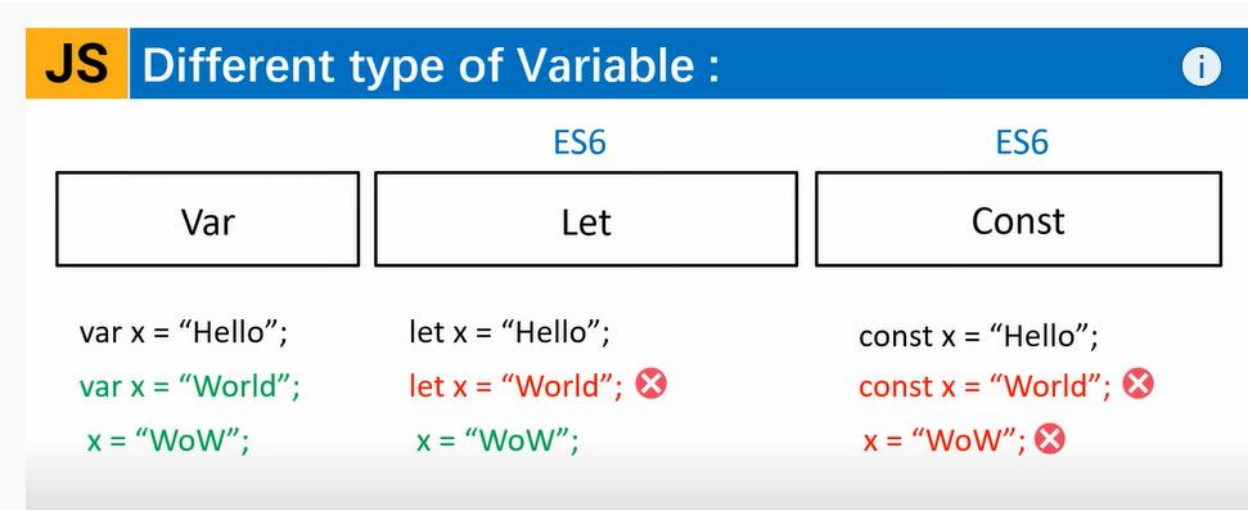
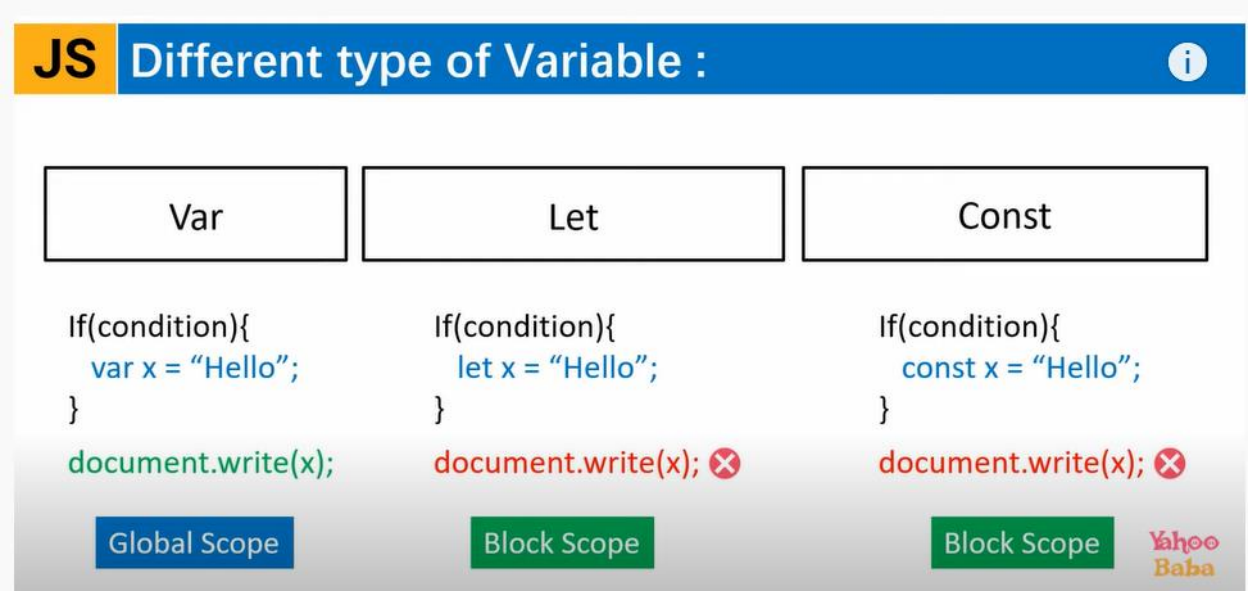
VARIABLE





TEMPLETE STRING

```
var user = "Yahoo Baba";
var greet =  "Hello " + user;
```

Template String

```
var user = "Yahoo Baba";
var greet =  `Hello ${user}`;
```

```
let user= "mehar dil";
let marks = 77;
let hello = `hello ${marks} ${user}`
document.write(hello);
```

```
let maths    = 30;
let  science = 70;
var marks;

function totalmarks(marks){
  marks = maths + science;
  marks = marks/200;
  var percentage =  marks*100;

  return ` your percentage is ${percentage}%`



}
let result = `${totalmarks(marks)}`;
document.write(result);
```

your percentage is 50%

JS Arrow Functions

```
function hello(){
   console.log("Hello");
}

hello();
```

```
let hello = function(){
   console.log("Hello");
}

hello();
```

