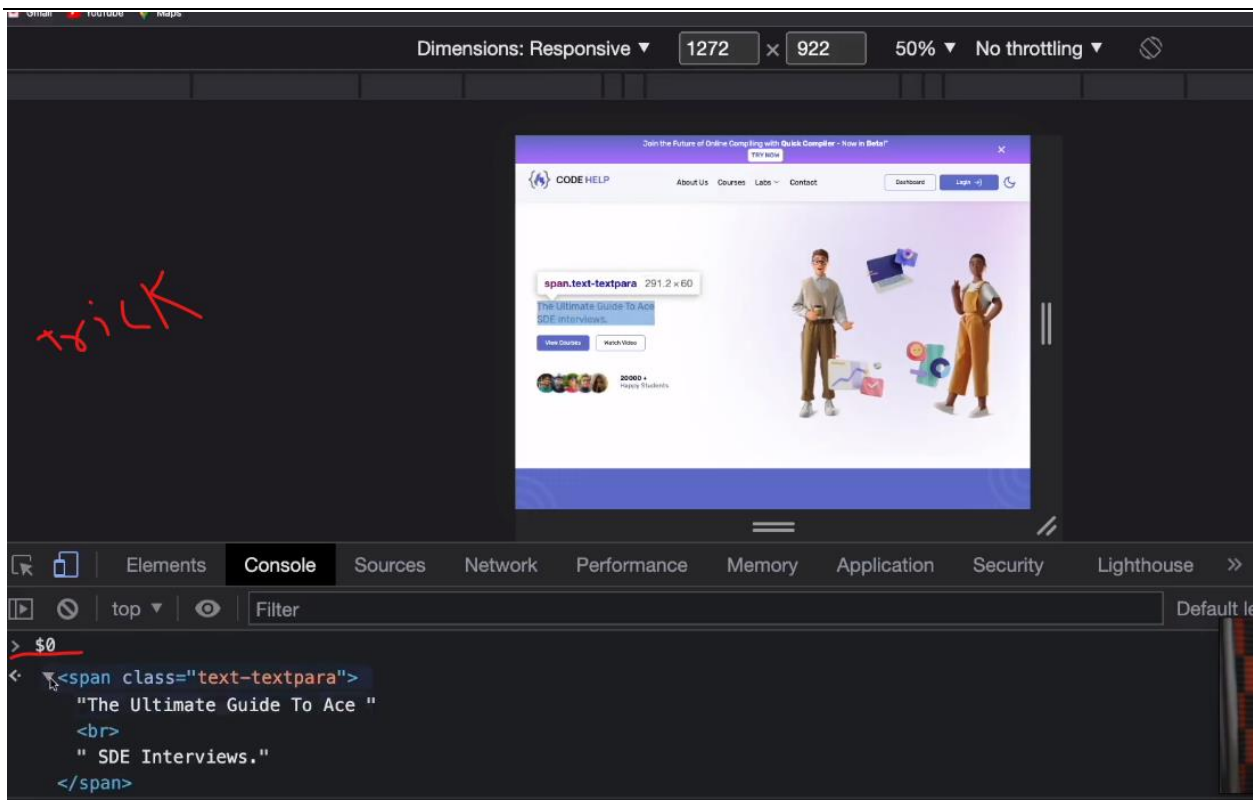
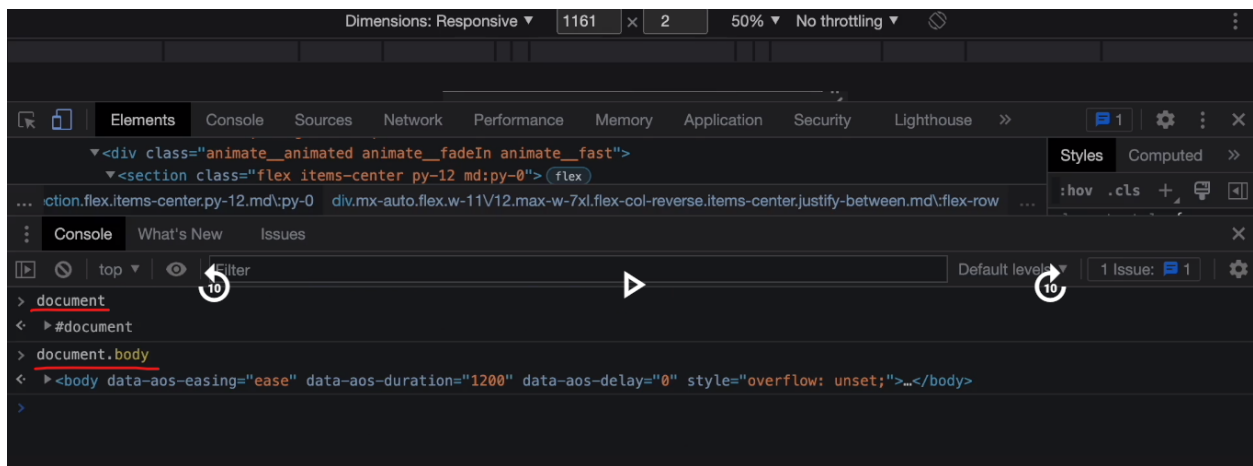
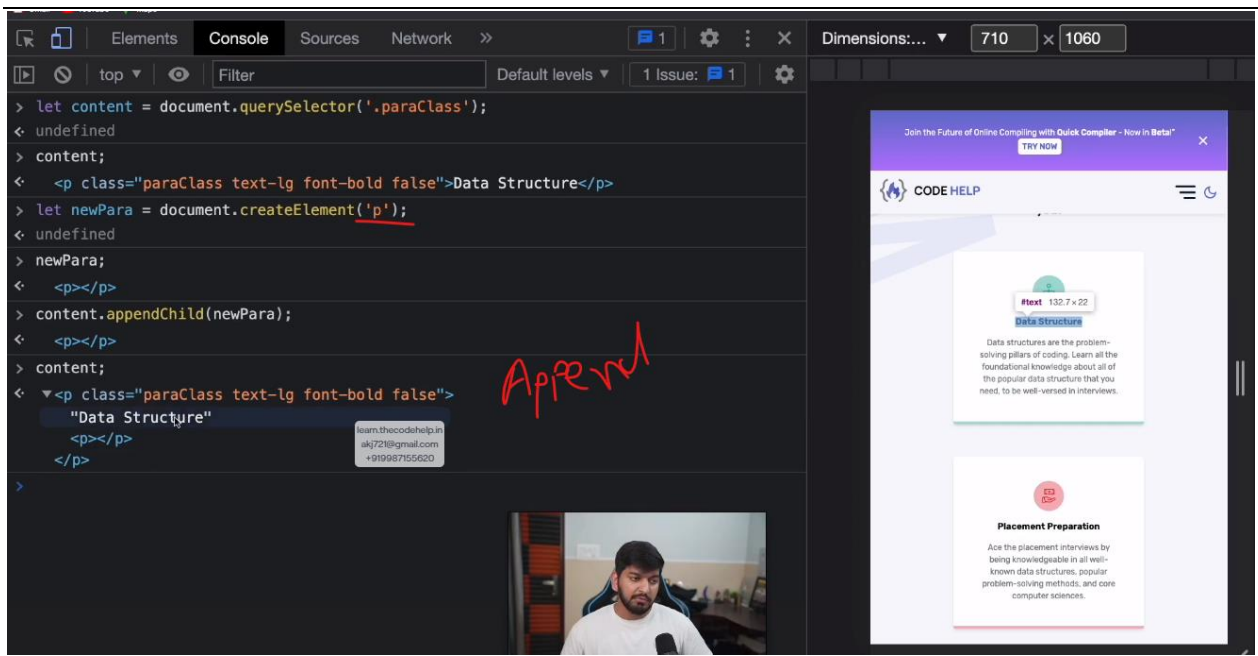
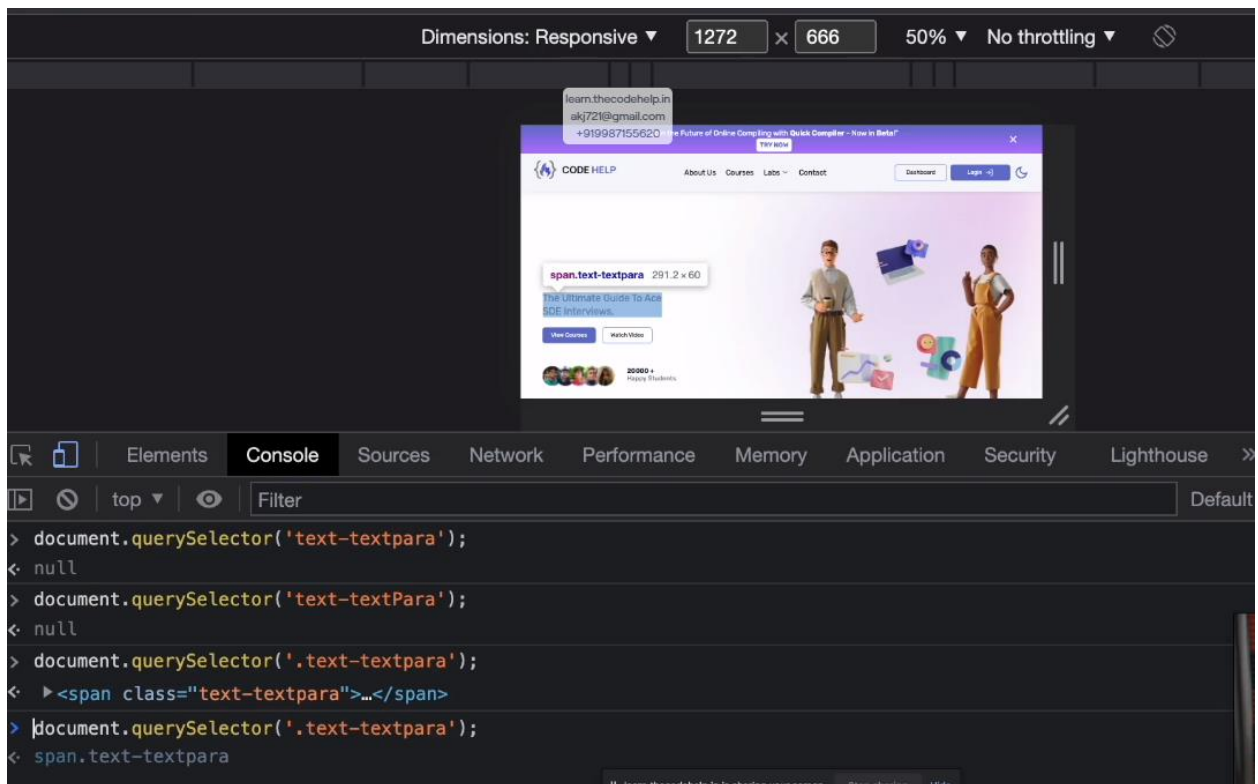


