
```

```
» console.log(`<div class="song">
    <p>
       ${song.name} - ${song.artist}
       (Featuring ${song.featuring})
   </p>
   </div>`);
   <div class="song">                                  debugger eval code:1:1
    <p>
       Dying to live - Tupac
       (Featuring Biggie Smalls)
   </p>
   </div>
← undefined
```

**Q4.** Extract all keys inside address object from user object using destructuring ?
const user = {
firstName: ‘Sahil’,
lastName: ‘Dua’,
Address: {
Line1: ‘address line 1’,
Line2: ‘address line 2’,
State: ‘Delhi’,
Pin: 110085,
Country: ‘India’,
City: ‘New Delhi’,
},
phoneNo: 9999999999
}

Ans.

The **destructuring assignment** syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

Ability to extract values from objects or arrays into variables.

```
← undefined
» let {Address:{Line1,Line2,State,Pin,Country,City}}=user;
← undefined
» console.log(Line1);
  console.log(Line2);
  console.log(State);
  console.log(Pin);

  console.log(Country);
  console.log(City);
  address line 1                                  debugger eval code:1:1
  address line 2                                  debugger eval code:2:1
  Delhi                                           debugger eval code:3:1
  110085                                          debugger eval code:4:1
  India                                           debugger eval code:6:1
  New Delhi                                       debugger eval code:7:1
```