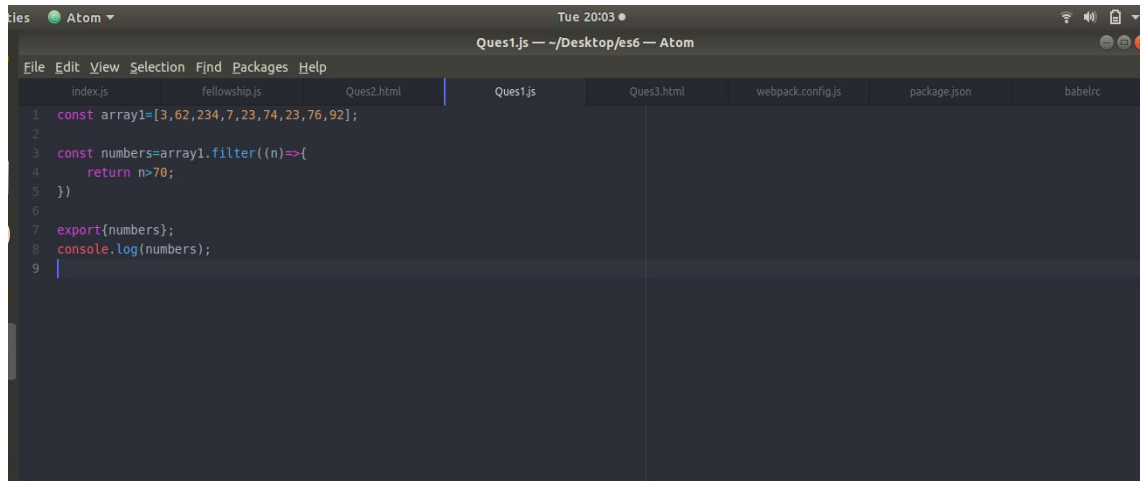


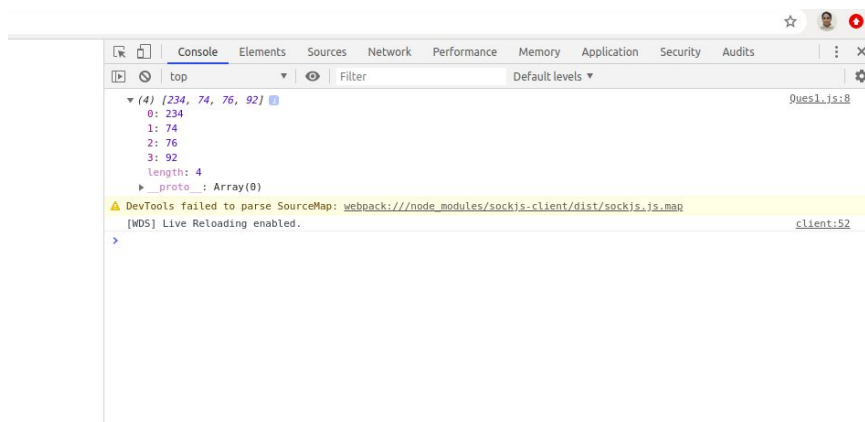
ES6:Part6

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`.



```
1 const array1=[3,62,234,7,23,74,23,76,92];
2
3 const numbers=array1.filter((n)=>{
4   return n>70;
5 })
6
7 export{numbers};
8 console.log(numbers);
9
```

OUTPUT:



Q2.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

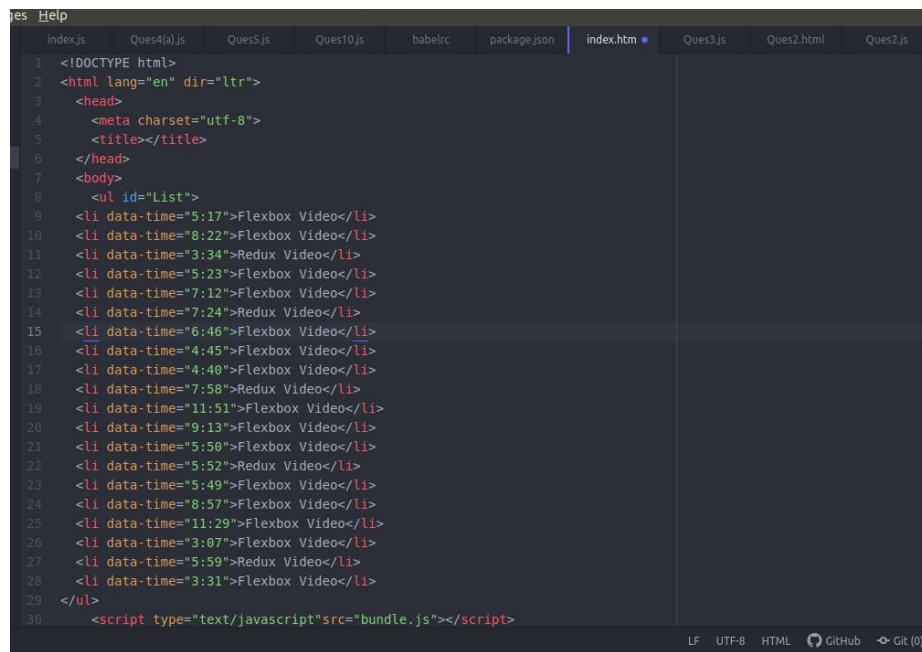
<li data-time="11:29">Flexbox Video

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

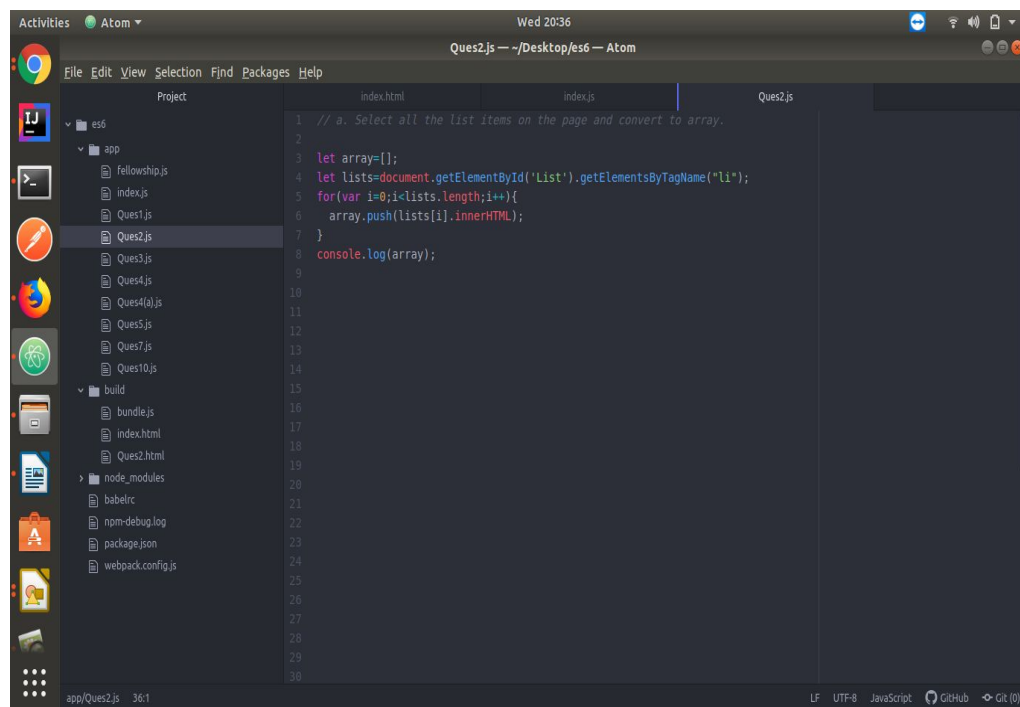
<li data-time="5:59">Redux Video

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

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

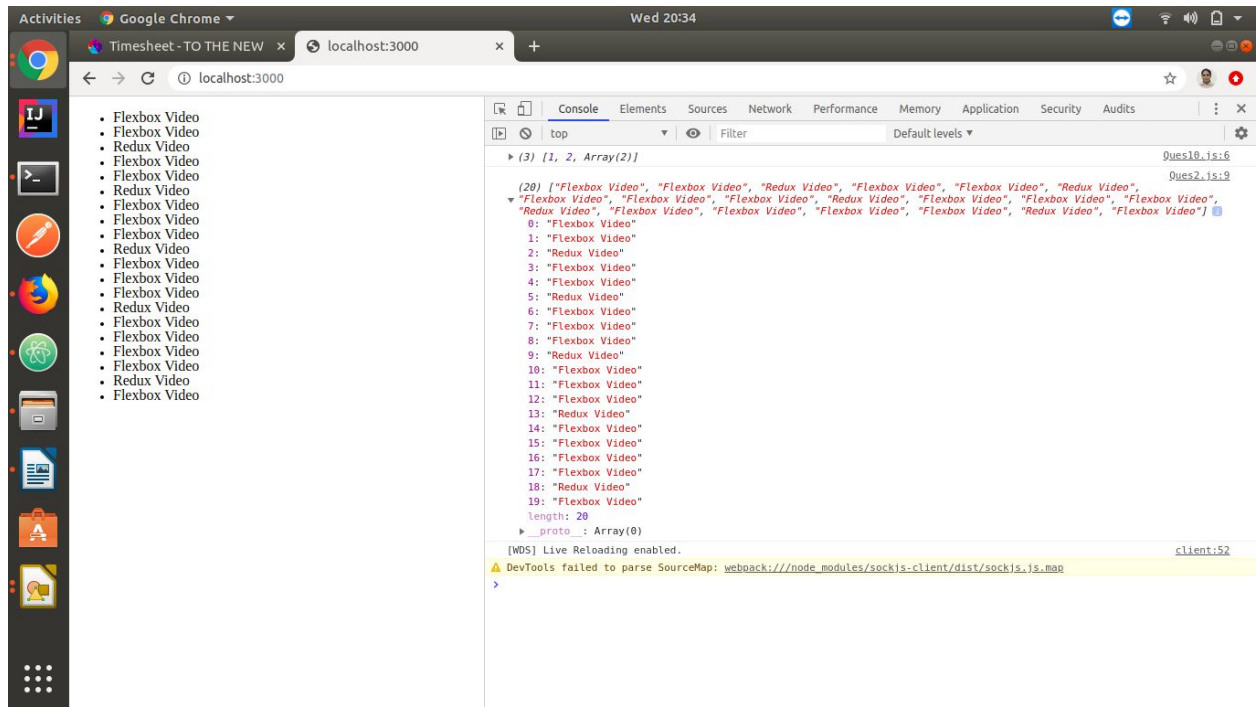


```
1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3   <head>
4     <meta charset="utf-8">
5     <title></title>
6   </head>
7   <body>
8     <ul id="List">
9       <li data-time="5:17">Flexbox Video</li>
10      <li data-time="8:22">Flexbox Video</li>
11      <li data-time="3:34">Redux Video</li>
12      <li data-time="5:23">Flexbox Video</li>
13      <li data-time="7:12">Flexbox Video</li>
14      <li data-time="7:24">Redux Video</li>
15      <li data-time="6:46">Flexbox Video</li>
16      <li data-time="4:45">Flexbox Video</li>
17      <li data-time="4:40">Flexbox Video</li>
18      <li data-time="7:58">Redux Video</li>
19      <li data-time="11:51">Flexbox Video</li>
20      <li data-time="9:13">Flexbox Video</li>
21      <li data-time="5:50">Flexbox Video</li>
22      <li data-time="5:52">Redux Video</li>
23      <li data-time="5:49">Flexbox Video</li>
24      <li data-time="8:57">Flexbox Video</li>
25      <li data-time="11:29">Flexbox Video</li>
26      <li data-time="3:07">Flexbox Video</li>
27      <li data-time="5:59">Redux Video</li>
28      <li data-time="3:31">Flexbox Video</li>
29    </ul>
30    <script type="text/javascript"src="bundle.js"></script>
```