## Class - 1

### DOM :Document Object Model





## insertAdjacentElement

```
> let content = $0;
< undefined
> content;
< <h2 class="chakra-heading css-1q0yyim">Accept Payments with Razorpay Payment Suite</h2>
> let textToAdd = '<h3> abcd </h3>';
< undefined
> content.insertAdjacentElement('beforeBegin', textToAdd);
* Uncaught TypeError: Failed to execute 'insertAdjacentElement' on 'Element': parameter 2 is not of type 'Element'.
  at <anonymous>:1:9
> let newText = document.createElement('h3');
< undefined
> newText.textContent = 'ABCD';
< 'ABCD'
> content.insertAdjacentElement('beforeBegin', newText);
< <h3>ABCD</h3>
```

## RemoveElement

```
> parentElement;
< <h1 class="chakra-heading css-wob8a0">Power your finance, grow your business</h1>
> content;
< <h2 class="chakra-heading css-1q0yyim">
  "Accept Payments with Razorpay Payment Suite"
  <h3 class="tempText">ABCD</h3>
</h2>
> content.removeChild(childElement);
< <h3 class="tempText">ABCD</h3>
> content;
< <h2 class="chakra-heading css-1q0yyim">Accept Payments with Razorpay Payment Suite</h2>
```

## Css style using JS Dom

The screenshot shows a web browser's developer console with the following JavaScript code:

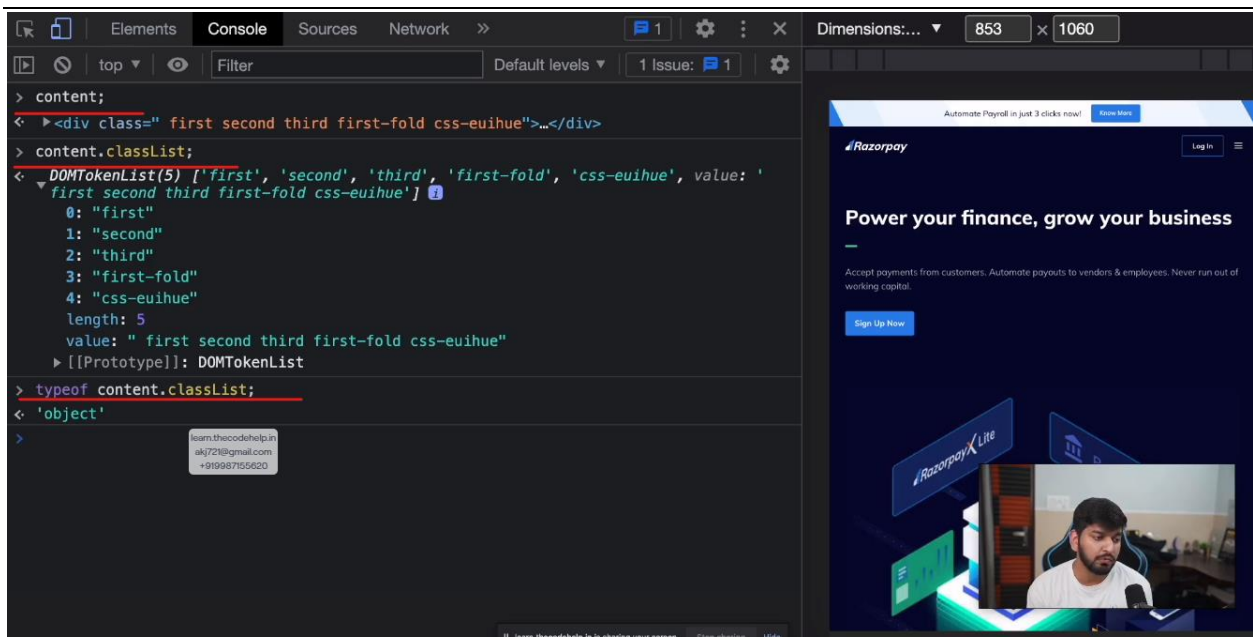
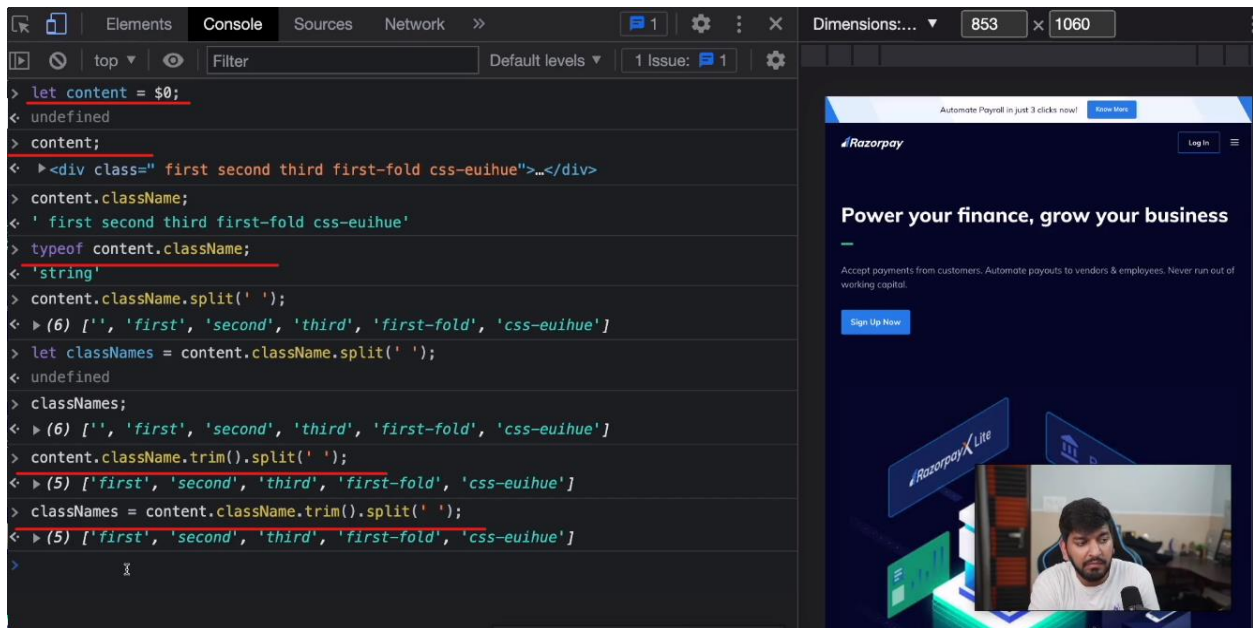
```
> let content = $0;
< undefined
> content;
< <h1>Element.remove()</h1>
> content.style.color = 'red';
< 'red'
> content.style.backgroundColor = 'white';
< 'white'
>
```