Arrow Functions

```
let hello = () => console.log("Hello");

hello();
```

```
let maths    = 30;
let  science = 70;
var marks;

 let totalmarks = (marks) =>  {
  marks = maths + science + marks;
  marks = marks/200;
  let percentage =  marks*100;

  return ` your percentage is ${percentage}%`
}


let result = `${totalmarks(31 )}`;
document.write(result);
```

your percentage is 65.5%

If we take more than two are argument its not work and argument is object

Than we used forin loop

JS Functions with Multiple Arguments

```
function sum(num1, num2){
    console.log(num1 + num2);
}
sum(20, 30);
sum(20, 30, 40);
sum(20, 30, 40, 50);
sum("Yahoo Baba", 20, 30);
sum("Yahoo Baba", 20, 30, 40);
```

```
function sum(){
    let sum = 0;
    for(let i in arguments){
        sum += arguments[i];
    }
    console.log(sum);
}
```

Rest Operator

```
function sum(num1, num2){
    console.log(num1 + num2);
}
sum(20, 30);
sum(20, 30, 40);
sum(20, 30, 40, 50);
sum("Yahoo Baba", 20, 30);
sum("Yahoo Baba", 20, 30, 40);
```

```
function sum(name, ...args){
    let sum = 0;
    for(let i in args){
        sum += args[i];
    }
    console.log(sum);
    console.log(name);
}
```

```
let maths   = 30;
let  science = 70;
var name1;
var name2;



 let totalmarks = (name1,name2, ...args) =>  {
let marks =0;
for(let i in args){
  marks += maths + science + args[i];
  // marks = marks/200;
  // marks =  marks*100;
  document.write(`your marks % ${marks} <br>` )
}

  return ` your percentage is ${marks}%`
}


let result = `${totalmarks("mehar","ali",31,23,30)}`;
document.write(result);
```

```
function sum(name, ...args){

}

sum("Yahoo Baba", 20, 30, 40);

let arr = [20, 30, 40];
sum("Yahoo Baba", arr);

sum("Yahoo Baba", ...arr);
```

Rest Operator

Spread Operator

```
your marks % 131
your marks % 254
your marks % 384
your percentage is 384%
```

Object Literals

```
let name = "Yahoo Baba";

let obj = {
    name : name
};
```

```
let obj = {
    name
};
```

If both SAME WRITE THIS ONE TIME

```
let n = "name";

let obj = {
    [n] : "Yahoo Baba";
};
```

```javascript

let n = "name";

var Obj = {
 [n] : "mehardil",
 work : "cloudtek",
 detials : function(){
   return `I am ${this.name} work at ${this.work}`
 }
} ;


document.write(Obj.name);
document.write(Obj.work);
document.write(Obj.detials());
```

JS Object Literals : New Function Syntax

```
let obj = {
    name : "Yahoo Baba",
    show : function(){
        console.log(this.name);
    }
};
```

```
let obj = {
    name : "Yahoo Baba",
    show(){
        console.log(this.name);
    }
};
```

Function in short form

```
    course : "Btech",
    'detail show'(){
        return `${this.studentname} is student of ${this.
            course}`
    }
};

console.log(obj);
console.log(obj['detail show']());
```

Function return object.

```
let fname = "Yahoo";
let lname = "Baba";
let course = "Btech";

function student(fname, lname, course){
    let fullname = fname + " " + lname;
    return {fullname , course};
}

let s = student(fname, lname, course);
console.log(s.fullname);
console.log(s.course);
```

object return

JS Destructuring Array

```
let user = ["Yahoo Baba", 25];

let name = user[0];

let age = user[1];
```

```
let [name, age] = user;
```

```
<script>
    let user = ["Yahoo Baba", 22, "Delhi", ["Male",
        25000]];

    let [name, age = 20, city, [gender, salary]] =
        user;

    console.log(name);
    console.log(age);
    console.log(city);
    console.log(gender);
    console.log(salary);

</script>
```

Pass array to any function

```
function user([name, age = 20, city]){
    console.log(name);
    console.log(age);
    console.log(city);
}

user(["Yahoo Baba", 22, "Delhi"]);
```

Get array return

```
function user(){
   return ["Yahoo Baba", 22, "Delhi"];
}

let [name,age,city] = user();

console.log(city);
```

JS Destructuring Object

```
let user = ["Yahoo Baba", 25];

let [name, age] = user;
```

```
let user = {
   name : "Yahoo Baba",
   age : 25
};

let { name , age } = user;
```

JS What is Class & Object ?

Class **Car**

Properties
- Color
- Engine
- Seats
- AC
- Price

	Maruti Swift	Toyota Fortuner
Color	4	5
Engine	1200 CC	2700 CC
Seats	5	7
AC	Yes	Yes
Price	Rs.5,50,000	Rs.28,00,000

← Object →

JS What is Properties & Methods ?

Class Calculation

Properties

```
let a;
let b;
let c;
```

Methods

```
sum(){
  c = a + b;
  return c;
}
```

```
sub(){
  c = a - b;
  return c;
}
```

```
class hello{
    message(){
      console.log("Hello Everyone");
    }

}

let a = new hello();
```

Methods

Object

Constructor automate call. For each object

Prototype first store in variable than call.

No need to make variable.

Constructor

```
constructor(){
   console.log("hello");
}
```

Prototype

```
message(){
  console.log("hello")
}
```

Static

```
static name(){
  console.log("hello")
}
```

error

```
class CLASS2{
  constructor(){
    lets = cars
   console.log("car have 4 seats");
  }

tyre(){
    console.log(`car have 4 tyre ${this.carname}`);
  }
}

let a = new CLASS2();
a.cars = "kiya picanto";
 a.tyre();
```