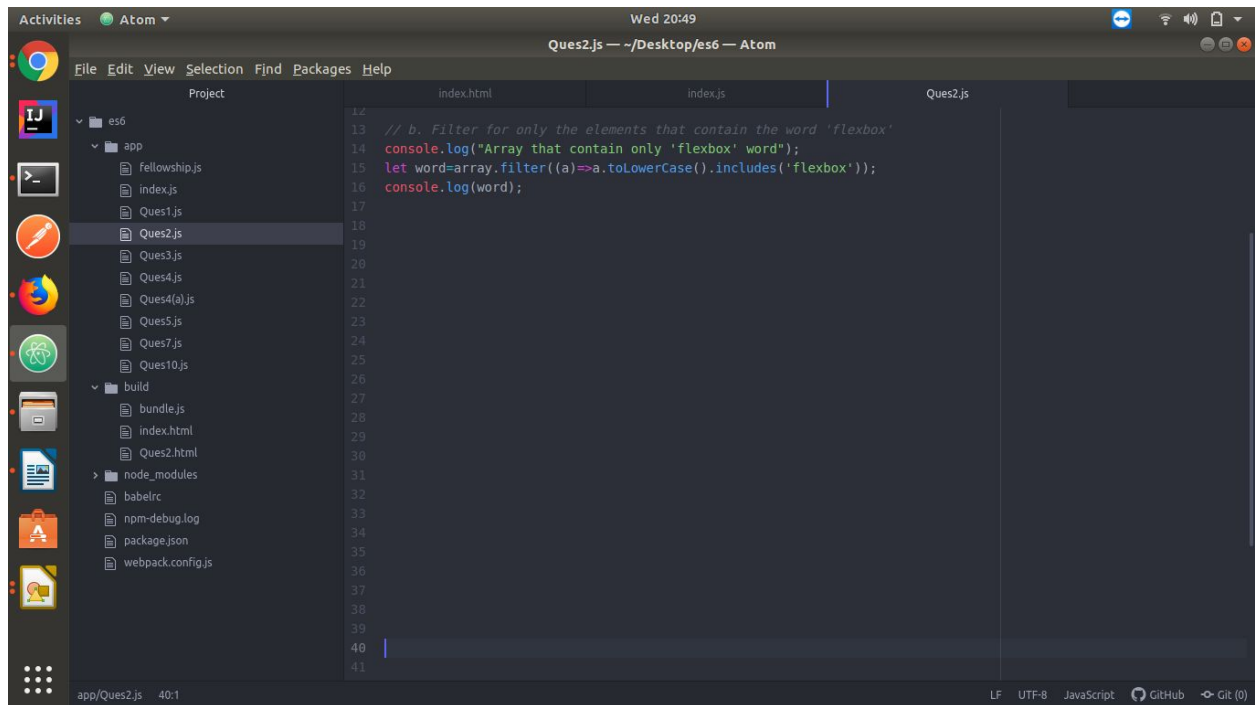


```
1 // a. Select all the list items on the page and convert to array.
2
3 let array=[];
4 let lists=document.getElementById('List').getElementsByTagName("li");
5 for(var i=0;i<lists.length;i++){
6   array.push(lists[i].innerHTML);
7 }
8 console.log(array);
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

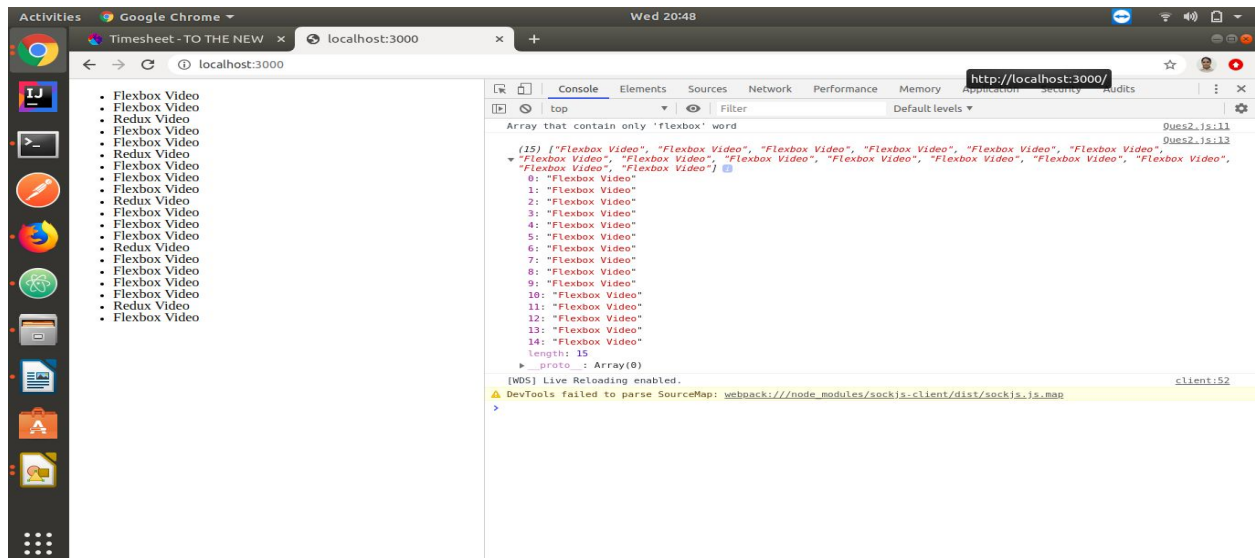
OUTPUT:



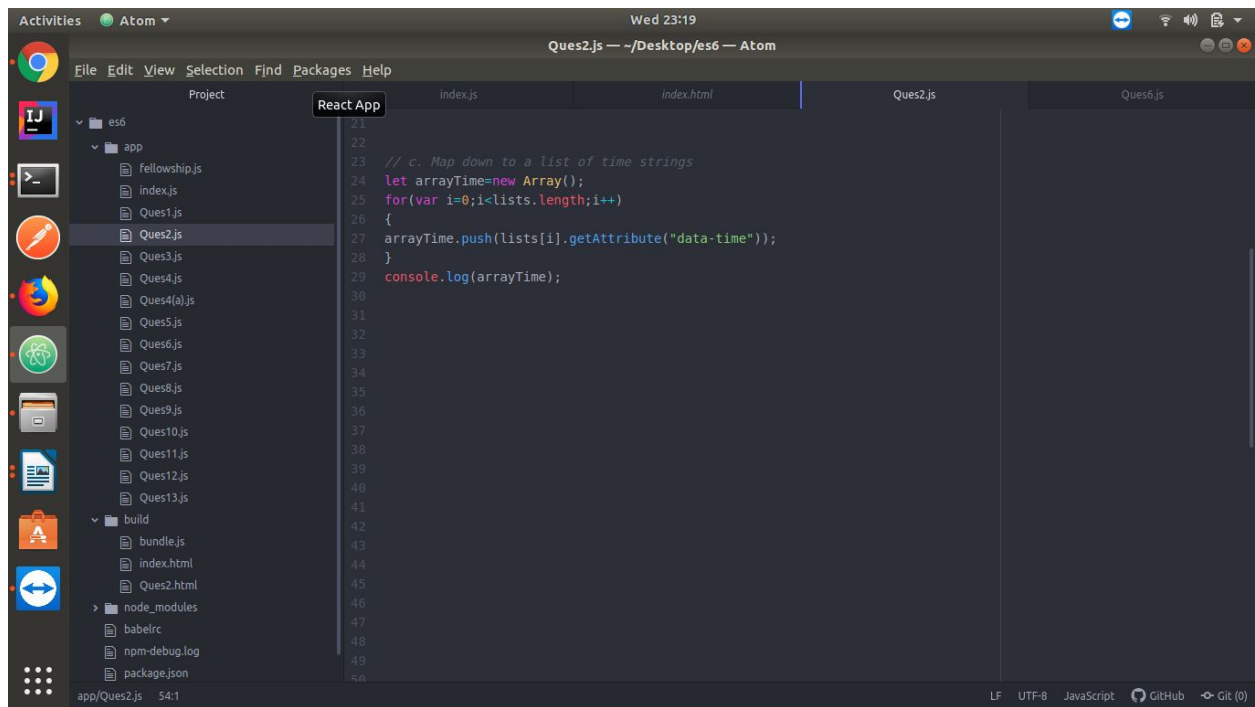
b. Filter for only the elements that contain the word 'flexbox'.



OUTPUT:

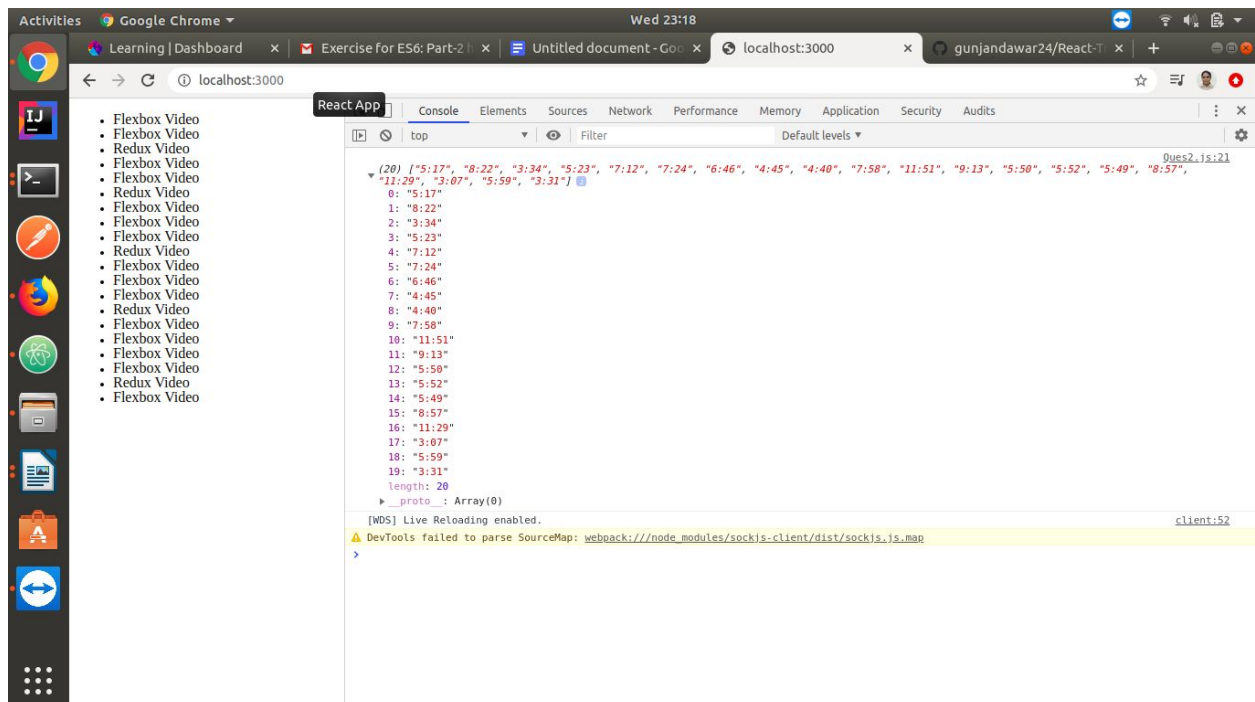


c. Map down to a list of time strings



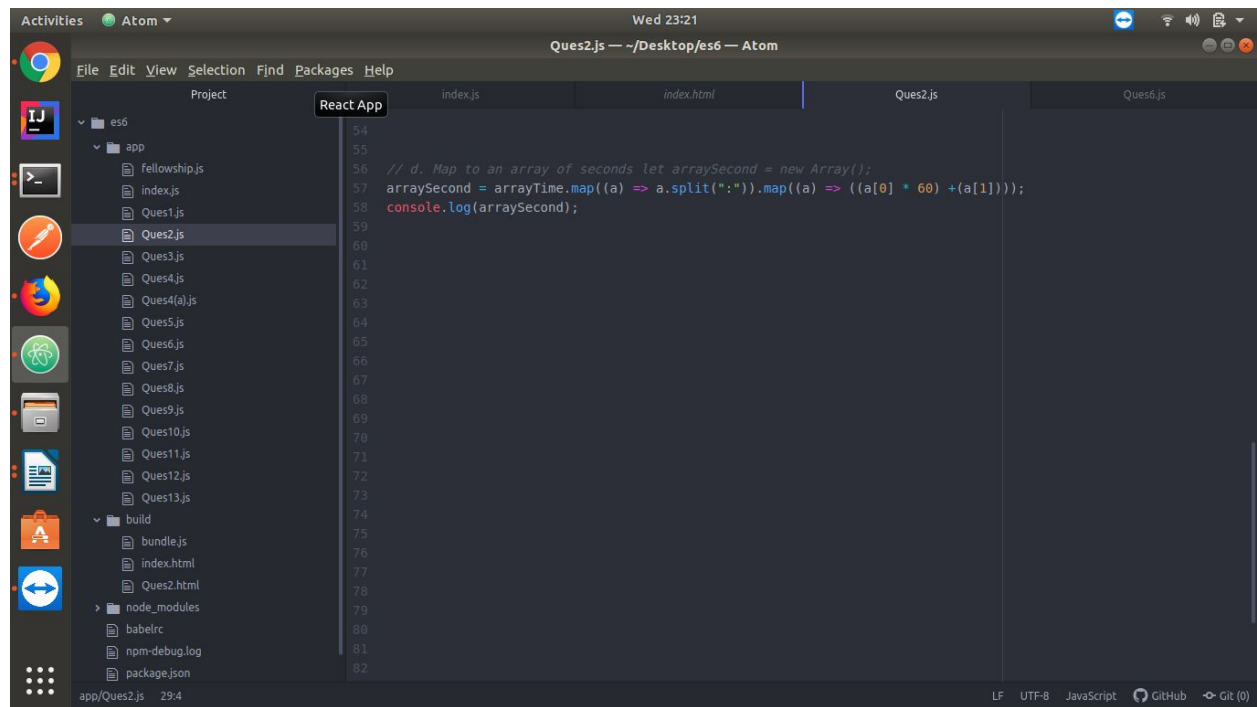
```
21
22
23 // c. Map down to a list of time strings
24 let arrayTime=new Array();
25 for(var i=0;i<lists.length;i++)
26 {
27   arrayTime.push(lists[i].getAttribute("data-time"));
28 }
29 console.log(arrayTime);
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
```

OUTPUT:



```
(20) ["5:17", "8:22", "3:34", "5:23", "7:12", "7:24", "6:46", "4:45", "4:40", "7:58", "11:51", "9:13", "5:50", "5:52", "5:49", "8:57", "11:29", "3:07", "5:59", "3:31"]
0: "5:17"
1: "8:22"
2: "3:34"
3: "5:23"
4: "7:12"
5: "7:24"
6: "6:46"
7: "4:45"
8: "4:40"
9: "7:58"
10: "11:51"
11: "9:13"
12: "5:50"
13: "5:52"
14: "5:49"
15: "8:57"
16: "11:29"
17: "3:07"
18: "5:59"
19: "3:31"
length: 20
__proto__: Array(0)
```

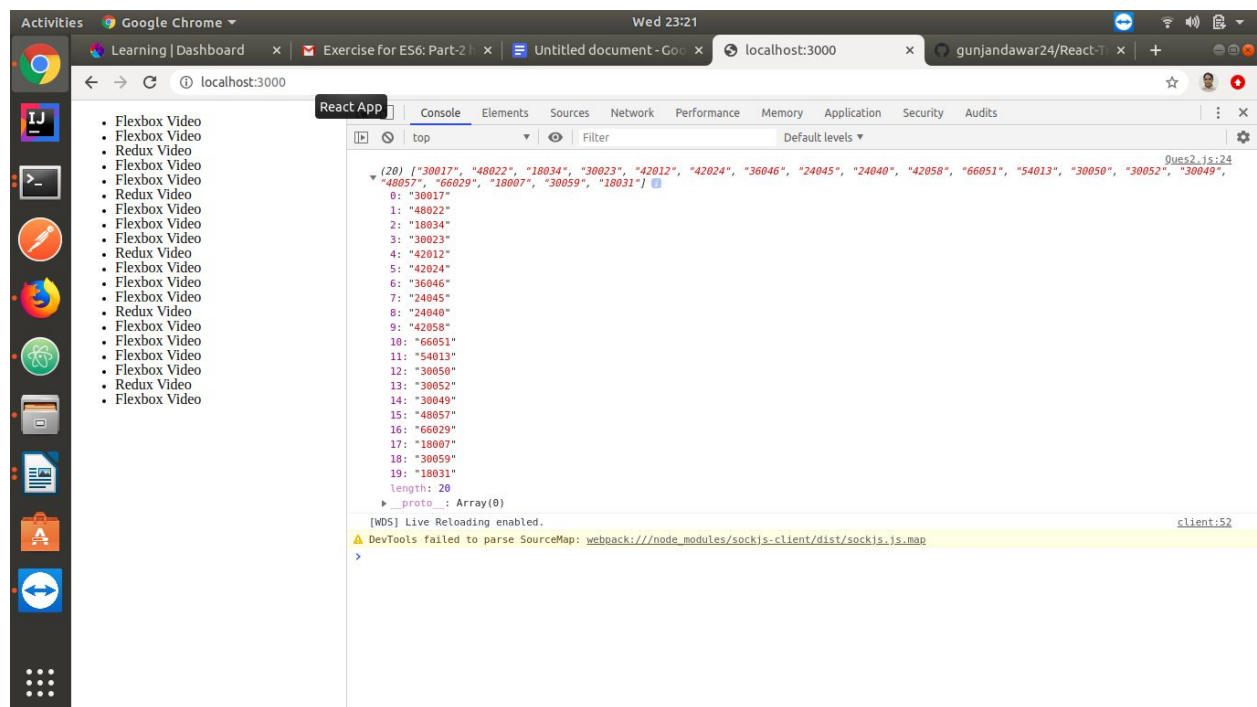
d. Map to an array of seconds



The screenshot shows the Atom editor with a project named 'React App'. The file explorer on the left shows a directory structure with 'app' and 'build' folders. The 'app' folder contains files from 'fellowship.js' to 'Ques13.js'. The 'build' folder contains 'bundle.js', 'index.html', and 'Ques2.html'. The 'node_modules' folder contains 'babelrc', 'npm-debug.log', and 'package.json'. The main editor window shows the 'Ques2.js' file with the following code:

```
54  
55  
56 // d. Map to an array of seconds let arraySecond = new Array();  
57 arraySecond = arrayTime.map(a => a.split(":").map(a => ((a[0] * 60) + (a[1]))));  
58 console.log(arraySecond);  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82
```

OUTPUT:

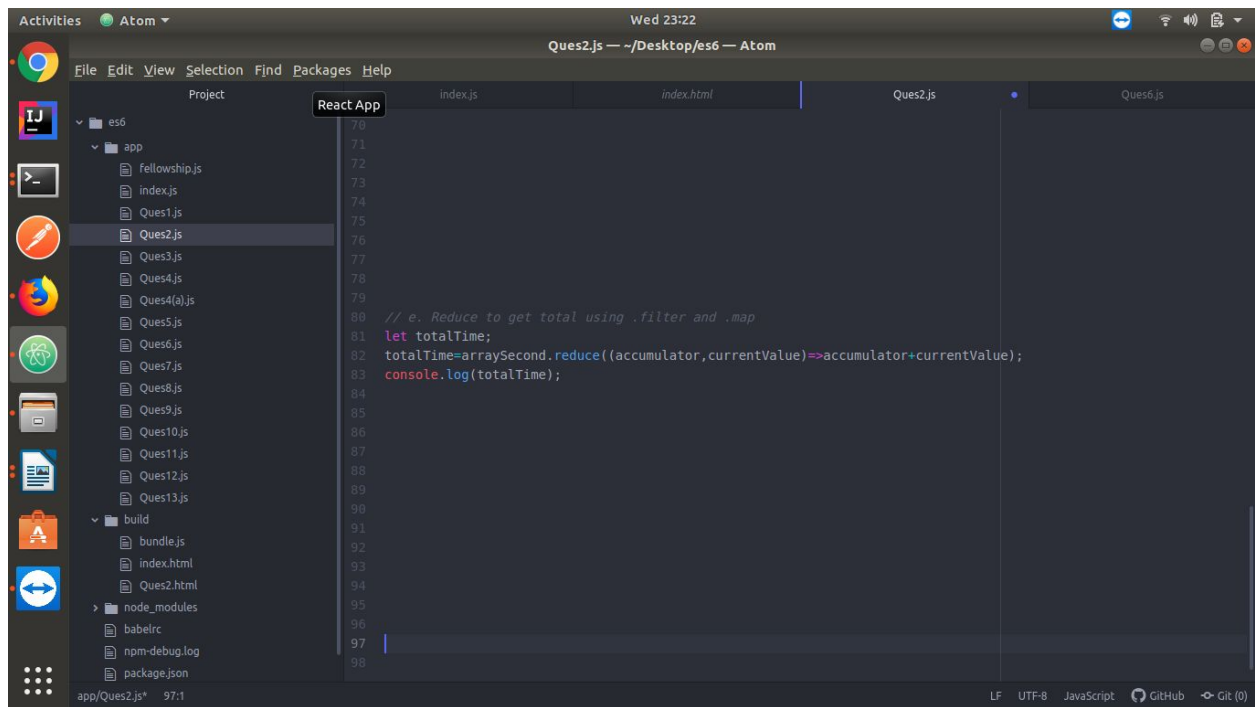


The screenshot shows the Google Chrome browser with the 'React App' running on 'localhost:3000'. The DevTools console is open, showing the output of the map function. The output is an array of 20 elements, each representing a time string converted to seconds. The array is displayed as follows:

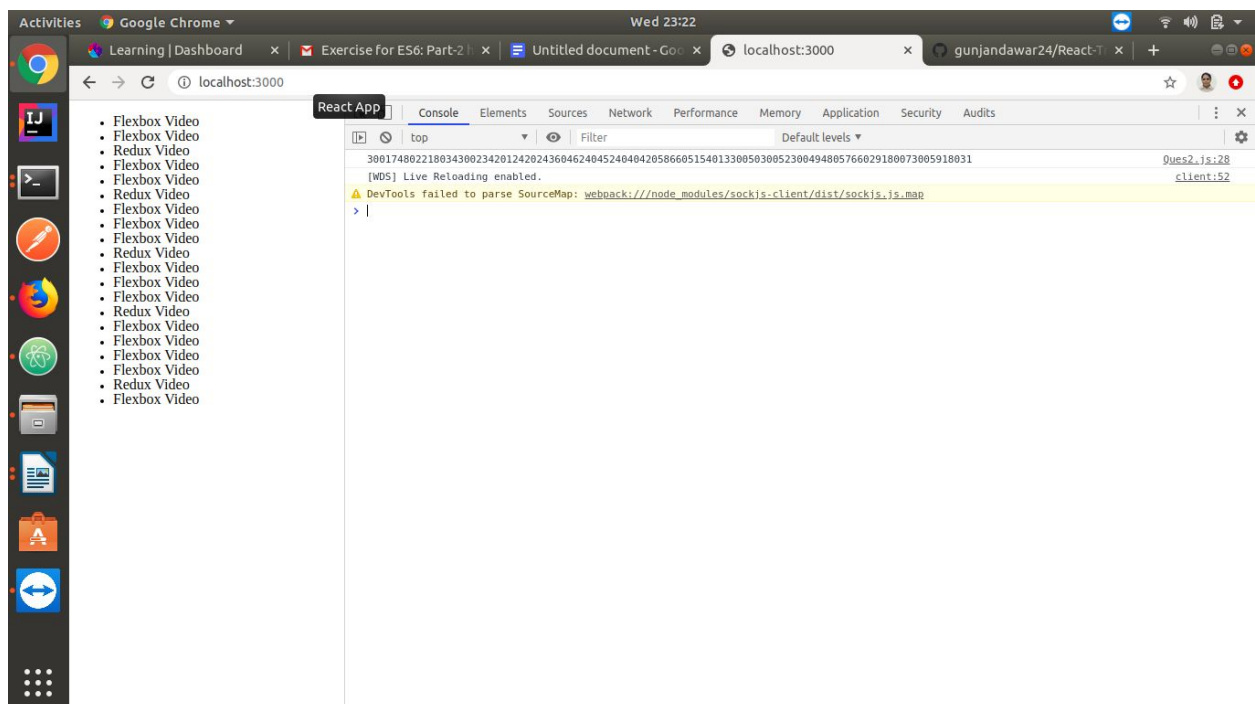
```
(20) ["30017", "48022", "18034", "30023", "42012", "42024", "36046", "24045", "24040", "42058", "66051", "54013", "30050", "30052", "30049",  
0: "30017"  
1: "48022"  
2: "18034"  
3: "30023"  
4: "42012"  
5: "42024"  
6: "36046"  
7: "24045"  
8: "24040"  
9: "42058"  
10: "66051"  
11: "54013"  
12: "30050"  
13: "30052"  
14: "30049"  
15: "48057"  
16: "66029"  
17: "18007"  
18: "30059"  
19: "18031"  
length: 20  
__proto__: Array(0)
```

Below the array, the DevTools console shows a message: '[WDS] Live Reloading enabled.' and a warning: 'DevTools failed to parse SourceMap: webpack:///node_modules/sockjs-client/dist/sockjs.js.map'.

e. Reduce to get total using .filter and .map



OUTPUT:



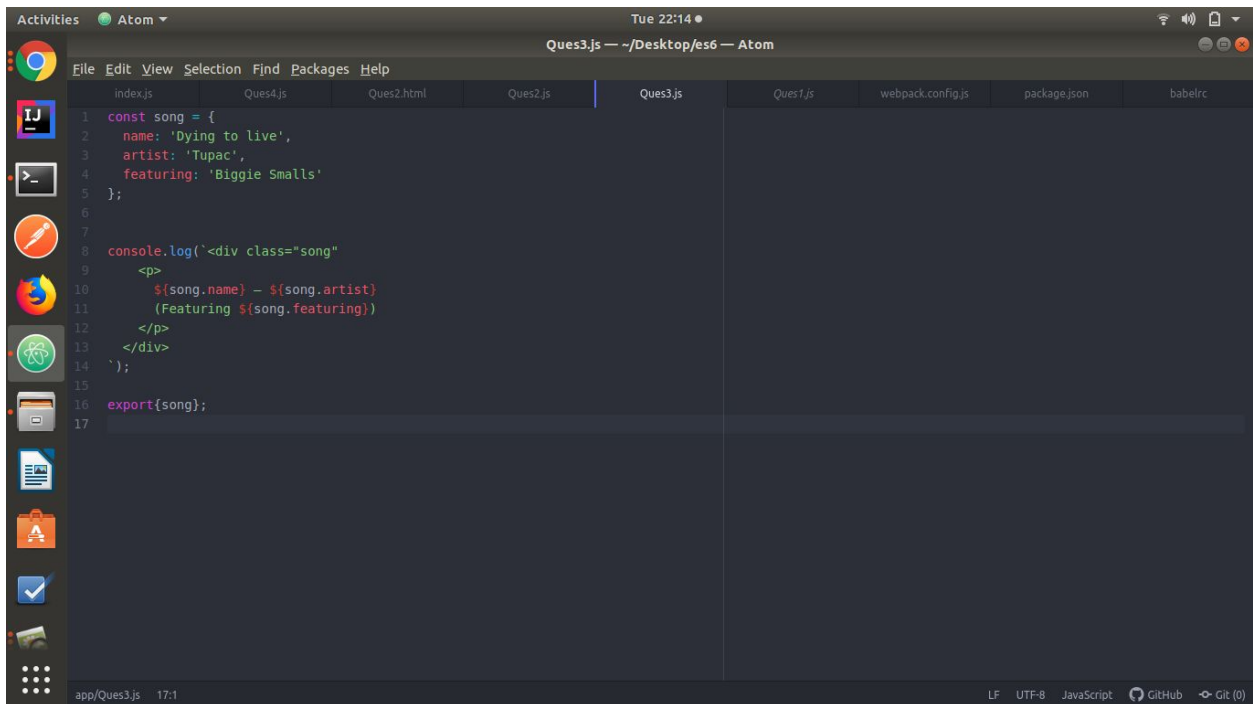
Q3. Create a markup template using string literal