The right sidebar displays the MDN documentation for `Element.remove()`, including its syntax, parameters, and return value. A video thumbnail of a person is visible in the bottom right corner of the sidebar.

The screenshot shows a web browser's developer console with the following JavaScript code:

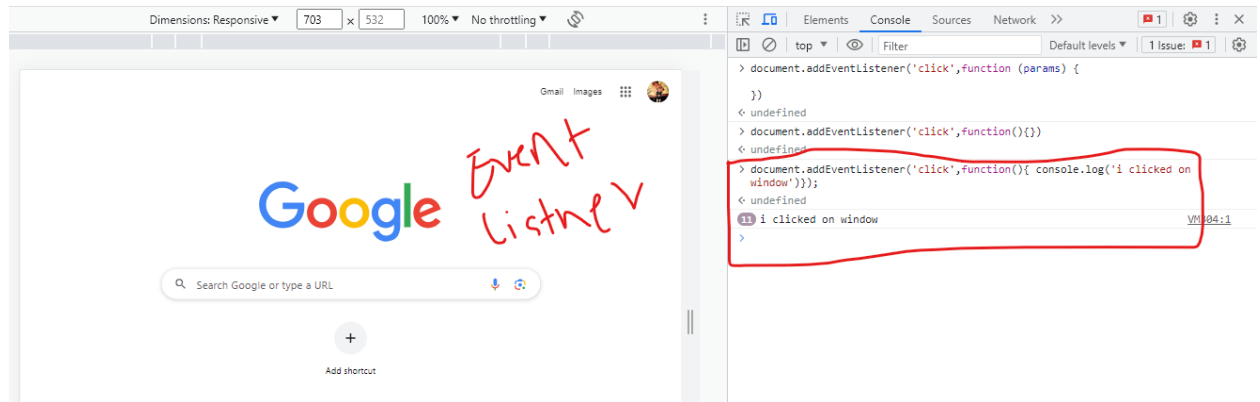
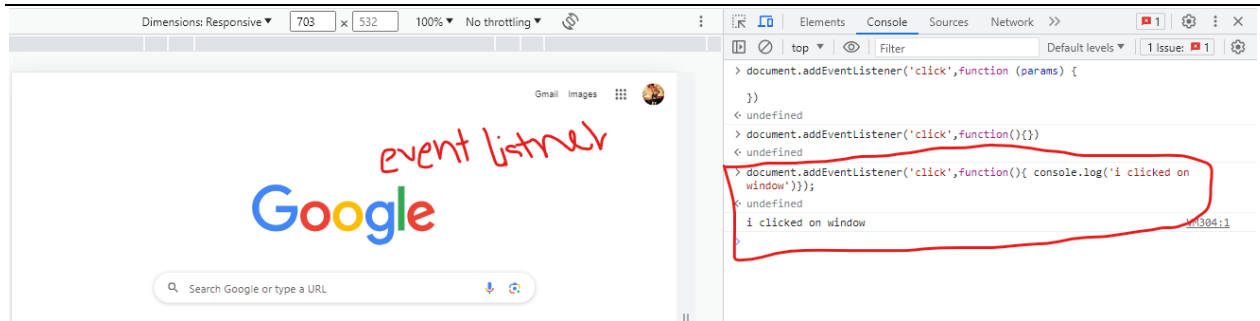
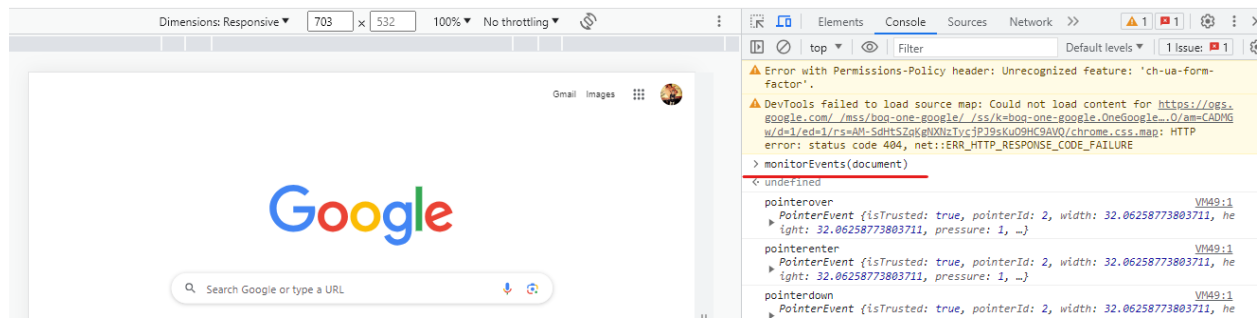
```
> let content = $0;
< undefined
> content;
< <h1>Element.remove()</h1>
> content.style.color = 'red';
< 'red'
> content.style.backgroundColor = 'white';
< 'white'
> content.style.cssText = 'color:green; background-color:yellow; font-size:4em;';
< 'color:green; background-color:yellow; font-size:4em;'
> content.setAttribute('style', 'color:white');
< undefined
> content.style.cssText = 'color:green; background-color:yellow; font-size:4em;';
< 'color:green; background-color:yellow; font-size:4em;'
> content.setAttribute('style', 'color:white');
< undefined
> content.setAttribute("style", "background-color:red;");
< undefined
> content.setAttribute("style", "color:orange;");
< undefined
>
```

The right sidebar displays the MDN documentation for `Element.remove()`, including its syntax, parameters, and return value. A video thumbnail of a person is visible in the bottom right corner of the sidebar.