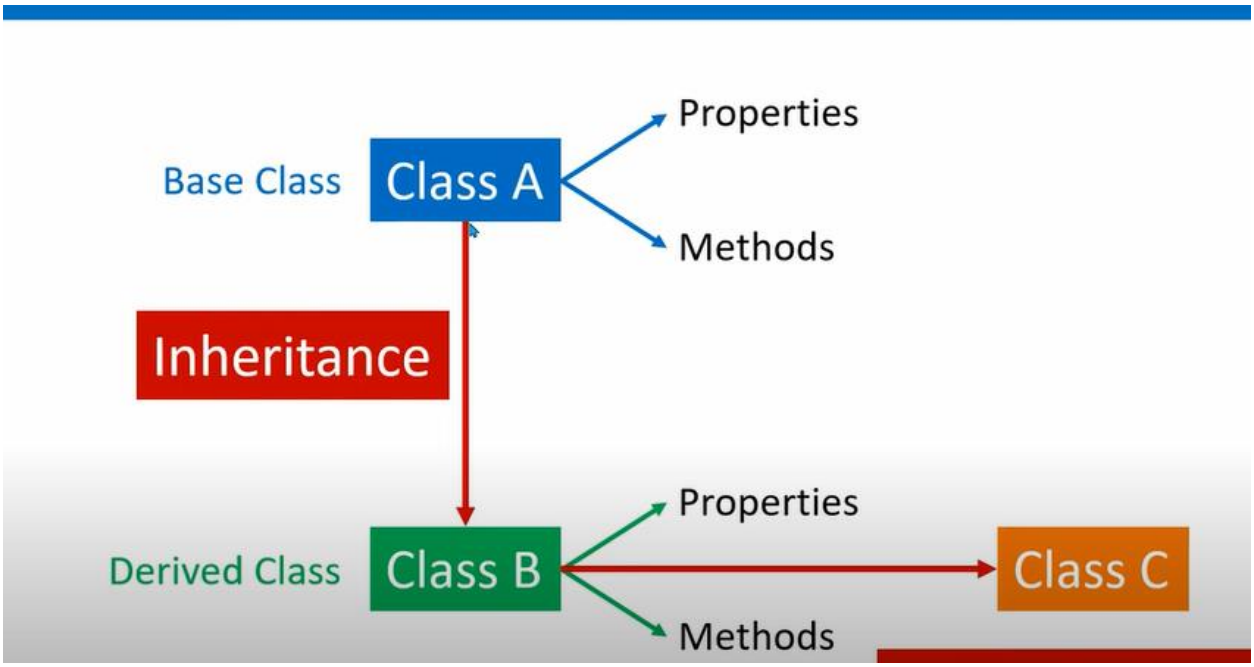
```
class CLASS2{
  constructor(name){
        this.carss =  name;

    console.log("car have 4 seats");
  }

tyre(){
    console.log(`car have 4 tyre ${this.carss}`);
  }
}

let a = new CLASS2("kiya picanto");
 a.tyre();
```

```
class student{
    constructor(name, age){
        this.studentname = name;
        this.studentage = age;
        console.log("constructor Function");
    }

    hello(){
        console.log(`Hello ${this.studentname}
            Your age is ${this.studentage}`);
    }
}

let a = new student("Yahoo baba",25);

a.hello();
```

```
<script>

    class student{
        constructor(name, age){
            this.studentname = name;
            this.studentage = age;
            console.log("constructor Function");
        }

        hello(){
            document.write(`Hello ${this.studentname}

                Your age is ${this.studentage}`);
        }
    }

    let a = new student("Yahoo baba",25);
    let b = new student("Ram Kumar",22);

    a.hello();
    b.hello();
```

```
    static staticMethod(){
        console.log("static Function");
    }
}

let a = new student("Yahoo baba",25);
let b = new student("Ram Kumar",22);

a.hello();
student.staticMethod();
```

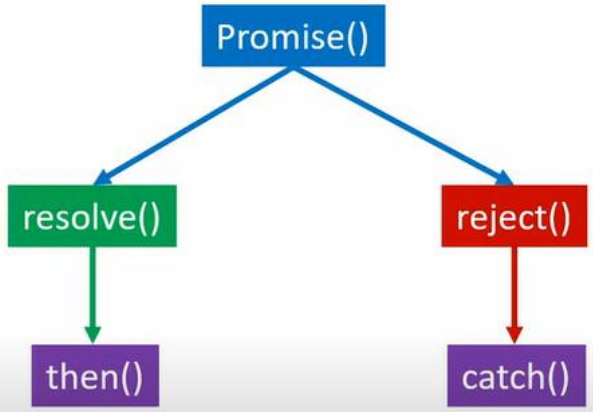
Classname


Properties
Base Class
Class A
Methods
Inheritance
Properties
Derived Class
Class B
Class C
Methods

```
class car{
  constructor(name){
        this.carname =  name;
        console.log("car name ");
  }
info()
{
  console.log("car is very small");
}
}
class carstructure extends car{

}


let a = new carstructure("MEHRAN");
a.info();
```

s

JS What is Promise ?




```
let prom = new Promise();

let prom = new Promise(function(){

});

let prom = new Promise(function(resolve, reject){

});
```

```
let p1 = new Promise(function(resolve, reject){
    console.log("First Promise");
    resolve("First");
});

let p2 = new Promise(function(resolve, reject){
    console.log("Second Promise");
    resolve("Second");
});

Promise.all( [p1, p2] ).then().catch();
```

```
 let x1 =  new Promise((resolve, reject) => {
        console.log("fetching data");
   setTimeout ( ()=>{

        resolve("you");
        console.log("ssss");
    }, 3000) });

let fullcondition =(result)=>{
        console.log(result);
        console.log("ggggggggg")
 }
let rejection =(error)=>{
    console.log(error);
}
let x2 = new Promise((resolve, reject) => {
        console.log("fetching data");
   setTimeout ( ()=>{
    resolve("you");
    console.log("ssss");
}, 3000) });
```