```
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>
```

“



The screenshot shows the Atom code editor with a dark theme. The file 'Ques3.js' is open, showing the following code:

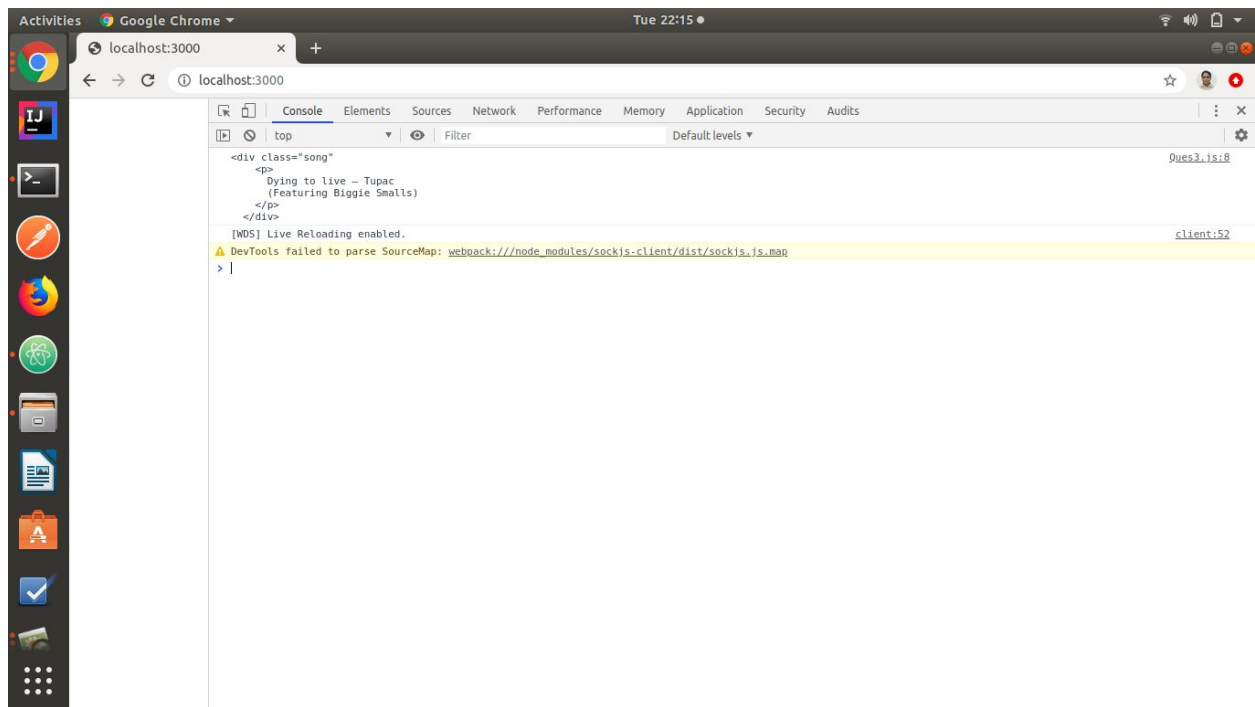
```
1 const song = {  
2   name: 'Dying to live',  
3   artist: 'Tupac',  
4   featuring: 'Biggie Smalls'  
5 };  
6  
7  
8 console.log('<div class="song"  
9  
10   <p>  
11     ${song.name} — ${song.artist}  
12     (Featuring ${song.featuring})  
13   </p>  
14 </div>  
15 `);  
16  
17 export(song);
```

The console output on the right shows the rendered HTML template:

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

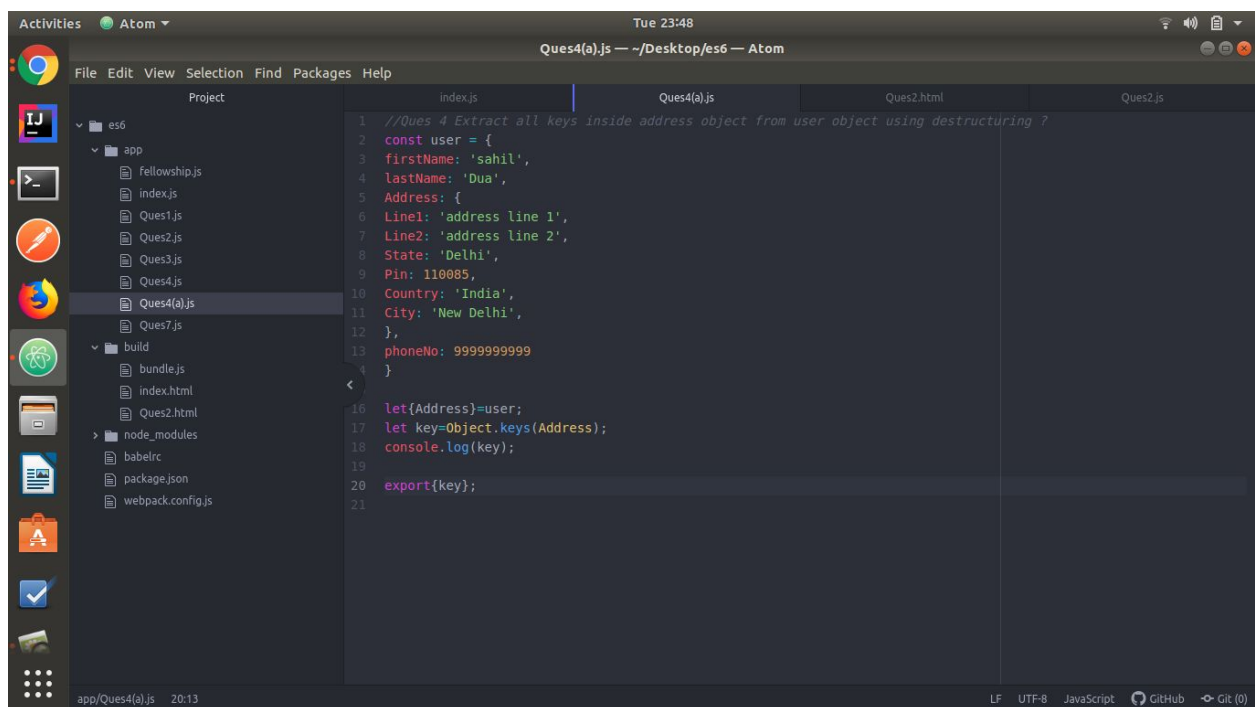
The status bar at the bottom indicates the file is 'app/Ques3.js' at line 17, column 1, with a UTF-8 encoding and JavaScript language.

OUTPUT:



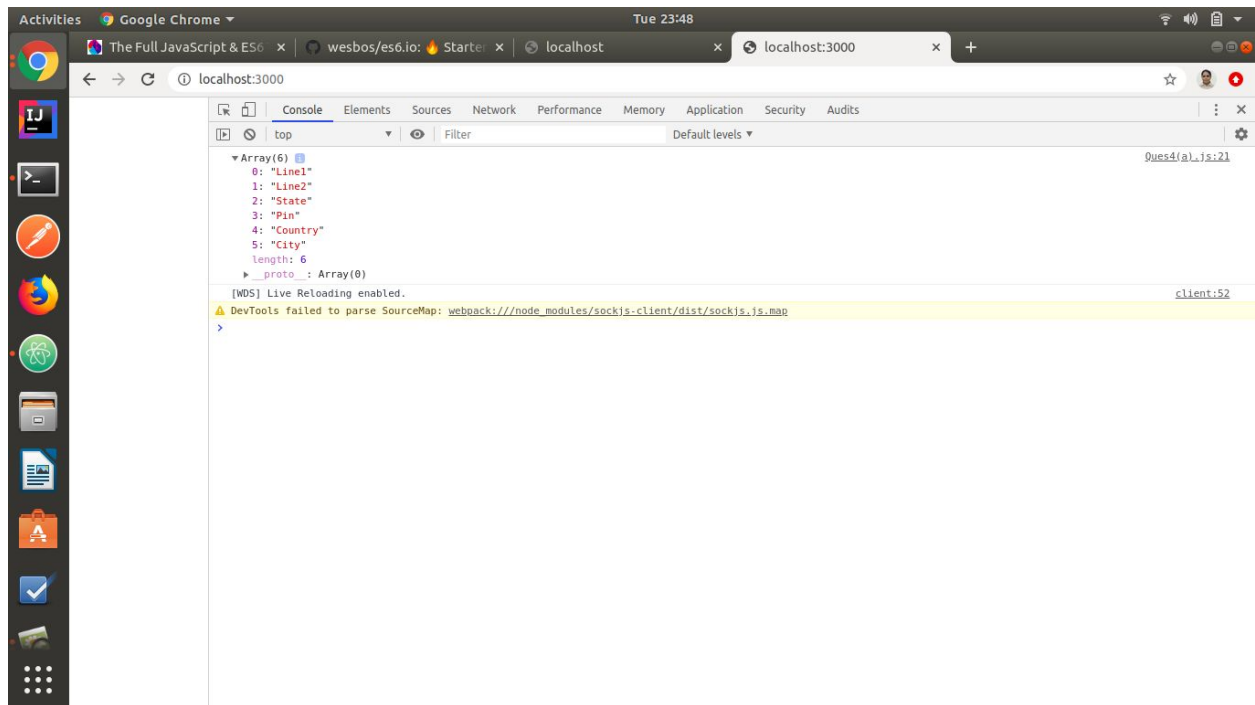
Q4. Extract all keys inside address object from user object using destructuring ?