## Class 2

### monitorEvents(document)





developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener

mdn web docs

References > addEventListener()

## EventTarget: addEventListener() method

The `addEventListener()` method of the `EventTarget` interface sets up a function that will be called whenever the specified event is delivered to the target.

Common targets are `Element`, or its children, `Document`, and `Window`, but the target may be any object that supports events (such as `XMLHttpRequest`).

**Note:** The `addEventListener()` method is the *recommended* way to register an event listener. The benefits are as follows:

- It allows adding more than one handler for an event. This is particularly useful for libraries, JavaScript modules, or any other

```
> let content=document.querySelector('h1');
< undefined
> content
< <h1>EventTarget: addEventListener() method</h1>
> POST https://developer.mozilla.org/pone/get placement-context.tsx:83
net::ERR_BLOCKED_BY_CLIENT
> content.addEventListener('click',function()
{content.style.background='green'});
< undefined
>
```

EventListener :

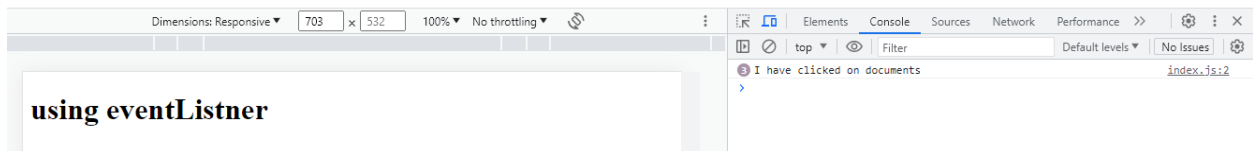
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>using eventListener</h1>
  <script src="/fun1/index.js"></script>
</body>
</html>
```

Index.js

```
document.addEventListener('click',function(){
  console.log('I have clicked on documents');
});
```

Output :



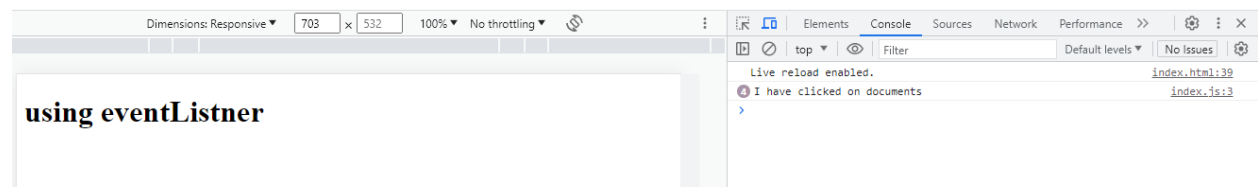
Index.js

```
function clickEventListener1(){
  console.log('I have clicked on documents ');
}

document.addEventListener('click',clickEventListener1);
```



output :



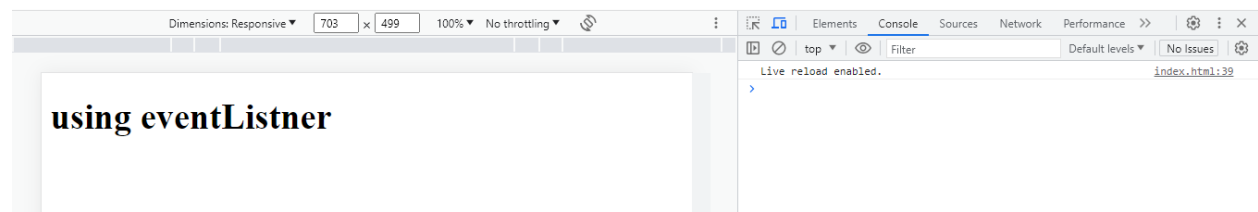
## RemoveEventListener

```
function clickEventListener1(){
    console.log('I have clicked on documents ');
}

document.addEventListener('click',clickEventListener1);

//apply remove event listner

document.removeEventListener('click',clickEventListener1);
```



```
1 document.addEventListener('click', function () {
2   console.log('I have clicked on documents ');
3 });
4
5
6
7
8 document.removeEventListener('click', function () {
9   console.log('I have clicked on documents ');
10 });
11
```

function in different object  
arjit

using eventListner

I have clicked on documents index.js:14

## EventObject

Index.html

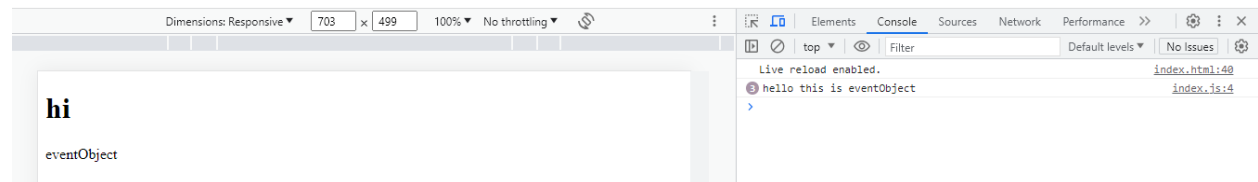
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>hi</h1>
  <p id="p1">eventObject</p>
  <script src="/fun2/index.js"></script>
</body>
</html>
```

Index.js

```
const content=document.querySelector('#p1');

content.addEventListener('click',function(event){
  console.log('hello this is eventObject');
});
```

Output

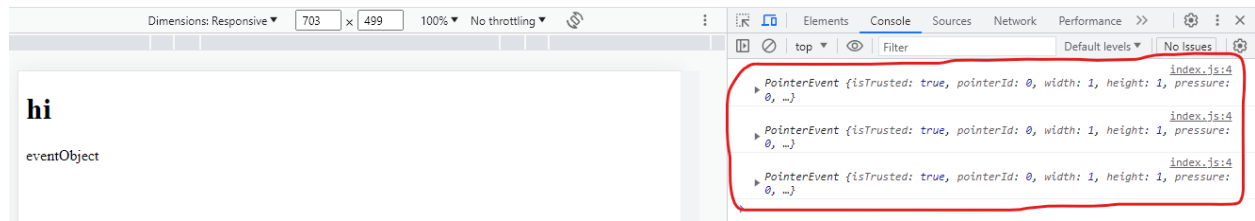


Index.js

```
const content=document.querySelector('#p1');

content.addEventListener('click',function(event){
  console.log(event);
});
```

Output



## PreventDefault

Index.html

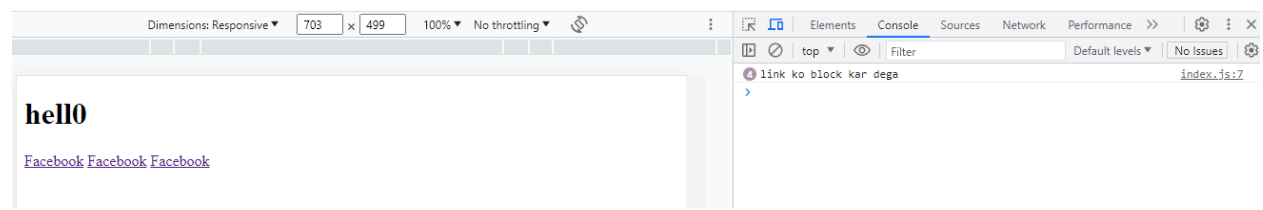
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>hell0</h1>
  <a href="https://www.facebook.com">Facebook</a>
  <a href="https://www.facebook.com">Facebook</a>
  <a href="https://www.facebook.com">Facebook</a>
  <script src="/fun3/index.js"></script>
</body>
</html>
```

Index.js

```
let links=document.querySelectorAll('a');

let thirdLink=links[2];

thirdLink.addEventListener('click',function(event){
  event.preventDefault();
  console.log('link ko block kar dega');
});
```



## Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>Hi</h1>
  <script src="/fun4/index.js"></script>
</body>
</html>
```