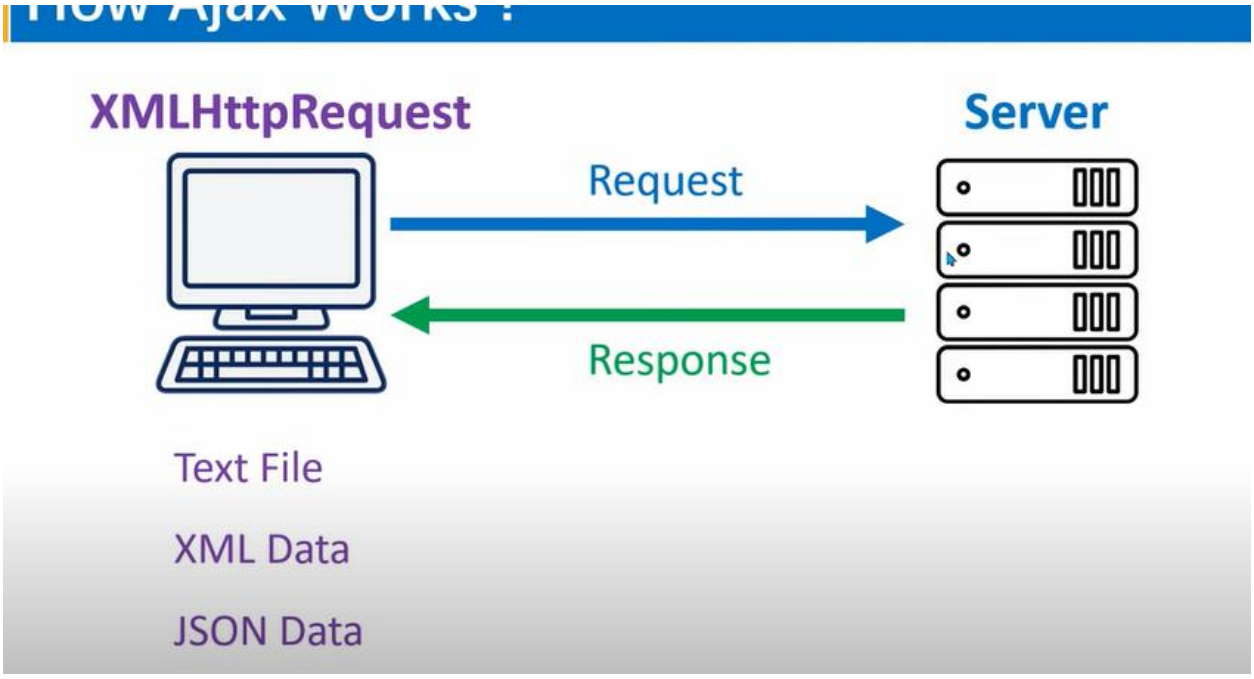
```
let fullcondition1 =(result)=>{
        console.log(result);
        console.log("ggggggggg")
 }
let rejection1 =(error)=>{
    console.log(error);
}

let x3 = new Promise((resolve, reject) => {
        console.log("fetching data");
   setTimeout ( ()=>{
    resolve("you");
    console.log("ssss");
}, 3000) });

let fullcondition2 =(result)=>{
        console.log(result);
        console.log("ggggggggg")
 }
let rejection2 =(error)=>{
    console.log(error);
}
Promise.all[x1,x2,x3].then(fullcondition).catch(rejection);
```

```
let completes;
function prom(a, b){
 return new Promise(function(come, nocome)

{  var c = a/b;
  if(c >  10){
      come(`i am coming ${c} `);
      }
  else{
    nocome(`i am not coming ${c} `);
  }
 });
}



let full = (result) => {
  console.log(result);
  console.log("hey");

}
let rejected =(error) =>
 {
  console.log(error)
 }
prom(2,3).then (full)
prom(false).catch (rejected)
```

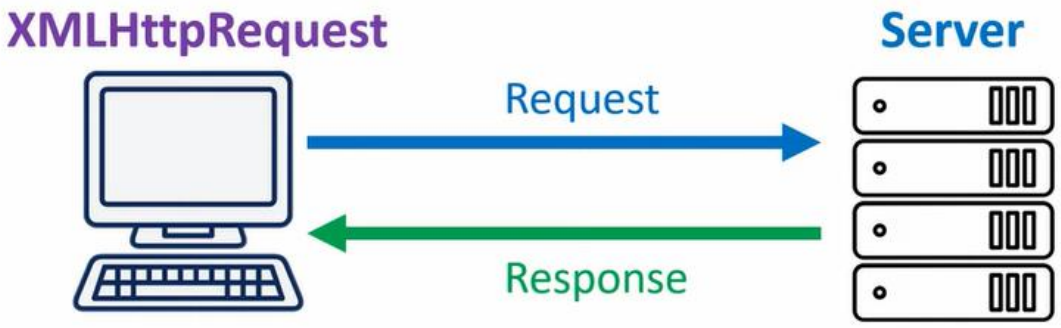
```html
<!DOCTYPE html>
<html>
<head>
 <title>Advance JS</title>
<script>
  let promiseCall = function(returnData, message) {
    return function(resolve, reject){
      setTimeout(() => {
        console.log(`The ${message} promise has resolved`);
        resolve(returnData);
      }, returnData * 100);
    }
  };

  let p1 = new Promise(promiseCall(10,"first"));
  let p2 = new Promise(promiseCall(20,"second"));
  let p3 = new Promise(promiseCall(30,"third"));

  var total = 0;
  Promise.all([p1,p2,p3]).then((result) =>{
```

```
}
let rejection2 =(error)=>{
    console.log(error);
}
let p1 = new Promise(x(10));
let p2 = new Promise(x(20));
let p3 = new Promise(x(30));
Promise.all[x1,x2,x3].then = (fullcondition) =>{
    console.log(p1);
    console.log(p2);
    console.log(p3);
}
```

In this we run three promise in which I given different parameter than combine this in one promis.all



Fast loading using ajax.