```
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  
}
```

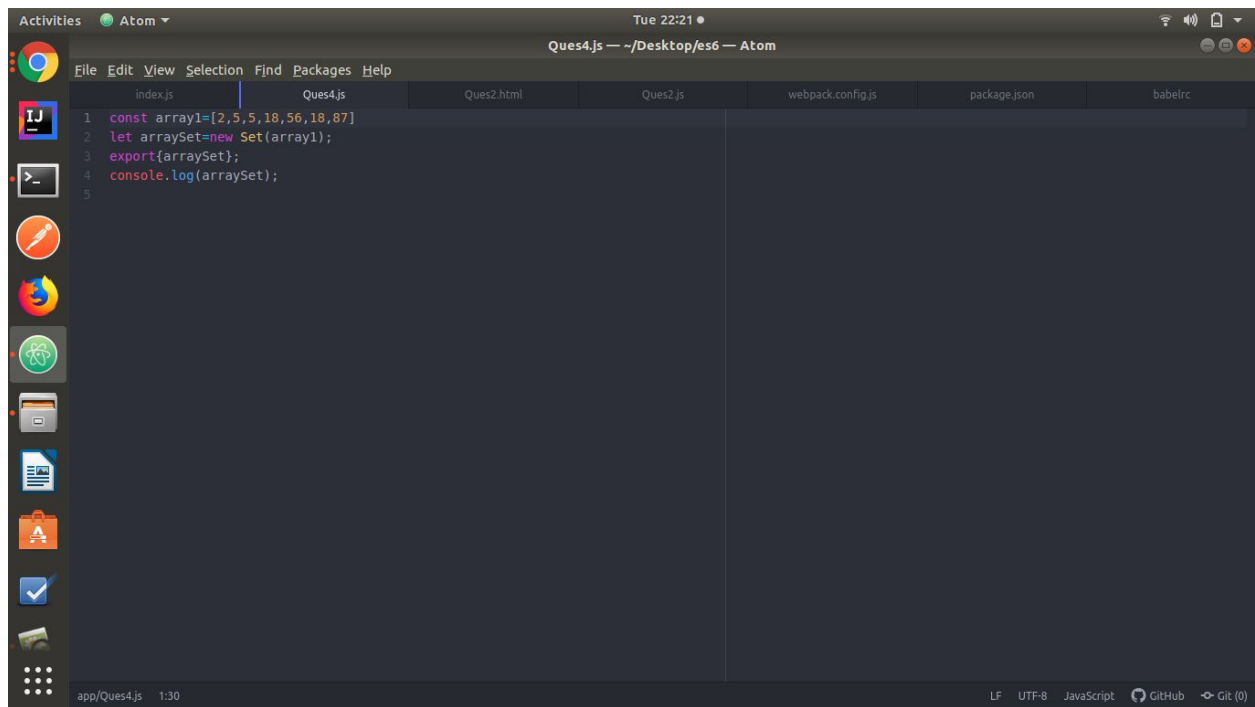


```
//Ques 4 Extract all keys inside address object from user object using destructuring ?  
const user = {  
  firstName: 'sahil',  
  lastName: 'Dua',  
  Address: {  
    Line1: 'address line 1',  
    Line2: 'address line 2',  
    State: 'Delhi',  
    Pin: 110085,  
    Country: 'India',  
    City: 'New Delhi',  
  },  
  phoneNo: 9999999999  
}  
  
let {Address}=user;  
let key=Object.keys(Address);  
console.log(key);  
export{key};
```

OUTPUT:



Q4. Filter unique array members using Set.

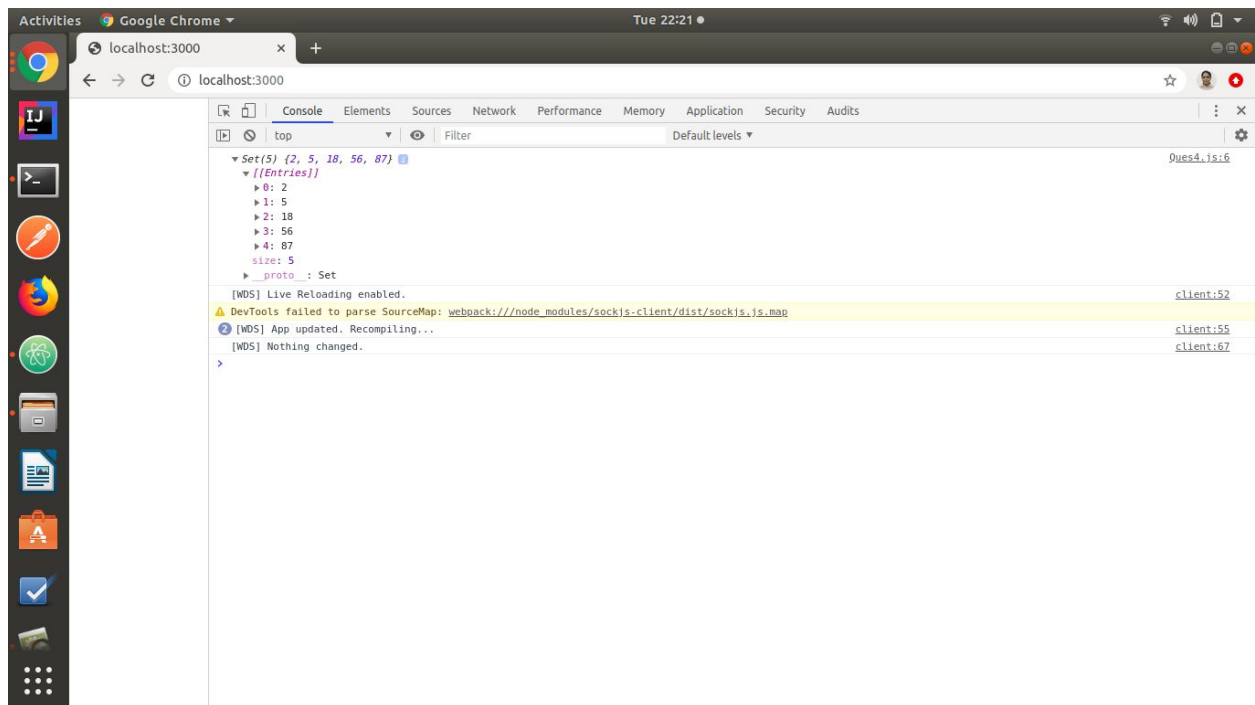


The screenshot shows the Atom editor interface with a file named `Ques4.js` open. The code in the editor is as follows:

```
1 const array1=[2,5,5,18,56,18,87]
2 let arraySet=new Set(array1);
3 export{arraySet};
4 console.log(arraySet);
5
```

The editor's status bar at the bottom indicates the file is `app/Ques4.js`, 1:30 lines, using the `LF` line ending, `UTF-8` encoding, `JavaScript` language, and is connected to `GitHub` and `Git (0)`.

OUTPUT:



The screenshot shows the Google Chrome DevTools Console with the output of the code from `Ques4.js`. The output is a `Set` object containing the unique elements of the array: `Set(5) {2, 5, 18, 56, 87}`. The console also shows messages from Webpack Dev Server (WDS) and a warning about a failed SourceMap parse.

```
Set(5) {2, 5, 18, 56, 87}
  [[Entries]]
    0: 2
    1: 5
    2: 18
    3: 56
    4: 87
  size: 5
  __proto__: Set
[WDS] Live Reloading enabled.
[WDS] App updated. Recompiling...
[WDS] Nothing changed.
```

Q5. Find the possible combinations of a string and store them in a MAP?

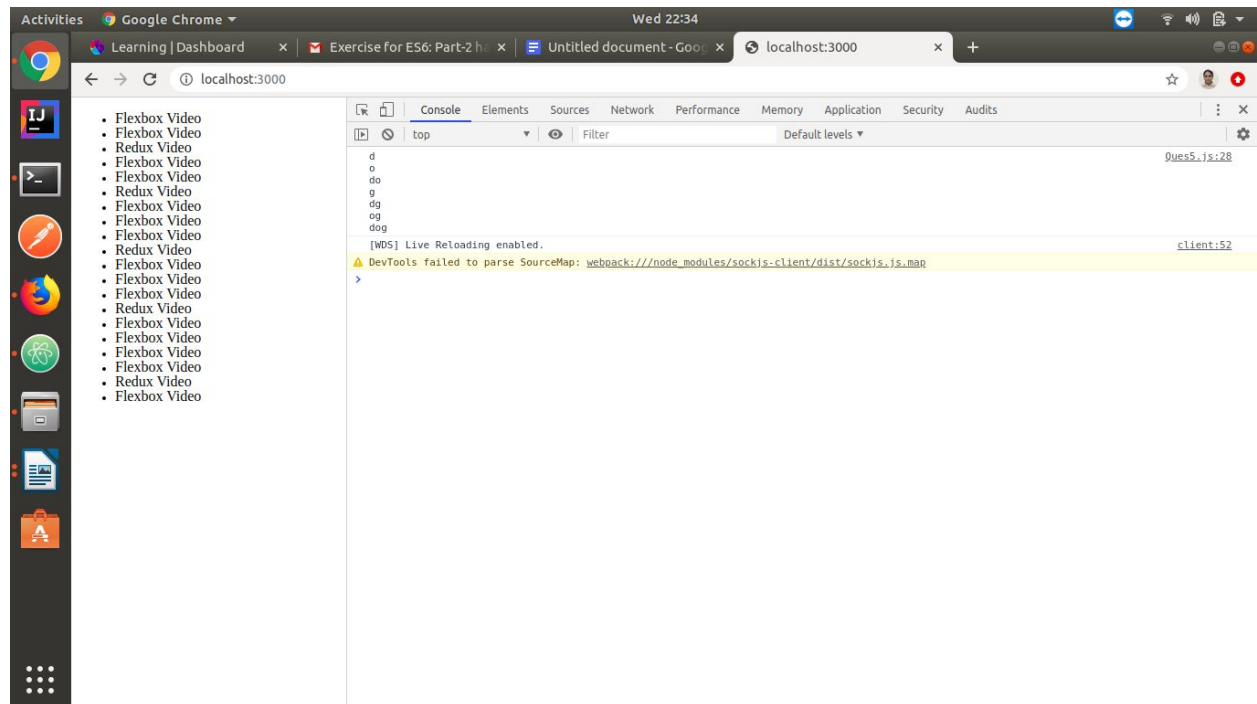
```
function substrings(str1)
{
var array1 = [];
  for (var x = 0, y=1; x < str1.length; x++,y++)
  {
    array1[x]=str1.substring(x, y);
  }
```

```
var combi = [];
var temp= "";
var slent = Math.pow(2, array1.length);
```

```
for (var i = 0; i < slent ; i++)
{
  temp= "";
  for (var j=0;j<array1.length;j++) {
    if ((i & Math.pow(2,j))) {
      temp += array1[j];
    }
  }
  if (temp !== "")
  {
    combi.push(temp);
  }
}
console.log(combi.join("\n"));
}
```

```
substrings("dog");  
export{substrings};
```

OUTPUT:



Q6. Write a program to implement inheritance upto 3 classes. The Class must public variables and static functions.