## Index.js

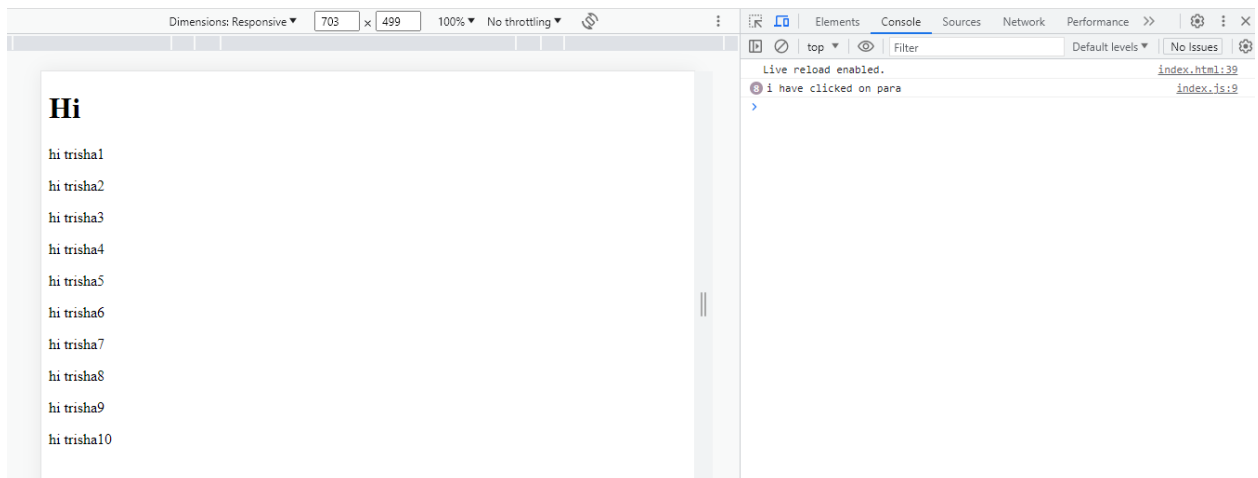
```
let myDiv=document.createElement('div');

for(let i=1;i<=10;i++){
  let paraElement=document.createElement('p');

  paraElement.textContent='hi trisha'+i;

  paraElement.addEventListener('click',function(event){
    console.log('i have clicked on para');
  });
  myDiv.appendChild(paraElement);
}

document.body.appendChild(myDiv);
```



## Event.target.textContent

Index.js

```
let myDiv=document.createElement('div');

function paraStatus(event){
    console.log('para'+event.target.textContent);
}

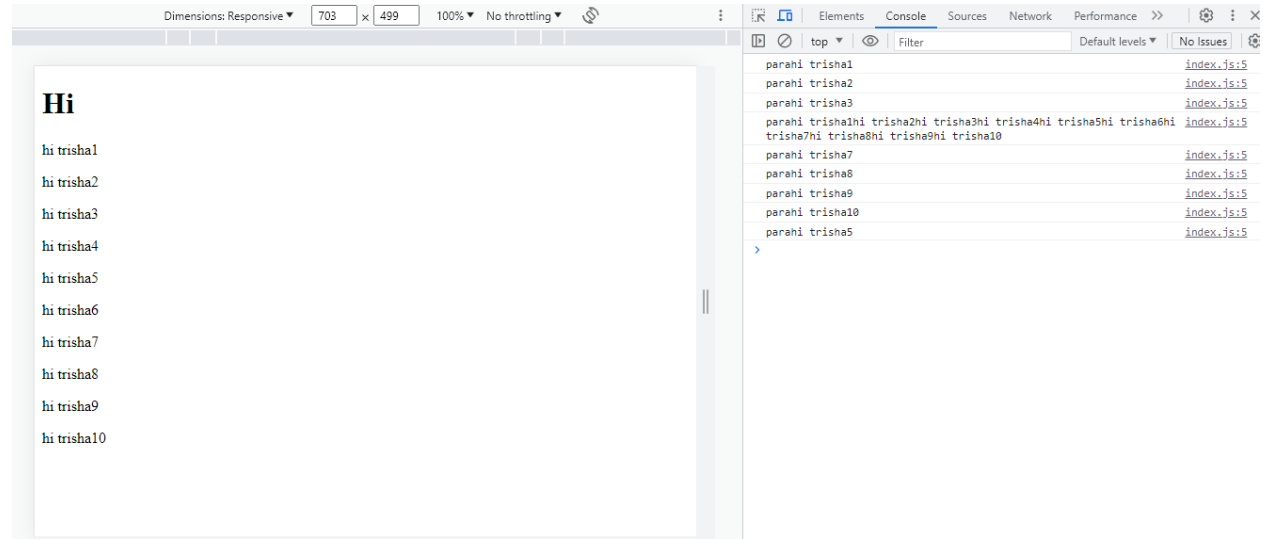
myDiv.addEventListener('click',paraStatus);

for(let i=1;i<=10;i++){
    let paraElement=document.createElement('p');

    paraElement.textContent='hi trisha'+i;

    myDiv.appendChild(paraElement);
}

document.body.appendChild(myDiv);
```





Only select element pe work karega (`event.target.nodeName`)

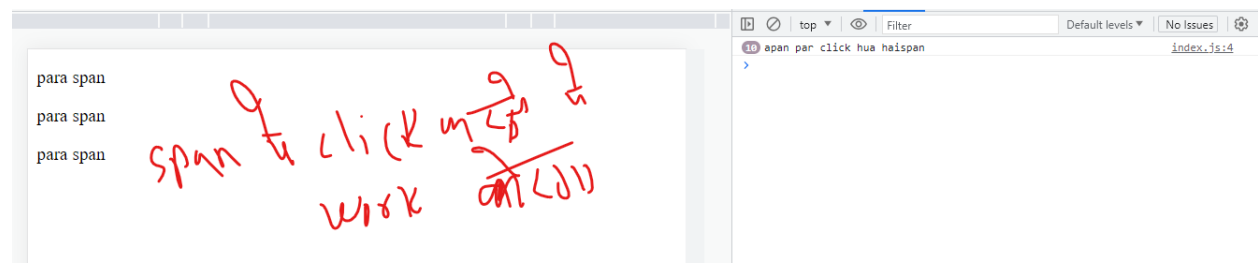
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

  <article id="wrapper">
    <p>para <span>span</span></p>
    <p>para <span>span</span></p>
    <p>para <span>span</span></p>
  </article>
  <script src="/fun5/index.js"></script>
</body>
</html>
```

Index.js

```
let element=document.querySelector('#wrapper');
element.addEventListener('click',function(event){
  if(event.target.nodeName==='SPAN'){
    console.log('apan par click hua hai' +event.target.textContent);
  }
});
```



## DOM + Modern JS –Class 3

performance.now()

```
1 //adding 100para
2 const t1 = performance.now();
3 for(let i=1; i<=100; i++) {
4   let newElement = document.createElement('p');
5   newElement.textContent = 'This is Para ' + i;
6   document.body.appendChild(newElement);
7 }
8
9 const t2 = performance.now();
10 console.log("this took " + (t2-t1) + " ms");

12 //optimising a bit
13 const t3 = performance.now();
14 let myDiv = document.createElement('div');
15
16 for(let i=1; i<=100; i++) {
17   let element = document.createElement('p');
18   element.textContent = 'This is Para ' + i;
19   myDiv.appendChild(element);
20 }
21
22
23 document.body.appendChild(myDiv);
24 const t4 = performance.now();
25 console.log("this took " + (t4-t3) + " ms");
```

Press Esc to exit full screen

Elements Console Sources Network Performance Memory Application

Filter

this took 0.4000009536743164 ms

this took 0.1000002384185791 ms

index.js:10

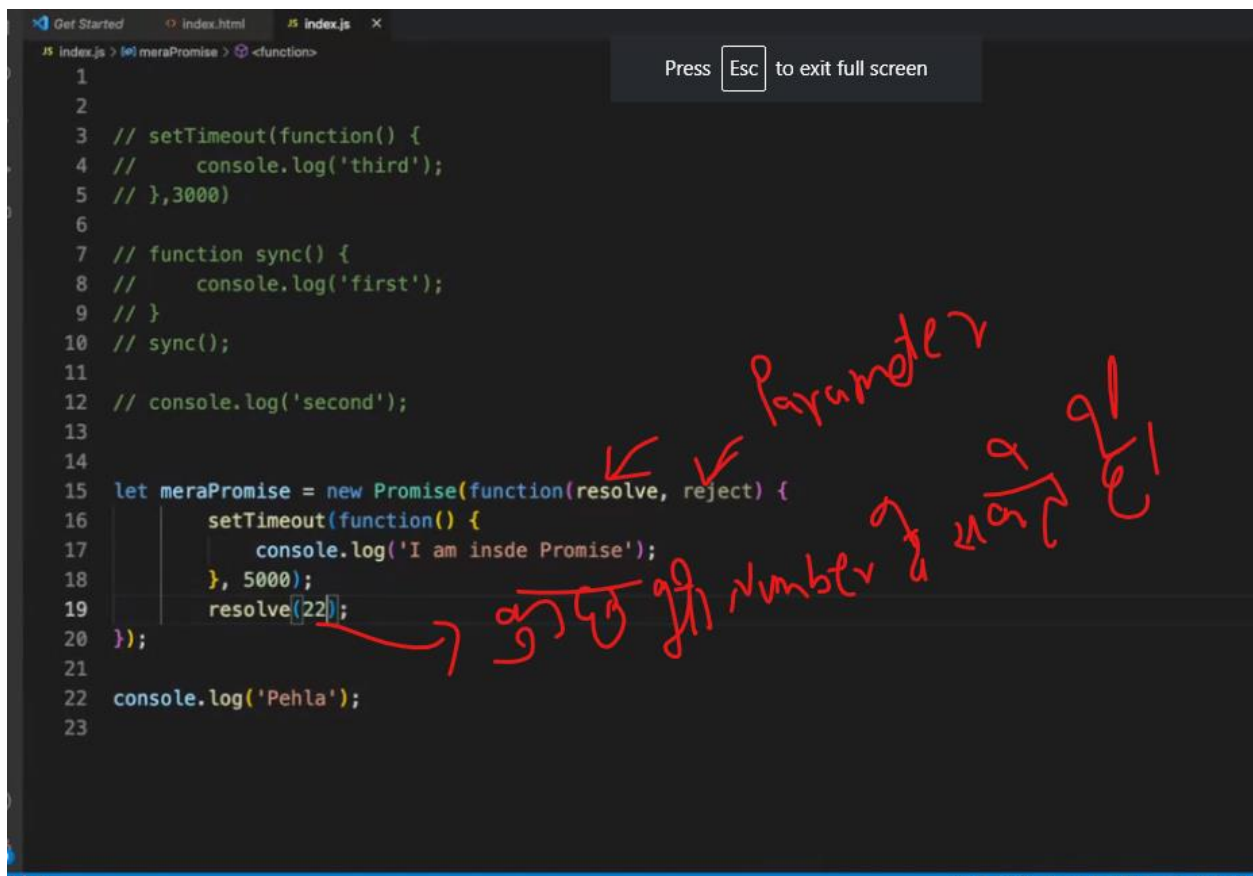
index.js:25

## Document fragment

```
28 let fragment = document.createDocumentFragment();
29 for(let i=1; i<=100; i++) {
30     let newElement = document.createElement('p');
31     newElement.textContent = 'This is Para ' + i;
32
33     fragment.appendChild(newElement);
34 }
35 document.body.appendChild(fragment); // 1 Reflow, 1 Repaint
```

## DOM + Modern JS -Class 4

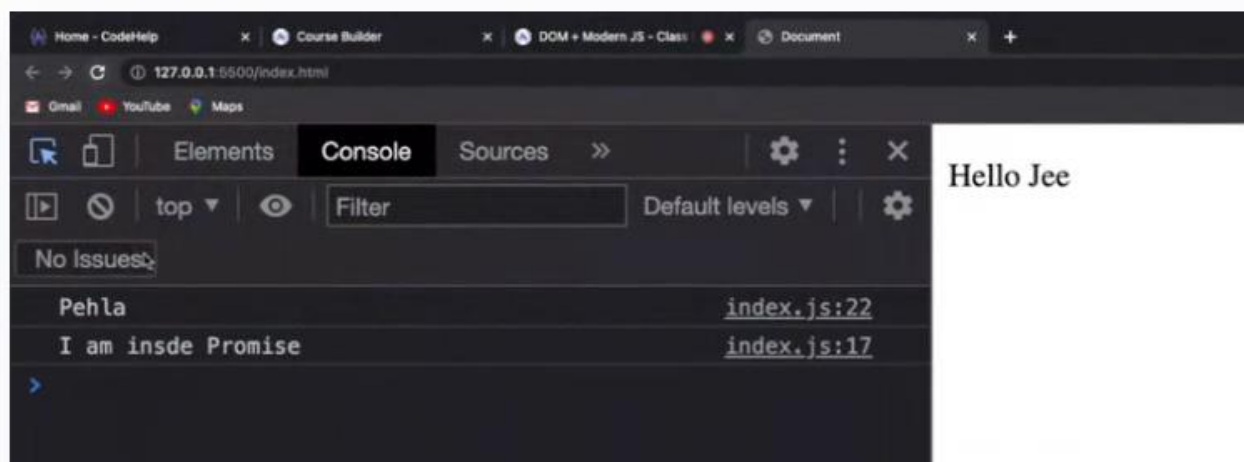
promise



```
1
2
3 // setTimeout(function() {
4 //   console.log('third');
5 // },3000)
6
7 // function sync() {
8 //   console.log('first');
9 // }
10 // sync();
11
12 // console.log('second');
13
14
15 let meraPromise = new Promise(function(resolve, reject) {
16   setTimeout(function() {
17     console.log('I am insde Promise');
18   }, 5000);
19   resolve(22);
20 });
21
22 console.log('Pehla');
23
```

Handwritten notes in Hindi:

- Parameter (pointing to resolve and reject)
- 22 number ka value (pointing to 22)



## Promise-resolve

Index.html

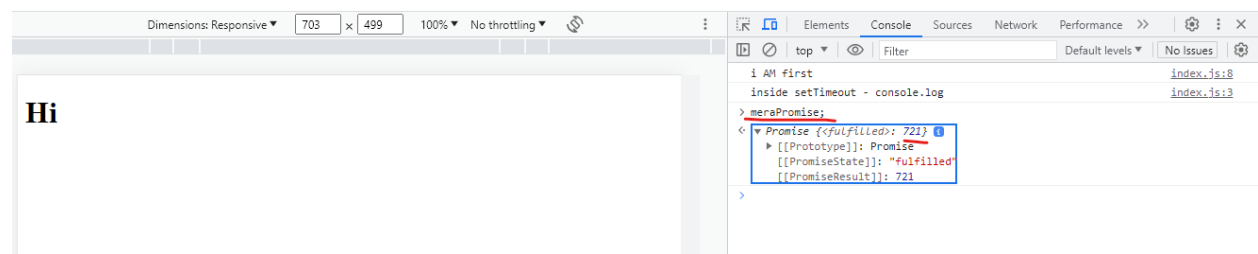
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>Hi</h1>
  <script src="/fun1/index.js"> </script>
</body>
</html>
```