5 Steps

readyState

0: request not initialized

1: server connection established

2: request received

3: processing request

4: request finished and response is ready

Status

http status code

200: "OK"

403: "Forbidden"

404: "Not Found"

responseText

or

responseXML

```
var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        document.getElementById("demo").innerHTML = this.responseText;
    }
};

xhttp.open("GET", "filename.txt", true);
xhttp.send();
```

This

Live Server

this function run untill this twon condition is true.

```
        <title>Basic Layout</title>

    </head>
        <body>
    <p id = "demo">  you are good person</p>
    <button onclick="loaddata()">click</button>

    </body>
    <script >
        function loaddata(){
        var xhttp = new XMLHttpRequest();
        // document.write(xhttp);
        xhttp.onreadystatechange =function() {
          if (this.readyState  == 4 && this.status == 200)
          {
            console.log(typeof(this.responseText));
            console.log(this.responseText);
            document.getElementById("demo").innerHTML =  this.responseText;
          }
        }


        xhttp.open('GET',"https://jsonplaceholder.typicode.com/posts",true);
        xhttp.send();


      }
```

CODE

```
<script >
    function loaddata(){
    var xhttp = new XMLHttpRequest();
    // document.write(xhttp);
    xhttp.onreadystatechange =function() {
      if (this.readyState  == 4 && this.status == 200)
      {
        console.log(typeof(this.responseText));
        console.log(this.responseText);
        document.getElementById("demo").innerHTML =  this.responseText;
      }
    }
```

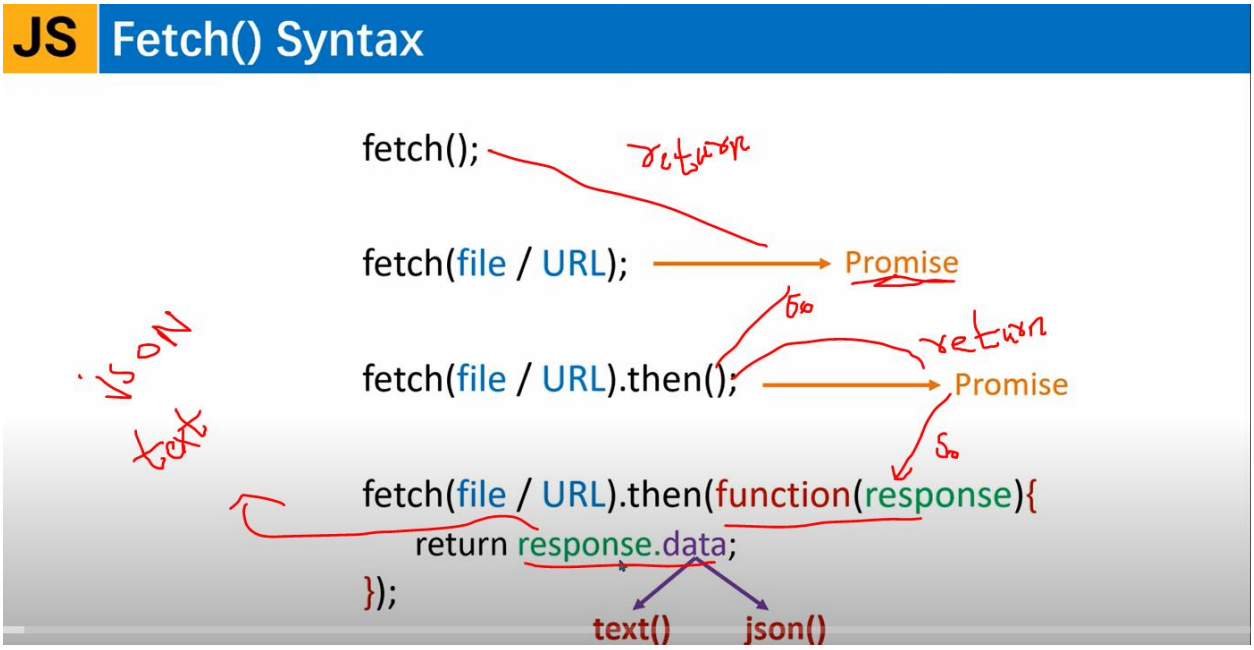
```
    xhttp.open('GET',"https://jsonplaceholder.typicode.com/posts",true);
    xhttp.send();


  }


</script>
```