```
class Animal {  
    constructor(name, height) {  
        this.name = name;  
        this.height = height;  
    }  
  
    hello() {  
        console.log(`Hi I am ${this.name} from animal class`);  
    }  
  
    static sound() {  
        return "Wohooooooooo";  
    }  
}
```

```
class Lion extends Animal {  
    constructor(name, height, fur) {  
        super(name, height);  
        this.fur = fur;  
    }  
    hello() {  
        console.log(`Hi I am ${this.name} from Lion class`);  
    }  
    static sound() {  
        return "Wohooooo Lion"  
    }  
}
```



```
}
```

```
class Tiger extends Lion {  
  constructor(name, height, fur, color) {  
    super(name, height, fur)  
    this.color = color;  
  }  
  hello() {  
  
    console.log(`Hi I am ${this.name} from tiger class`);  
  }  
  static colour() {  
    return "Yellow";  
  }  
}
```

```
let king = new Animal("Animal", 6.5);  
console.log(king);  
king.hello();  
console.log("-----");
```

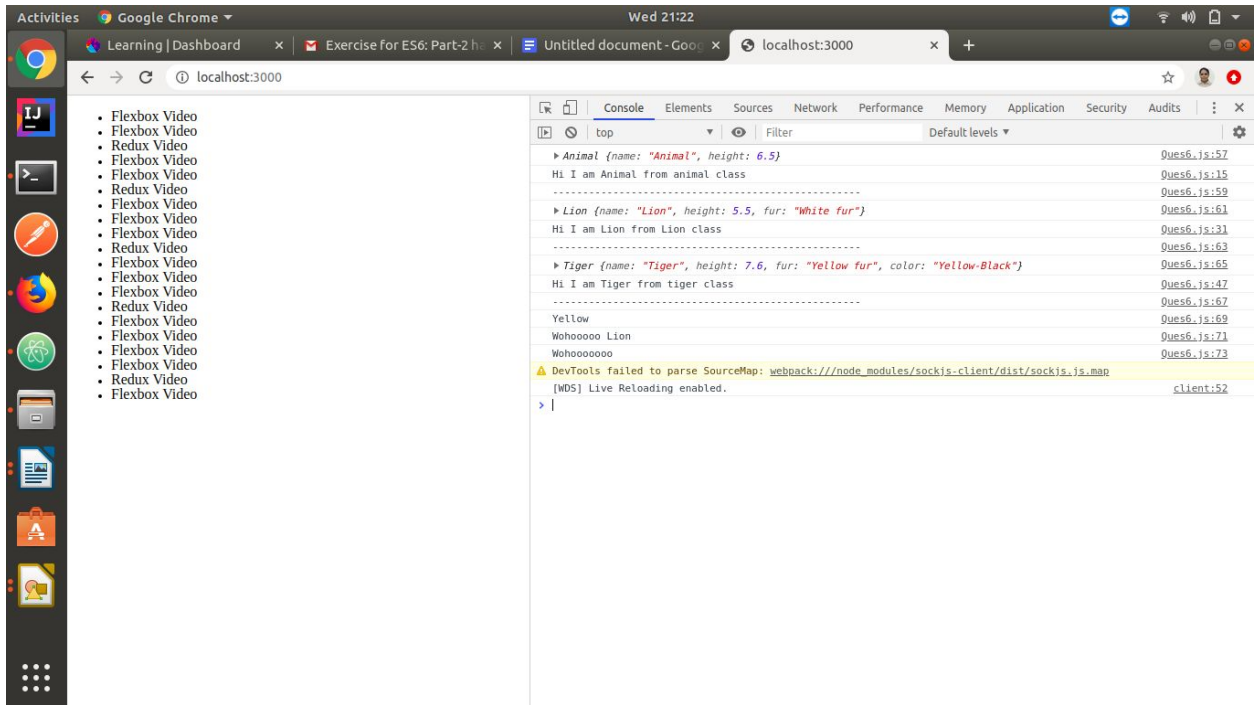
```
let lion = new Lion("Lion", 5.5, "White fur");  
console.log(lion);  
lion.hello();  
console.log("-----");
```

```
let tiger = new Tiger("Tiger", 7.6, "Yellow fur", "Yellow-Black");  
console.log(tiger);  
tiger.hello();  
console.log("-----");
```

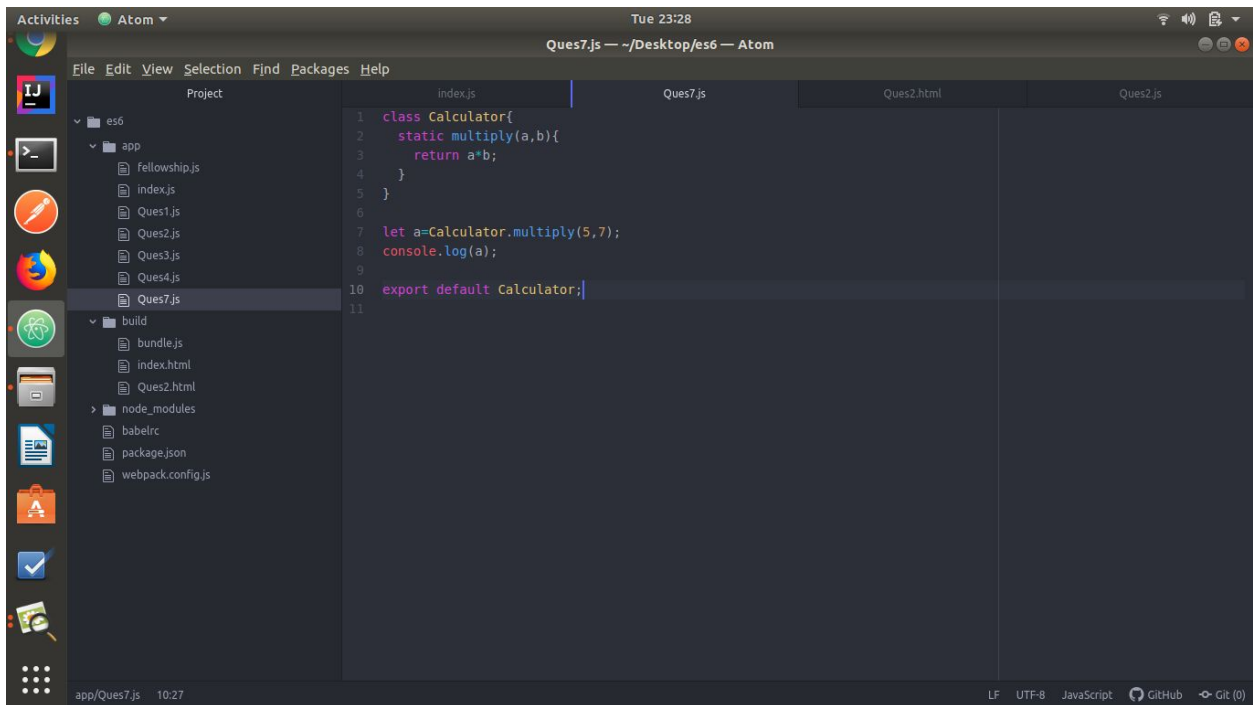
```
let staticMethod = Tiger.colour();  
console.log(staticMethod);  
let staticMethodLion = Lion.sound();  
console.log(staticMethodLion);  
let staticMethodAnimal = Animal.sound();  
console.log(staticMethodAnimal);
```

```
export {  
  king,  
  lion,  
  tiger,  
  staticMethod,  
  staticMethodLion,  
  staticMethodAnimal  
};
```

OUTPUT:



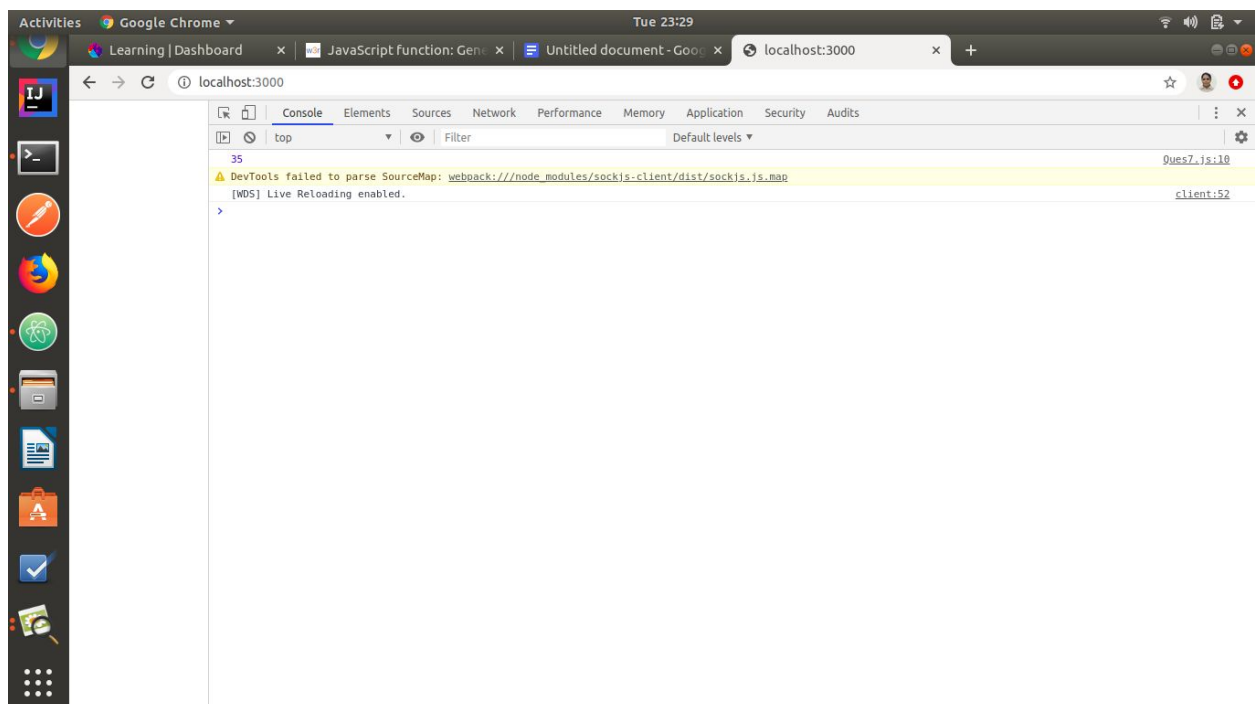
Q7. Write a program to implement a class having static functions.



The screenshot shows the Atom code editor with a project named 'es6' open. The file explorer on the left shows a directory structure with files like 'fellowship.js', 'index.js', 'Ques1.js', 'Ques2.js', 'Ques3.js', 'Ques4.js', and 'Ques7.js'. The 'Ques7.js' file is selected and its content is displayed in the editor. The code defines a 'Calculator' class with a static 'multiply' method and uses it in a script.

```
1 class Calculator{
2   static multiply(a,b){
3     return a*b;
4   }
5 }
6
7 let a=Calculator.multiply(5,7);
8 console.log(a);
9
10 export default Calculator;
11
```

OUTPUT:



Q8. Import a module containing the constants and method for calculating area of circle, rectangle, cylinder.

```
const areaCircle=(radius)=>{  
  const pi=3.14;  
  let area=pi*(radius*radius);  
  return `area of circle with radius ${radius} is ${area}`  
}
```

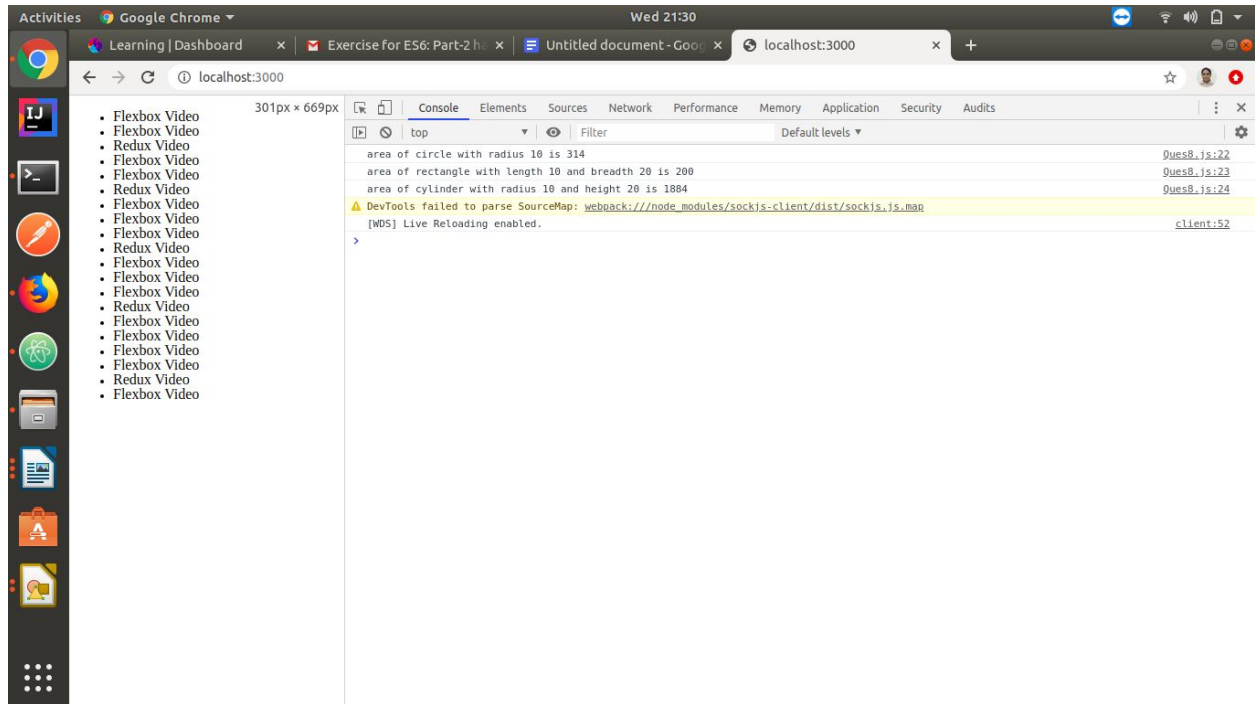
```
const areaRectangle=(length,breadth)=>{  
  return `area of rectangle with length ${length} and breadth ${breadth} is ${length*breadth}`  
}
```