Index.js

```
let meraPromise=new Promise(function(resolve,reject){
  setTimeout(function(){
    console.log('inside setTimeout - console.log');
  },5000);
  resolve(721);
});

console.log('i AM first');
```

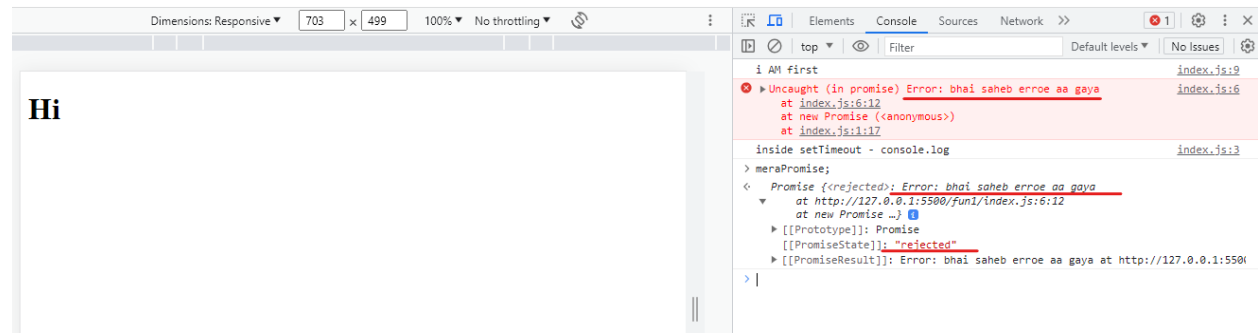
output



## Promise- reject

```
let meraPromise=new Promise(function(resolve,reject){
  setTimeout(function(){
    console.log('inside setTimeout - console.log');
  },5000);
  //resolve(721);
  reject(new Error('bhai saheb erroe aa gaya'));
});

console.log('i AM first');
```



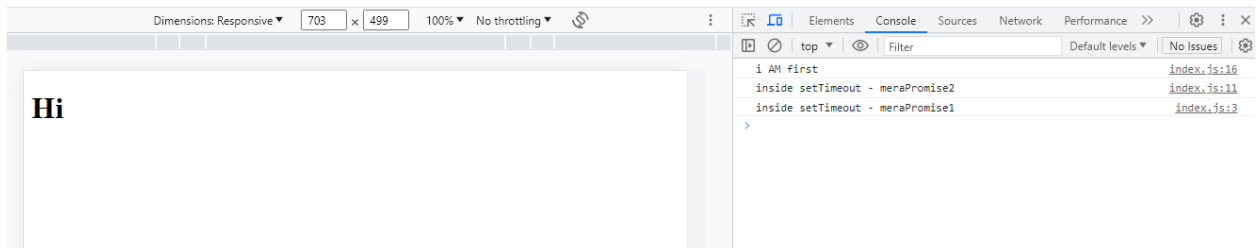
## Parallely run promise

Index.js

```
let meraPromise1=new Promise(function(resolve,reject){
  setTimeout(function(){
    console.log('inside setTimeout - meraPromise1');
  },5000);
});

let meraPromise2=new Promise(function(resolve,reject){
  setTimeout(function(){
    console.log('inside setTimeout - meraPromise2');
  },3000);
});

console.log('i AM first');
```



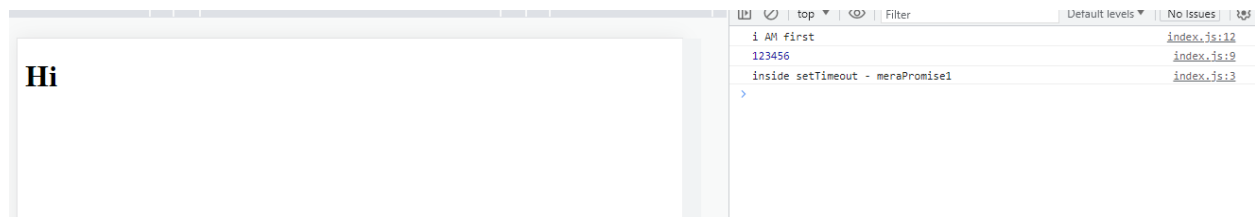


## Promise-then()

```
let meraPromise1=new Promise(function(resolve,reject){
  setTimeout(function(){
    console.log('inside setTimeout - meraPromise1');
  },5000);
  resolve(123456);
});

meraPromise1.then(function(val){
  console.log(val);
});

console.log('i AM first');
```



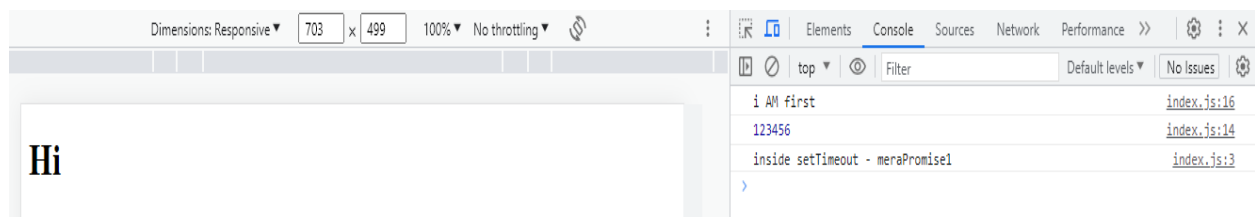
```
let meraPromise1=new Promise(function(resolve,reject){
  setTimeout(function(){
    console.log('inside setTimeout - meraPromise1');
  },5000);
  resolve(123456);
});

// meraPromise1.then(function(val){
//   console.log(val);
// });

// make arrow function

meraPromise1.then((val)=>console.log(val) );

console.log('i AM first');
```

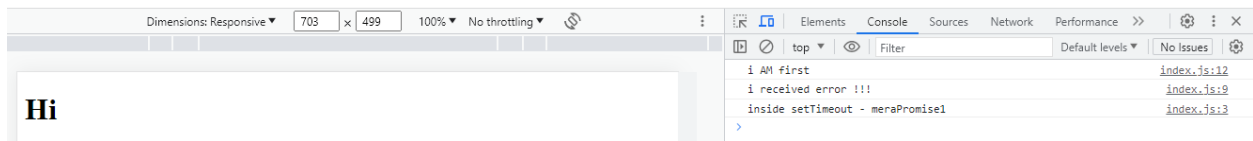


## Promise-catch()

```
let meraPromise1=new Promise(function(resolve,reject){
  setTimeout(function(){
    console.log('inside setTimeout - meraPromise1');
  },5000);
  reject(new Error('bhai saheb erroe aa gaya !!!'));
});

meraPromise1.catch(function(val){
  console.log('i received error !!!');
});

console.log('i AM first');
```




```
let p=new Promise(function(resolve,reject){
  setTimeout(() => {
    console.log("hi i am setTimeout function ,p")
  }, 5000);
  reject(new Error("bhai saheb erroe aa gaya !!!"));
});

p.then(function(val){
  console.log(val);
},function(error){
  console.log('i received error!!!')
});

console.log('i AM first');
```

output → i AM first  
i received error!!  
hi i am setTimeout function ,p

same code



```

let meraPromise1=new Promise(function(resolve,reject){
  setTimeout(function(){
    console.log('inside setTimeout - meraPromise1');
  },5000);
  reject(new Error('bhai saheb erroe aa gaya !!!'));
});

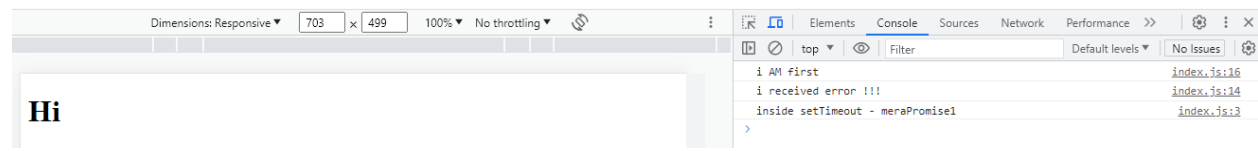
// meraPromise1.catch(function(val){
//   console.log('i received error !!!');
// });

// make arrow function

meraPromise1.catch((val)=>console.log('i received error !!!') );

console.log('i AM first');