quas\nvoluptate dolores velit et doloremque molestiae" }, { "userId": 1, "id": 7, "title": "magnam facilis autem", "body": "dolore placeat quibusdam ea quo vitae\nmagni quis enim qui quis quo nemo aut saepe\nquidem repellat excepturi ut quia\nsunt ut sequi eos ea sed quas" }, { "userId": 1, "id": 8, "title": "dolorem dolore est ipsam", "body": "dignissimos aperiam dolorem qui eum\nfacilis quibusdam animi sint suscipit qui sint possimus cum\nquaerat magni maiores excepturi\nipsam ut commodi dolor voluptatum modi aut vitae" }, { "userId": 1, "id": 9, "title": "nesciunt iure omnis dolorem tempora et accusantium", "body": "consectetur animi nesciunt iure dolore\nenim quia ad\nveniam autem ut quam aut nobis\net est aut quod aut provident volupta autem voluptas" }, { "userId": 1, "id": 10, "title": "optio molestias id quia eum", "body": "quo et expedita modi cum officia vel magni\ndoloribus qui repudiandae\nvero nisi sit\nquos veniam quod sed accusamus veritatis error" }, { "userId": 2, "id": 11, "title": "et ea vero quia laudantium autem", "body": "delectus reiciendis molestiae occaecati non minima eveniet qui voluptatibus\naccusamus in eum beatae sit\nvel qui neque voluptates ut commodi qui incidunt\nut animi commodi" }, { "userId": 2, "id": 12, "title": "in quibusdam tempore odit est dolorem", "body": "itaque id aut magnam\npraesentium quia et ea odit et ea voluptas et\nsapiente quia nihil amet occaecati quia id voluptatem\nincidunt ea est distinctio odio" }, { "userId": 2, "id": 13, "title": "dolorum ut in voluptas mollitia et saepe quo animi", "body": "aut dicta possimus sint mollitia voluptas commodi quo doloremque\niste corrupti reiciendis voluptatem eius rerum\nsit cumque quod eligendi laborum minima\nperferendis recusandae assumenda consectetur porro architecto ipsum ipsam" }, { "userId": 2, "id": 14, "title": "voluptatem eligendi optio", "body": "fuga et accusamus dolorum perferendis illo voluptas\nnon doloremque neque facere\nad qui dolorum molestiae beatae\nsed aut voluptas totam sit illum" }, { "userId": 2, "id": 15, "title": "eveniet quod temporibus", "body": "reprehenderit quos placeat\nvelit minima officia dolores impedit repudiandae molestiae nam\nvoluptas recusandae quis delectus\nofficiis harum fugiat vita }, { "userId": 2, "id": 16, "title": "sint suscipit perspiciatis velit dolorum rerum ipsa laboriosam odio", "body": "suscipit nam nisi quo aperiam aut\naspeniores eos fugit maiores voluptatibus quia\nvoluptatem quis ullam qui in alias quia est\nconsequatur magni mollitia accusamus ea nisi voluptate dicta" }, { "userId": 2, "id": 17, "title": "fugit voluptas sed molestias voluptatem provident", "body": "eos voluptas et aut odit natus earum\naspernatur fuga molestiae ullam\ndeserunt ratione qui eos\nqui nihil ratione nemo velit ut aut id quo" }, { "userId": 2, "id": 18, "title" "voluptate et itaque vero tempora molestiae", "body": "eveniet quo quis\nlaborum totam consequatur non dolor\nut et est repudiandae\nest voluptatem vel debitis et magnam" }, { "userId": 2,

Inspector | Console | Debugger | Network | Style Editor | Performance | Memory | Storage | Accessibility | Application

Filter Output | Errors | Warnings | Logs | Info | Debug | CSS | XHR | Reques


```
    "id": 3,
    "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
    "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et velit
aut"
  },
  {
    "userId": 1,
    "id": 4,
    "title": "eum et est occaecati",
```

```
fetch(file / URL).then(function(response){
    return response.data;
}).then(function(result){
    console.log(result);
}).catch(function(error){
    console.log(error);
});
```

Fetch() method works on Live Server

```
fetch("MEHAR.txt")
.then((response)=>response.text())
.then((data)=>document.write(data))
```

Code

```
fetch("https://jsonplaceholder.typicode.com/posts")
.then((response)=>{
    return  response.json()
})
.then((data)=>{
    console.log(data);
    for (var x in data){
        document.write(x + " " + "<br>")
        document.write(data[x].userId + " ")
```

```
    }
})
.catch((data)=>{console.log("error in fetching");
})
```