```
const areaCylinder=(radius,height)=>{  
  const pi=3.14;  
  let area=(2*pi*radius*height)+(2*pi*radius*radius);  
  return `area of cylinder with radius ${radius} and height ${height} is ${area}`  
}
```

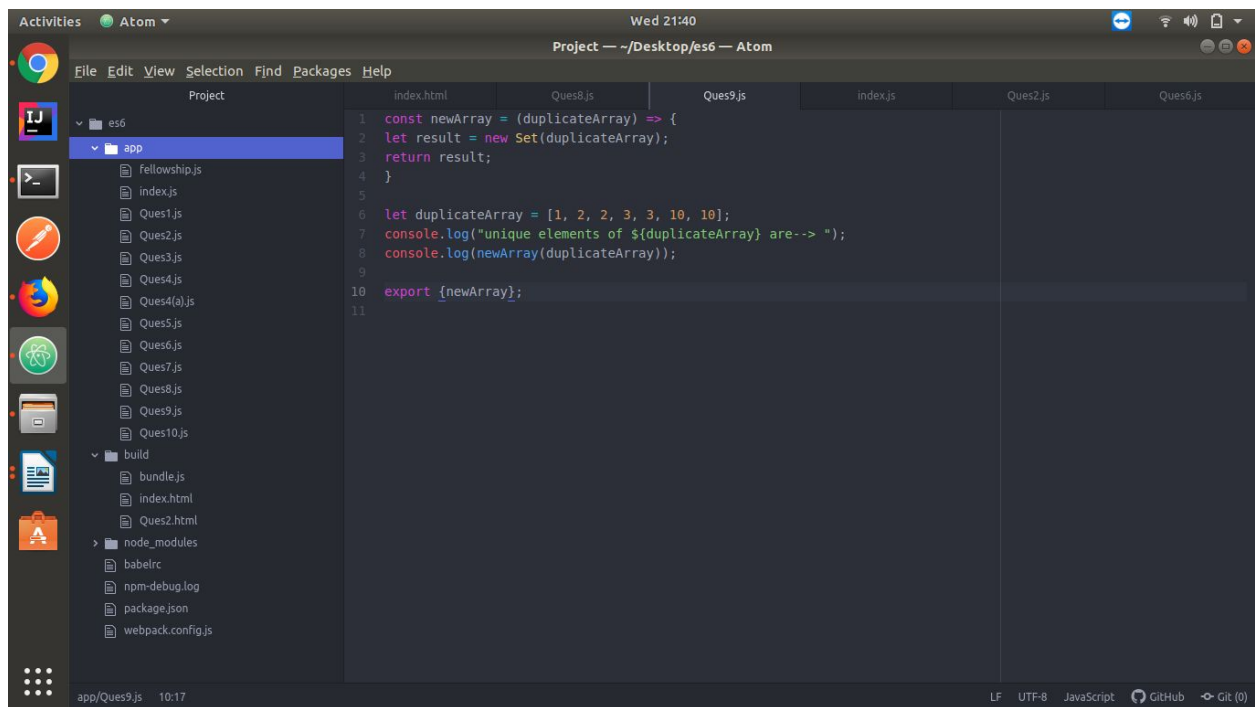
```
export{areaCircle,areaCylinder,areaRectangle};
```

```
console.log(areaCircle(10));  
console.log(areaRectangle(10, 20));  
console.log(areaCylinder(10, 20));
```

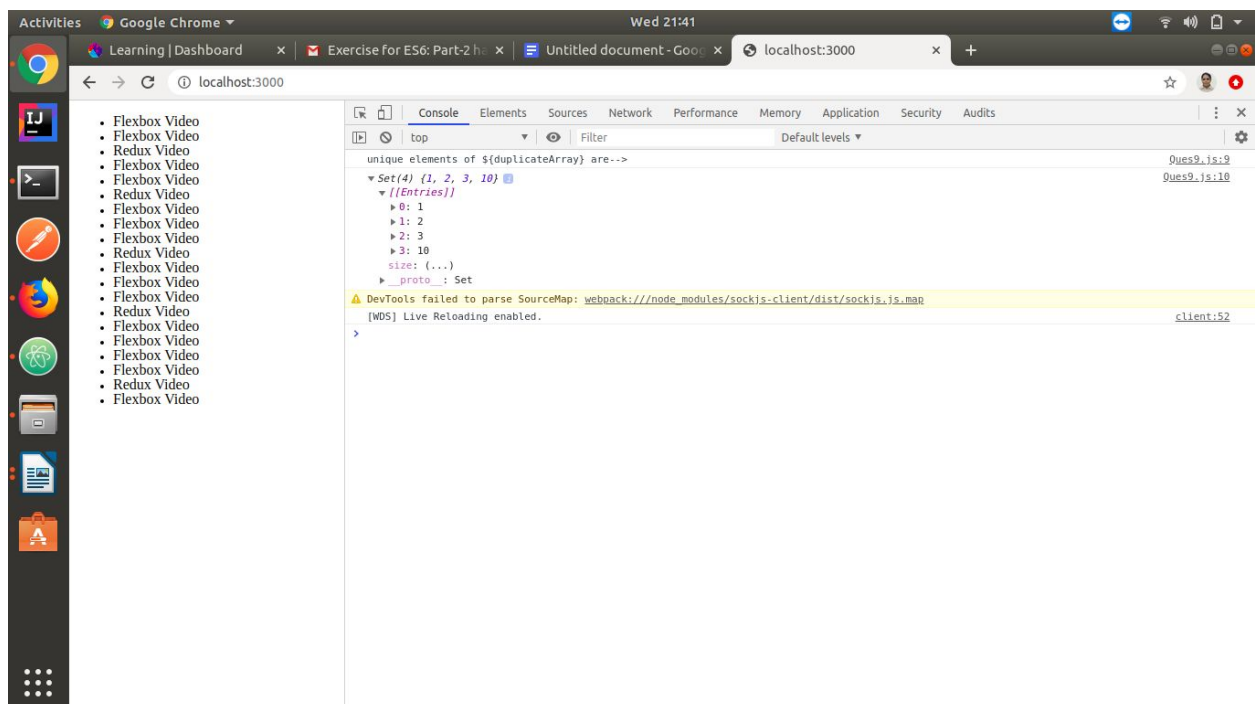
OUTPUT:



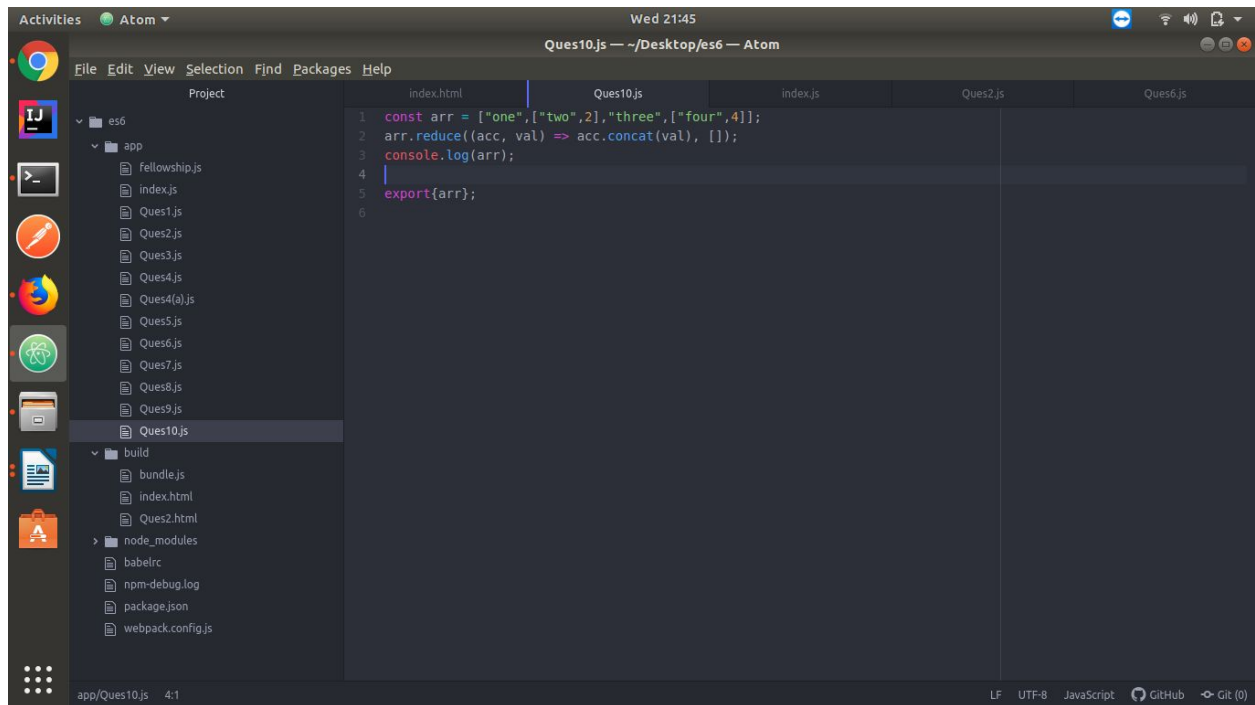
Q9. Import a module for filtering unique elements in an array.



OUTPUT:

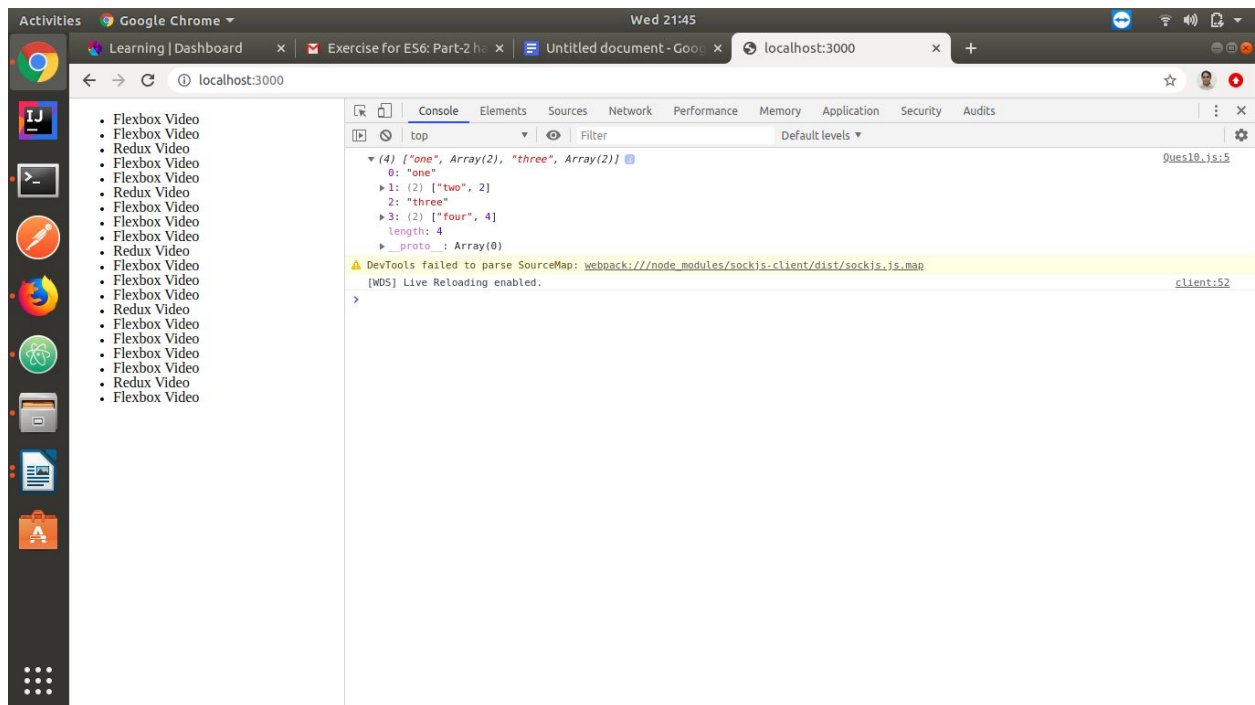


Q10. Write a program to flatten a nested array to single level using arrow functions.



The screenshot shows the Atom code editor with a project named 'es6' open. The file explorer on the left shows a directory structure with 'app' and 'build' folders. The 'app' folder contains files from 'fellowship.js' to 'Ques10.js'. The 'build' folder contains 'bundle.js', 'index.html', and 'Ques2.html'. The 'node_modules' folder contains 'babelrc', 'npm-debug.log', 'package.json', and 'webpack.config.js'. The main editor window shows 'Ques10.js' with the following code:

```
1 const arr = ["one",["two",2],"three",["four",4]];
2 arr.reduce((acc, val) => acc.concat(val), []);
3 console.log(arr);
4
5 export{arr};
6
```



The screenshot shows the Google Chrome browser with the address bar set to 'localhost:3000'. The console shows the output of the JavaScript code from 'Ques10.js:5':

```
(4) ["one", Array(2), "three", Array(2)]
  0: "one"
  1: (2) ["two", 2]
  2: "three"
  3: (2) ["four", 4]
  length: 4
  __proto__: Array(0)
```

Below the output, there is a warning message: 'DevTools failed to parse SourceMap: webpack:///node_modules/sockjs-client/dist/sockjs.js.map'. At the bottom, it says '[WDS] Live Reloading enabled.'

Q11. Implement a singly linked list in es6 and implement addFirst() addLast(), length(), getFirst(), getLast(). (without using array)

```
class Node {  
  constructor(data, next = null) {  
    this.data = data;  
    this.next = next;  
  }  
}
```

```
class LinkedList {  
  constructor() {  
    this.head = null;  
  }
```

```
  addFirst(data) {  
    let newNode = new Node(data);  
    //The pointer next is assigned head pointer so that both pointers now point at the same node.  
    newNode.next = this.head;  
    //As we are inserting at the beginning the head pointer needs to now point at the newNode.  
    this.head = newNode;  
  }
```

```
  addLast(data) {  
    let newNode = new Node(data);  
    // When head = null i.e. the list is empty, then head itself will point to the newNode.  
    if (!this.head) {  
      this.head = newNode;  
    }
```

// Else, traverse the list to find the tail (the tail node will initially be pointing at null), and update the tail's next pointer.

```
let tail = this.head;

while (tail.next !== null) {

  tail = tail.next;

}

tail.next = newNode;

}
```

```
lengthOfList() {

  var count = 1;

  let tail = this.head;

  while (tail.next !== null) {

    tail = tail.next;

    count += 1;

  }

  return count;

}
```

```
getLast() {

  let tail = this.head;

  while (tail.next !== null)

    tail = tail.next;

  return tail.data;

}
```

```
getFirst() {
```

```
return this.head.data;

}

}
```

```
let list=new LinkedList();

list.addFirst(5);

list.addFirst(2);

list.addFirst(3);

list.addLast(6);

console.log("Linked List :-> ",list);

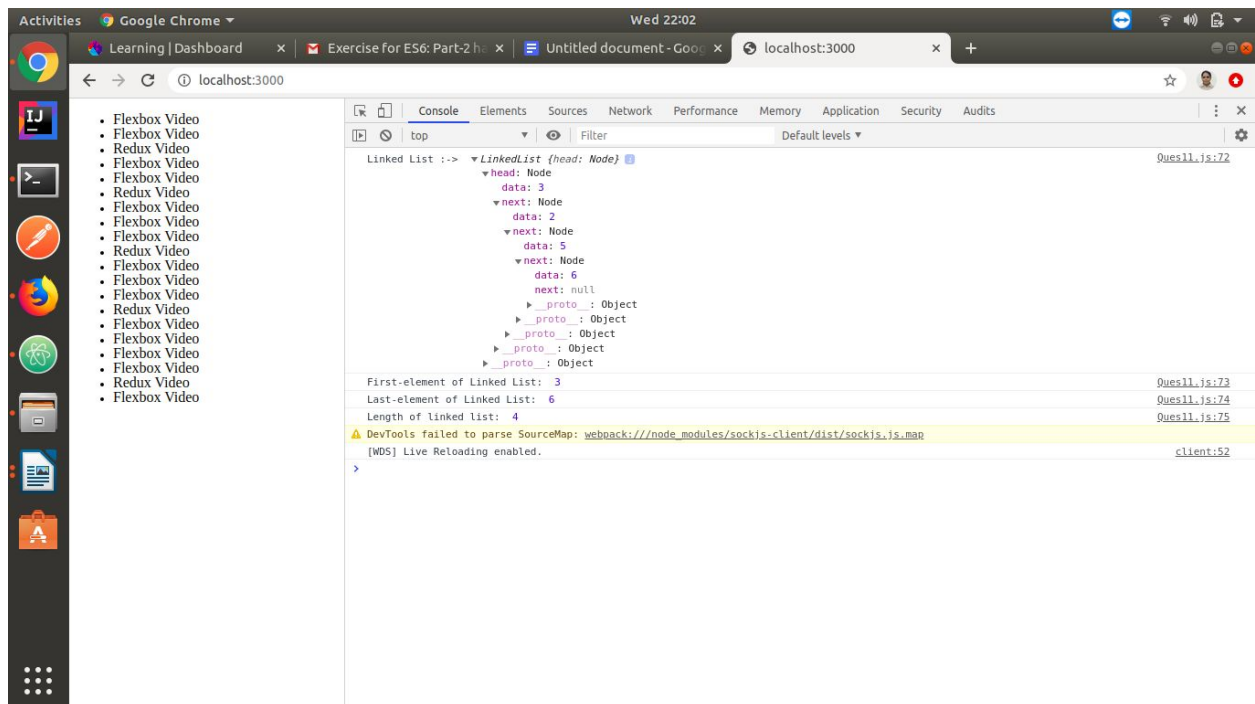
console.log("First-element of Linked List: ",list.getFirst());

console.log("Last-element of Linked List: ",list.getLast());

console.log("Length of linked list: ",list.lengthOfList());

export {LinkedList}
```

OUTPUT:



Q12. Implement Map and Set using Es6

```
// Q12. Implement Map using Es6
const arrayToMap = (array) => {
  let valMap = new Map(array);
  console.log(valMap);
  console.log("Iterating through map : ")
  for (let [key, value] of valMap.entries()) {
    console.log(`${key} points to ${value}`);
  }
}
```

```
console.log("Implementing Map: ");
let array1=[[1,"one"],[2,"two"],[3,"three"]];
arrayToMap(array1);
```

```
export {arrayToMap};
```

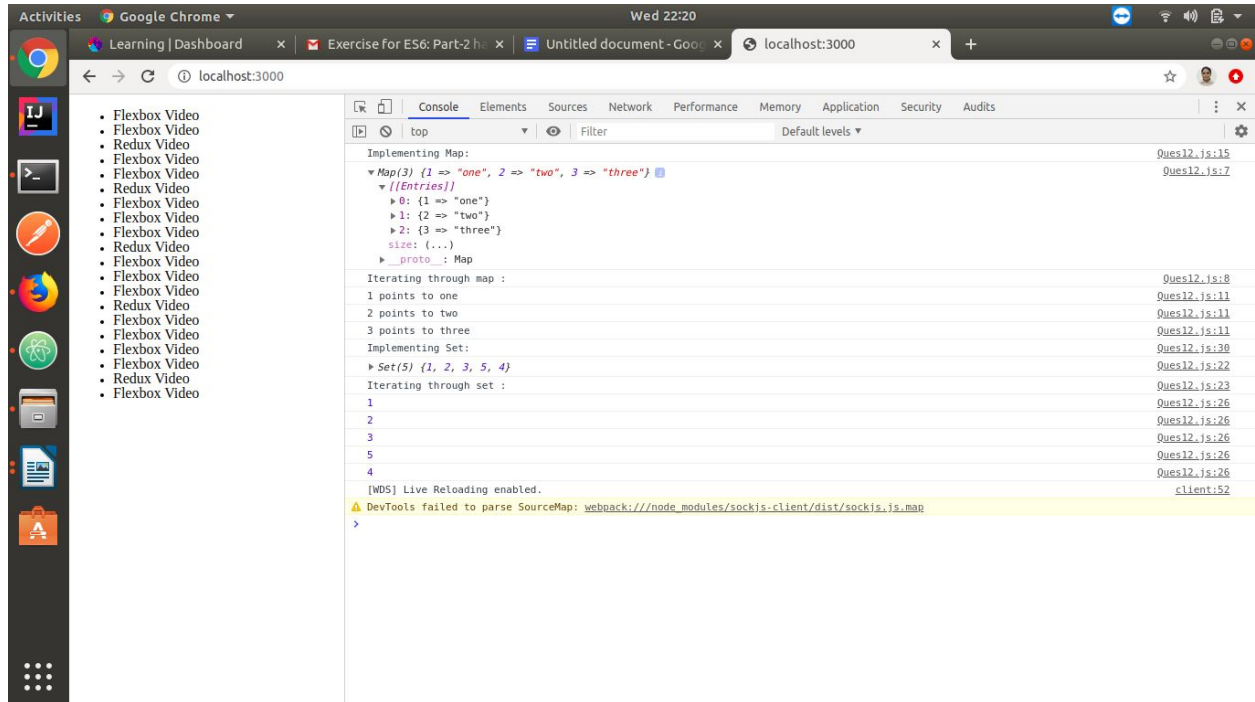
```
//Q12. Implement Set using Es6
```

```
const arrayToSet = (array) => {
  let valSet = new Set(array);
  console.log(valSet);
  console.log("Iterating through set : ")
  for (let elements of valSet.values()) {
    console.log(elements);
  }
}
```

```
console.log("Implementing Set: ");
let array2=[1,2,3,5,4,1];
arrayToSet(array2);
```

```
export {arrayToSet};
```

OUTPUT:



Q13. Implementation of stack (using linked list)

```
function stackUsingLL() {
```

```
    let Node = function (elm) {  
        this.element = elm;  
        this.next = null;  
    }
```

```
    //To keep track of the size
```

```
    let length = 0;
```

```
    //To keep track of the list
```

```
    let head = null;
```

```
    //Push data in the stack
```

```
    this.push = function (elm) {
```

```
        //Create a new node
```

```
        let node = new Node(elm);
```

```
        var current;
```

```
        //Add the new node at the top
```

```
        current = head;
```

```
        node.next = current;
```

```
        head = node;
```

```
        length++;
```

```
    }
```

```
    //Pop the item from the stack
```

```
this.pop = function () {  
  let current = head;  
  //If there is item then remove it  
  //and make the next element as the first  
  if (current) {  
    let elm = current.element;  
    current = current.next;  
    head = current;  
    length--;  
    return elm;  
  }  
  return null;  
}
```

//Return the first element in the stack

```
this.peek = function () {  
  if (head) {  
    return head.element;  
  }  
  return null;  
}
```

//Convert the stack to an array

```
this.toArray = function () {  
  let arr = [];  
  let current = head;
```

```
while (current) {  
  arr.push(current.element);  
  current = current.next;  
}  
return arr;  
}
```

```
//Check if stack is empty  
this.isEmpty = function () {  
  return length === 0;  
}
```

```
//Return the size of the stack  
this.size = function () {  
  return length;  
}
```

```
//Clear the stack  
this.clear = function () {  
  head = null; length = 0;  
}  
}
```



```

let stack = new stackUsingLL();

stack.push(10);

stack.push(20);

stack.push(30);

console.log("Stack : ",stack.toArray());

console.log("top element of stack",stack.peek());

console.log("is stack empty",stack.isEmpty());

console.log("stack size-",stack.size());

console.log("pop function->",stack.pop());

console.log("stack after pop",stack.toArray());

console.log("stack size-",stack.size());

console.log("clearing the stack");

stack.clear(); //clear the stack

console.log("is stack empty",stack.isEmpty());

export{stackUsingLL}

```

OUTPUT:

