// Callback0 sync code ============================>

```
const posts =[
   {'title':"Post One","body":"This is Post One"},
   {'title':"Post Two","body":"This is Post Two"}
];

function createPost(post){
   setTimeout(function(){
      posts.push(post);
   },2000);
}

function getPosts(){
   setTimeout(function(){
      let output='';
      posts.forEach(function(post){
        output +=`<li>${post.title}</li>`
                });
      document.body.innerHTML=output;
   },1000);
}

createPost({"title":"Post Three","body":"This is Post Three"});
getPosts();
```

// callback1 Async Code ========================>

```
const posts =[
   {'title':"Post One","body":"This is Post One"},
   {'title':"Post Two","body":"This is Post Two"}
];

function createPost(post, callback) {
  setTimeout(function() {
    posts.push(post);
    callback();
  }, 2000);
}


function getPosts() {
  setTimeout(function() {
```

```
    let output = '';
    posts.forEach(function(post){
      output += `<li>${post.title}</li>`;
    });
    document.body.innerHTML = output;
  }, 1000);
}

createPost({title: 'Post Three', body: 'This is post three'}, getPosts);
```

// Easy HTTP library

// easyhttp0.js ==>

```
// es5 library type
function easyHTTP(){
    this.http = new XMLHttpRequest();

}
/* Section (1)
easyHTTP.prototype.get= function(url){
    this.http.open('GET',url,true);
    this.http.onload= function(){
        if (this.http.status=== 200){
            console.log(this.http.responseText);
            }
    }
    this.http.send();
}

*/

/* section (2),by using self
    due to this is not defined in the following function , we will workaround by useing another
variable is called self
    also, you can use arrow function to solve this problem if you will apply es6
    easyHTTP.prototype.get= function(url){
    this.http.open('GET',url,true);
    let self=this;
```

```
    this.http.onload= function(){
        if (self.http.status=== 200){
            console.log(self.http.responseText);

            }
    }
    this.http.send();
}

*/
/*section (3)
easyHTTP.prototype.get= function(url){
    this.http.open('GET',url,true);
    let self=this;
    this.http.onload= function(){
        if (self.http.status=== 200){
            return self.http.responseText ;
            }
    }
    this.http.send();
}
*/
/* Section (4) will use callback function
easyHTTP.prototype.get= function(url,callback){
    this.http.open('GET',url,true);
    let self=this;
    this.http.onload= function(){
        if (self.http.status=== 200){
            callback(self.http.responseText) ;
        }
    }
    this.http.send();
}

*/
/* Section (5) adding null in case error */
easyHTTP.prototype.get= function(url,callback){
    this.http.open('GET',url,true);
    let self=this;
    this.http.onload= function(){
        if (self.http.status=== 200){
            callback(null,self.http.responseText) ;
        }else {
```

```
            callback('Error :' + self.http.status) ;
        }
    }
    this.http.send();
}
```

/ app0.js ================================>

```
const http = new easyHTTP();

 /*Section (1,2)

   http.get('https://jsonplaceholder.typicode.com/posts');

*/



/* (3) In this case will get undefined because data was returned before actually is added

the following code is in * { sync code mode } "

const posts=http.get('https://jsonplaceholder.typicode.com/posts');
console.log(posts)

*/

/* section (4) to solve this problem , use callback function  as follow

    the following code is in * { Aync code mode } "
    http.get('https://jsonplaceholder.typicode.com/posts',function(posts){
    console.log(posts)
});


*/


/* section (5) using null

    http.get('https://jsonplaceholder.typicode.com/posts',function(err,posts){
    if (err){
```

```
        console.log(err)
    }else{
        console.log(posts)
    }

});
*/

/* section (6) Get one post*/

    http.get('https://jsonplaceholder.typicode.com/posts/1',function(err,post){
    if (err){
        console.log(err)
    }else{
        console.log(post)
    }

});
```