to update data on server we used put

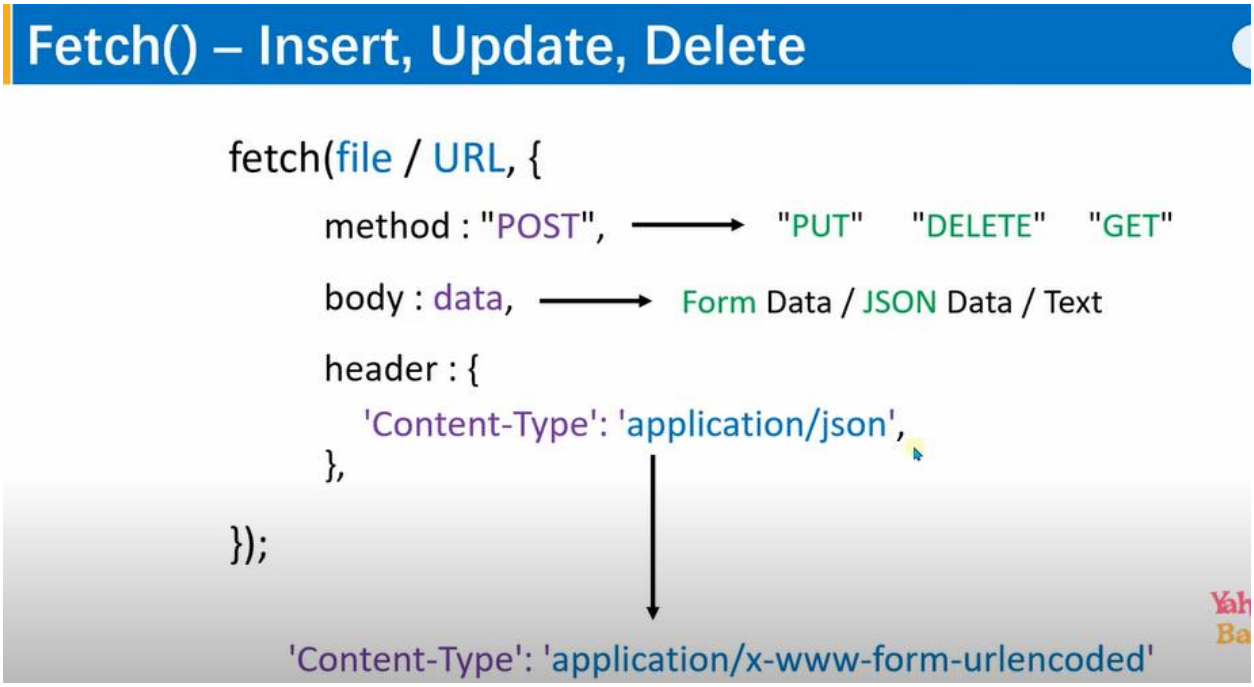
to delete data on server we used delete

to upload data on server we used get

body --- used to tell what type of data we used such as

form data, json data, text

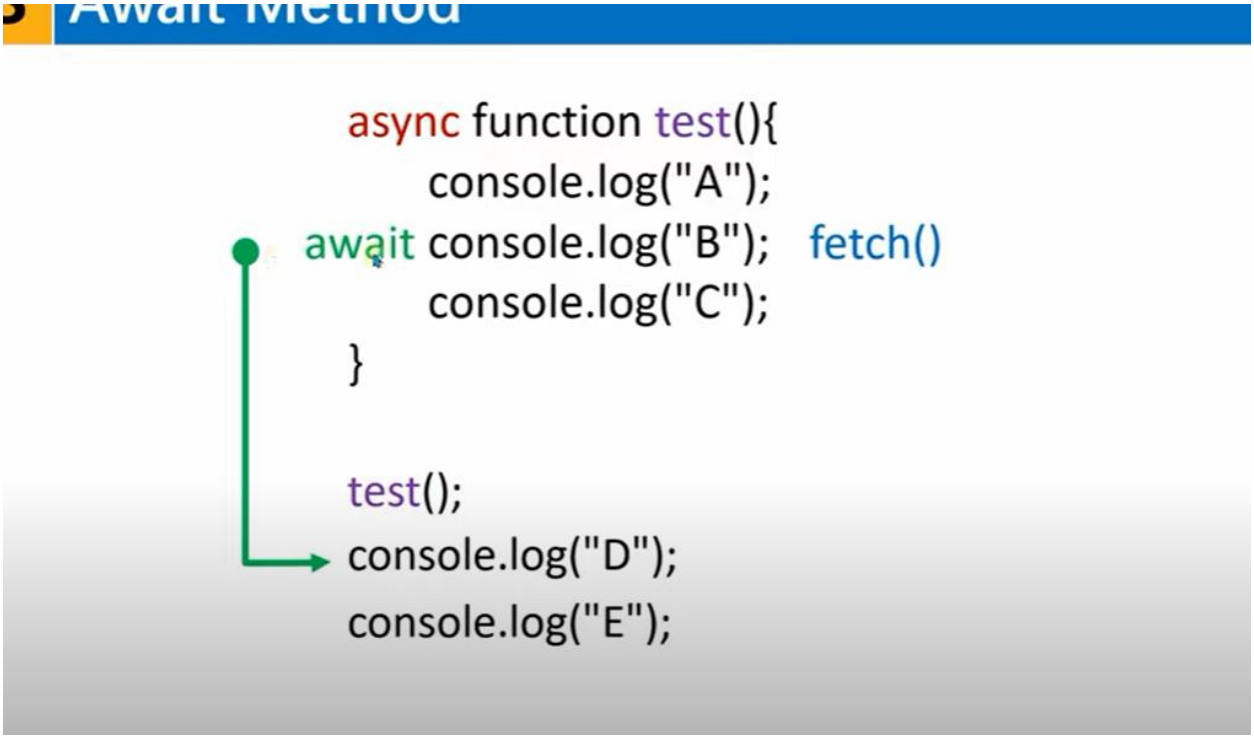
header --- give content type



Asyc function always return promise.

```html
<head>
 <title>Advance JS</title>
  <script>
      async function test(){
        return "Hello";
      }

      console.log(test());
  </script>
</head>
<body>

</body>
</html>
```

await



```html
<html>
<head>
 <title>Advance JS</title>
  <script>
      async function test(){
       console.log("2 : Message");
       await console.log("3 : Message");
       console.log("4 : Message");
      }

      console.log("1 : Message");
      test();
      console.log("5 : Message");
  </script>
</head>
<body>

</body>
</html>
```

```
Elements   Console   »
top   Filter   Def.
1 : Message                    async-await.html:12
2 : Message                    async-await.html:7
3 : Message                    async-await.html:8
5 : Message                    async-await.html:14
4 : Message                    async-await.html:9
Runtime                        page.js:1
▸{action: "tab_url_change", recognizing: false}
false
Runtime                        page.js:1
▸{action: "set_shortcut", shortcut: Array(4)}
false
>
```

```html
</head>
<body>
  <form id="myForm">
    Title <input type="text" id="titleText"> <br><br>
    Body <input type="text" id="bodyText"> <br><br>
    User Id <input type="text" id="userid"> <br><br>
    <input type="submit" id="saveForm">
  </form>

  <script>
      document.getElementById("titleText")

    var obj = {
        title: document.getElementById("titleText").value,
        body: document.getElementById("bodyText").value,
        userId: document.getElementById("userid").value,
      };
```

JavaScript : Iterators

- while()
- do..while()
- for()
- for..of()
- for..in()
- foreach()
- map()

```
var x = ["Apple", "Orange", "Grapes"];
```

```
for (let i = 0; i < x.length; i++) {
   console.log(x[i]);
}
```

```
for (let x of values) {
   console.log(values);
}
```

for of

JS JavaScript : Iterators

- String
- Object

```
var x = ["Apple", "Orange", "Grapes"];

let y = x[Symbol.iterator]();

y.next();

y.next();

y.next();
```

```html
<script>
  let numbers = [100,200,300];

  let iter = numbers[Symbol.iterator]();

  console.log(iter.next());
  console.log(iter.next());
  console.log(iter.next());
  console.log(iter.next());
</script>
</head>
<body>

</body>
</html>
```

Elements Console »

top Filter

iterator.html:10
▸ {value: 100, done: false}

iterator.html:11
▸ {value: 200, done: false}

iterator.html:12
▸ {value: 300, done: false}

iterator.html:13
▸ {value: undefined, done: true}

```html
<script>
  let numbers = [100,200,300];

  let iter = numbers[Symbol.iterator]();

  iter.next();
  console.log(iter.next().value);
  console.log(iter.next().done);

</script>
```