```



## Promise – then() & catch()

```

let meraPromise1=new Promise(function(resolve,reject){
  setTimeout(function(){
    console.log('inside setTimeout - meraPromise1');
  },5000);
  reject(new Error('bhai saheb erroe aa gaya !!!'));
});

meraPromise1.then((val)=> {console.log(val)},(error)=>{console.log('i received error !!!')}) ;

console.log('i AM first');

```

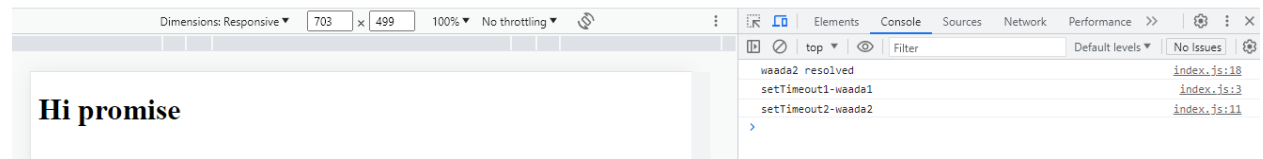


## Promise program code

```
let waada1=new Promise(function(resolve,reject){
  setTimeout(()=>{
    console.log('setTimeout1-waada1');
  },2000);
  resolve(true);
});

let output=waada1.then(function(){
  let waada2=new Promise(function(resolve,reject){
    setTimeout(()=>{
      console.log('setTimeout2-waada2');
    },3000);
    resolve('waada2 resolved');
  });
  return waada2;
});

output.then((value)=>console.log(value));
```



## Async & await

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>async & wait</h1>
  <script src="/fun3/index.js"></script>
</body>
</html>
```

Index.js

```
async function fun1(){
  return 7;
}
```

Output

The screenshot shows a web browser window with the text "async & wait" displayed. A red handwritten note "async always return Promise" is written over the text, with a red arrow pointing from the note to the console output. The browser's developer tools are open, showing the console with the following output:

```
Live reload enabled.
> fun1();
< Promise {<fulfilled>: 7}
  ▶ [[Prototype]]: Promise
  [[PromiseState]]: "fulfilled"
  [[PromiseResult]]: 7
```

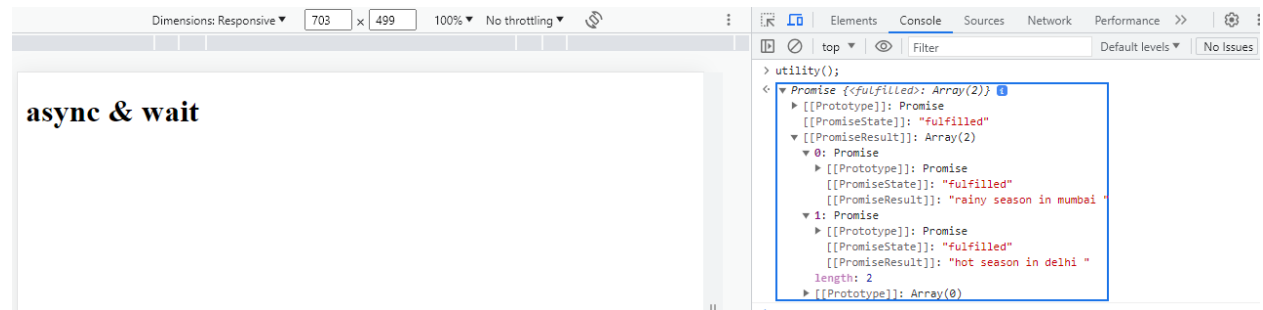
## Program 2)

```
async function utility(){

  let mumbaiMausam=new Promise((resolve,reject)=>{
    setTimeout(()=>{
      resolve('rainy season in mumbai ');
    },5000);
  });

  let delhiMausam=new Promise((resolve,reject)=>{
    setTimeout(()=>{
      resolve('hot season in delhi ');
    },6000);
  });

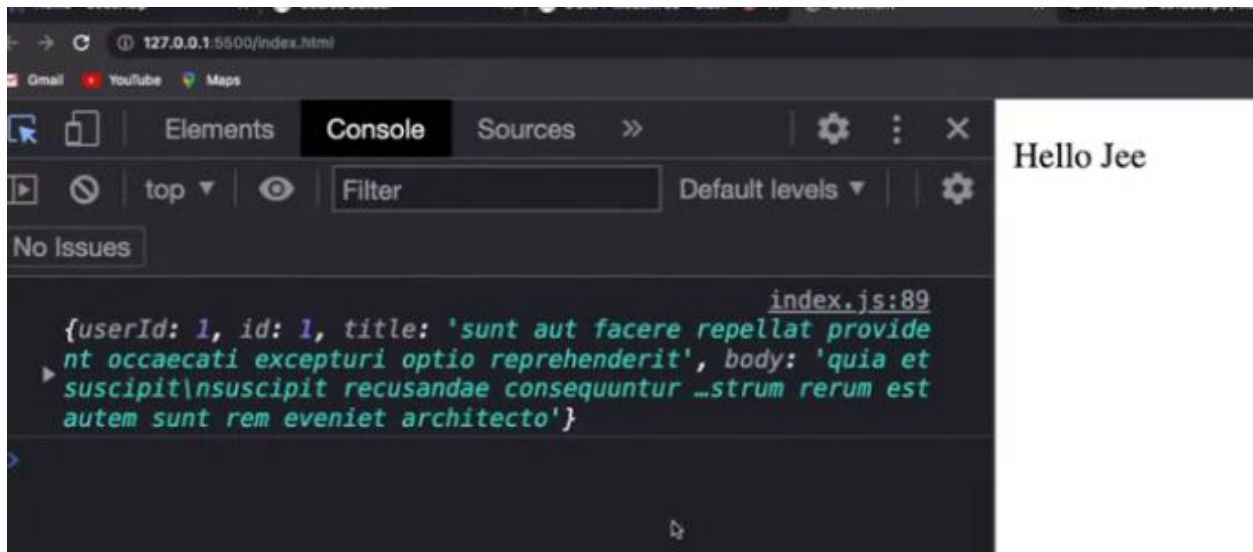
  let mm=mumbaiMausam;
  let dm=delhiMausam;
  return [mm,dm];
}
```



## API

```
83 let obj = {
84   heading:"head"
85 };
86 async function utility() {
87   let content = await fetch('https://jsonplaceholder.typicode.com/posts/1');
88   let output = await content.json();
89   console.log(output);
90 }
91 utility();
92
93
```

Handwritten notes in red:   
- "data" is written above line 87.   
- "next line" is written above line 88.   
- "data" is written below line 88.





## Program with Fake API

```
93  async function helper() {
94
95      let options = {
96          method: 'POST',
97          body: JSON.stringify({
98              title: 'foo',
99              body: 'bar',
100              userId: 1,
101          }),
102          headers: {
103              'Content-type': 'application/json; charset=UTF-8',
104          },
105      };
106
107      let content = await fetch('https://jsonplaceholder.typicode.com/posts', options);
108      let response = content.json();
109      return response;
110  }
111
```

```
112
113  async function utility() {
114      let ans = helper();
115      console.log(ans);
116  }
117
118  utility();
```