// easyhttp1 ==>

```
// es5 library type
function easyHTTP(){
    this.http = new XMLHttpRequest();
}
// Make GET Request
easyHTTP.prototype.get= function(url,callback){
    this.http.open('GET',url,true);
    let self=this;
    this.http.onload= function(){
        if (self.http.status=== 200){
            callback(null,self.http.responseText) ;
        }else {
            callback('Error :' + self.http.status) ;
        }
    }
    this.http.send();
}

// Make POST Request
```

```
easyHTTP.prototype.post=function(url,data,callback){
    this.http.open('POST',url,true);
    this.http.setRequestHeader('content-type','application/json');
    let self=this;
    this.http.onload=function(){
        callback(null,self.http.responseText);
    }
    this.http.send(JSON.stringify(data));

}


// Make an HTTP PUT Request
easyHTTP.prototype.put = function(url, data, callback) {
  this.http.open('PUT', url, true);
  this.http.setRequestHeader('Content-type', 'application/json');

  let self = this;
  this.http.onload = function() {
    callback(null, self.http.responseText);
  }

  this.http.send(JSON.stringify(data));
}

// Make an HTTP DELETE Request
easyHTTP.prototype.delete = function(url, callback) {
  this.http.open('DELETE', url, true);

  let self = this;
  this.http.onload = function() {
    if(self.http.status === 200) {
      callback(null, 'Post Deleted');
    } else {
      callback('Error: ' + self.http.status);
    }
  }

  this.http.send();
}
```

```
// app1.js ============================>
const http = new easyHTTP;

// Get Posts
// http.get('https://jsonplaceholder.typicode.com/posts', function(err, posts) {
//   if(err) {
//     console.log(err);
//   } else {
//     console.log(posts);
//   }
// });

// Get Single Post
// http.get('https://jsonplaceholder.typicode.com/posts/1', function(err, post) {
//   if(err) {
//     console.log(err);
//   } else {
//     console.log(post);
//   }
// });

// Create Data
const data = {
  title: 'Custom Post',
  body: 'This is a custom post'
};

// Create Post
// http.post('https://jsonplaceholder.typicode.com/posts', data, function(err, post) {
//   if(err) {
//     console.log(err);
//   } else {
//     console.log(post);
//   }
// });

// Update Post
 http.put('https://jsonplaceholder.typicode.com/posts/5', data, function(err, post) {
   if(err) {
     console.log(err);
   } else {
```

```
    console.log(post);
  }
});

// Delete Post
//http.delete('https://jsonplaceholder.typicode.com/posts/1', function(err, response) {
//  if(err) {
//    console.log(err);
//  } else {
//    console.log(response);
//  }
//});
```

// promise ============================>

```
const posts = [
  {title: 'Post One', body:'This is post one'},
  {title: 'Post Two', body: 'This is post two'}
];

function createPost(post) {
  return new Promise(function(resolve, reject){
    setTimeout(function() {
      posts.push(post);

      const error = false;

      if(!error) {
        resolve();
      } else {
        reject('Error: Something went wrong');
      }
    }, 2000);
  });
}

function getPosts() {
  setTimeout(function() {
    let output = '';
    posts.forEach(function(post){
      output += `<li>${post.title}</li>`;
    });
```

```
    document.body.innerHTML = output;
  }, 1000);
}

createPost({title: 'Post Three', body: 'This is post three'})
.then(getPosts);
```

// app.js fetch api==========>

```
document.getElementById('button1').addEventListener('click',getText);
document.getElementById('button2').addEventListener('click',getJson);
document.getElementById('button3').addEventListener('click',getExternal);


// Get Local Text file
function getText(){
    fetch('test.txt')
        .then(function(res){
//            console.log(res) ;
//            console.log(res.text());
        return res.text();
    })

    .then(function (data){
        console.log(data);
        document.getElementById("output").innerHTML=data;
    })
    .catch(function(err){
        console.log(err)
    })
}

// Get local json file
function getJson(){
    fetch('posts.json')
        .then(function(res){
        return res.json();
    })
    .then(function (data){
        console.log(data)
```

```
        let output='';
        data.forEach(function(post){
            output+=`<li>${post.title}</li>`
        })
        document.getElementById("output").innerHTML=output;
    })
    .catch(function(err){
        cosnole.log(err)
    })

}

// Get from External

function getExternal(){
    fetch('https://api.github.com/users')
        .then(function(res){
        return res.json();
    })
    .then(function (data){
        console.log(data)
        let output='';
        data.forEach(function(user){
            output+=`<li>${user.login}</li>`
        })
        document.getElementById("output").innerHTML=output;
    })
    .catch(function(err){
        cosnole.log(err)
    })

}
```

// Arrow Function ==========

```
// const sayHello = function() {
//   console.log('Hello');
// }

// const sayHello = () => {
//   console.log('Hello');
// }
```

```
// One line function does not need braces
// const sayHello = () => console.log('Hello');

// One line returns
// const sayHello = () => 'Hello';

// Same as above
// const sayHello = function() {
//   return 'Hello';
// }

// Return object
// const sayHello = () => ({ msg: 'Hello' });

// Single param does not need parenthesis
// const sayHello = name => console.log(`Hello ${name}`);

// Multuiple params need parenthesis
// const sayHello = (firstName, lastName) => console.log(`Hello ${firstName} ${lastName}`);

// sayHello('Brad', 'Traversy');

const users = ['Nathan', 'John', 'William'];

// const nameLengths = users.map(function(name) {
//   return name.length;
// });

// Shorter
// const nameLengths = users.map((name) => {
//   return name.length;
// });

// Shortest
const nameLengths = users.map(name => name.length);

console.log(nameLengths);
```

// easyhttp2 es6 ================================>

```
/**
```

```
* EasyHTTP Library
* Library for making HTTP requests
*
**/

class EasyHTTP {

 // Make an HTTP GET Request
 get(url) {
   return new Promise((resolve, reject) => {
     fetch(url)
     .then(res => res.json())
     .then(data => resolve(data))
     .catch(err => reject(err));
   });
 }

 // Make an HTTP POST Request
 post(url, data) {
   return new Promise((resolve, reject) => {
     fetch(url, {
       method: 'POST',
       headers: {
         'Content-type': 'application/json'
       },
       body: JSON.stringify(data)
     })
     .then(res => res.json())
     .then(data => resolve(data))
     .catch(err => reject(err));
   });
 }

  // Make an HTTP PUT Request
  put(url, data) {
   return new Promise((resolve, reject) => {
     fetch(url, {
       method: 'PUT',
       headers: {
         'Content-type': 'application/json'
       },
       body: JSON.stringify(data)
     })
```

```
      .then(res => res.json())
      .then(data => resolve(data))
      .catch(err => reject(err));
    });
  }

  // Make an HTTP DELETE Request
  delete(url) {
    return new Promise((resolve, reject) => {
      fetch(url, {
        method: 'DELETE',
        headers: {
          'Content-type': 'application/json'
        }
      })
      .then(res => res.json())
      .then(() => resolve('Resource Deleted...'))
      .catch(err => reject(err));
    });
  }

}
```

// app2.js es6 ======>

```
const http = new EasyHTTP;

// Get Users
// http.get('https://jsonplaceholder.typicode.com/users')
//   .then(data => console.log(data))
//   .catch(err => console.log(err));

// User Data
const data = {
  name: 'Yasser Ali',
  username: 'yasserali',
  email: 'yali@gmail.com'
}

// Create User
 http.post('https://jsonplaceholder.typicode.com/users', data)
```

```
  .then(data => console.log(data))
  .catch(err => console.log(err));

// Update Post
 http.put('https://jsonplaceholder.typicode.com/users/2', data)
  .then(data => console.log(data))
  .catch(err => console.log(err));

// Delete User
http.delete('https://jsonplaceholder.typicode.com/users/2')
.then(data => console.log(data))
.catch(err => console.log(err));
```