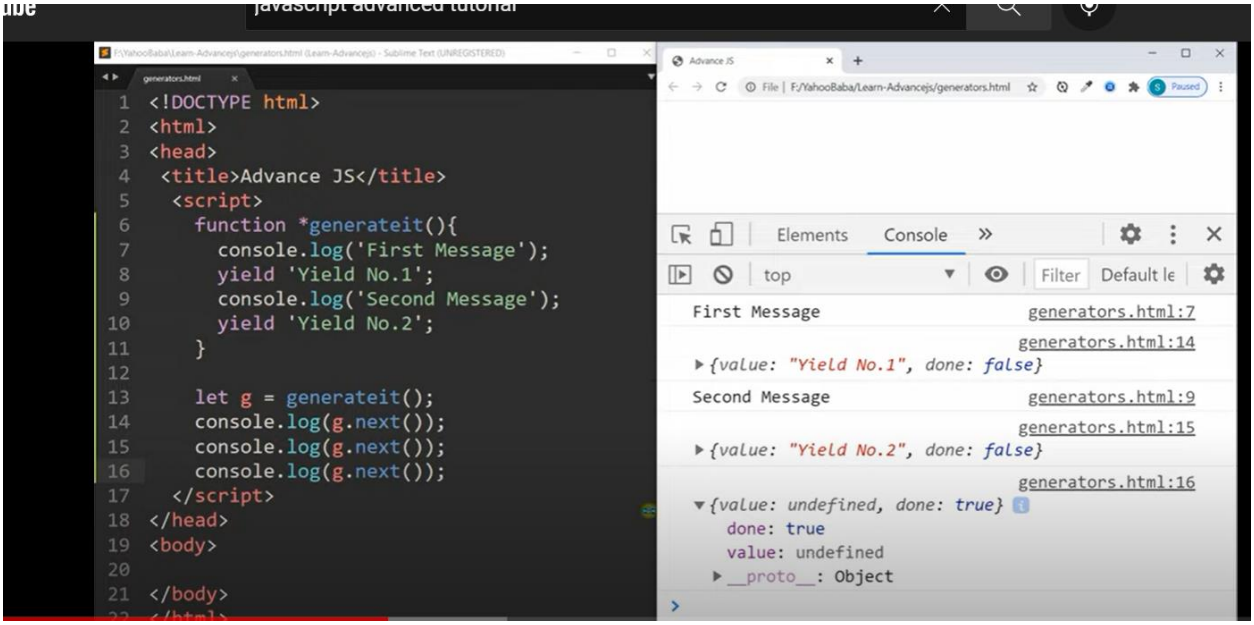
```html
<script>
  let numbers = [100,200,300,400,500];

  let iter = numbers[Symbol.iterator]();

  let result = iter.next();

  while(!result.done){
    console.log(result.value);
    result = iter.next();
  }

</script>
</head>
```

```
<title>Advance JS</title>
<script>
  // let numbers = [100,200,300,400,500];
  let str = "Yahoo Baba";

  let iter = str[Symbol.iterator]();

  let result = iter.next();

  while(!result.done){
    console.log(result.value);
    result = iter.next();
  }

</script>
</head>
```

```
<script>
  function numberIterator(arr){
    var nextNum = 0;
    return {
      next(){
        if(nextNum < arr.length){
            return{
              value : arr[nextNum++],
              done : false
            }
        }else{
            return{
              done : true
            }
        }

      }
    }
  }
```

Next Self Iterator

```
let numbers = [100,200,300,400,500];

let num = numberIterator(numbers);
console.log(num.next());
console.log(num.next());
console.log(num.next());
console.log(num.next());


</script>
```



```html
<!DOCTYPE html>
<html>
<head>
 <title>Advance JS</title>
  <script>
    function *generateit(){
      console.log('First Message');
      yield 'Yield No.1';
      console.log('Second Message');
      yield 'Yield No.2';
    }

    let g = generateit();
    console.log(g.next());
    console.log(g.next());
    console.log(g.next());
  </script>
</head>
<body>

</body>
```

```
First Message                                   generators.html:7
                                                generators.html:14
▶{value: "Yield No.1", done: false}
Second Message                                  generators.html:9
                                                generators.html:15
▶{value: "Yield No.2", done: false}
                                                generators.html:16
▼{value: undefined, done: true}
    done: true
    value: undefined
  ▶__proto__: Object
```

```html
<head>
 <title>Advance JS</title>
  <script>
    function *generateit(){
      yield 55;
      yield ['Node','Angular','React'];
    }

    let g = generateit();
    console.log(g.next());
    console.log(g.next());


  </script>
</head>
<body>

</body>
</html>
```

```
▶{value: 55, done: false}                       generators.html:12
                                                generators.html:13
▼{value: Array(3), done: false}
    done: false
  ▼value: Array(3)
      0: "Node"
      1: "Angular"
      2: "React"
      length: 3
    ▶__proto__: Array(0)
  ▶__proto__: Object
```

22:33 / 27:58

```html
<html>
<head>
 <title>Advance JS</title>
  <script>
    "use strict";
    lea = 10;
    console.log(a);

  </script>
</head>
<body>

</body>
</html>
```

```
▶Uncaught ReferenceError: strict.html:7
a is not defined
    at strict.html:7
```

```html
<!DOCTYPE html>
<html>
<head>
 <title>Advance JS</title>
  <script>
    try{
      console.log('Start of Try');

      blabla;

      console.log('End of Try');
    }catch{
      console.log('Error has occurred.');
    }

  </script>
</head>
<body>

</body>
</html>
```

```
Start of Try                    error-handling.html:7
Error has occurred.             error-handling.html:13
```

```html
<head>
 <title>Advance JS</title>
  <script>
    try{
      setTimeout(function(){
        blabla;
      }, 1000)
    }catch(error){
      // console.log('Error has occurred.')
      console.log(error);

    }

  </script>
</head>
<body>
```

```
Uncaught ReferenceError:        error-handling.html
blabla is not defined
    at error-handling.html:8
```