promise

Promise   Promise then

setTimeout()   ajax   ajax

```javascript
function a() {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            console.log('    a');
            resolve('   a ');
        }, 1000);
    });
}

function b(value) {
    console.log(value)
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            console.log('    b');
            resolve('   b ');
        }, 2000);
    });
}

function c() {
    console.log('    c')
}
a().then(b).then(c);
```

* then return   promise resolve   then
* then return   promise   then

promise

promise   angular1.x $http

```
//1
```

```
$http({
 method: 'GET',
 url: 'news.json',
}).then(function successCallback(response) {
 console.log(response)
}, function errorCallback(response) {
 console.log(response)
})
//2
.then(function() {
 return $http({
  method: 'GET',
  url: 'data.json',
 })
}).then(function(data) {
 console.log(data)
})
//3
.then(function() {
 setTimeout(function() {
  console.log("   ")
 }, 1000)
})
```

await

await   await

       Promise   await

      Promise   await   Promise   resolve   resolve   await

```
function a() {
  return new Promise(function(resolve){
    setTimeout(()=>{
     console.log("a")
     resolve()
    },1000)
  });
}
```

```
function b() {
  return new Promise(function(resolve){
    setTimeout(()=>{
      console.log("b")
      resolve()
    },1000)
  });
}

function c() {
  return new Promise(function(resolve){
    setTimeout(()=>{
      console.log("c")
      resolve()
    },1000)
  });
}

//ES6
a()
  .then(b)
  .then(c);

//ES2017
await a();
await b();
await c();
```

await   promise   .then(..)   await   then

```
var sleep = function (time) {
    return new Promise(function (resolve, reject) {
        setTimeout(function () {
            resolve();
        }, time);
    })
};

var start = async function () {
```

```
    //
    console.log('start');
    await sleep(3000);
    console.log('end');
};
start();
```

- async   async   await
- await   promise
- await   promise

deferred

$.ajax()   1.5.0   jQuery   XHR   1.5.0   deferred   done

```
$.when(function(dtd) {
　var dtd = $.Deferred(); //    deferred
　setTimeout(function() {
　　console.log(0);
　　dtd.resolve(1); //  deferred        done
　　//dtd.reject(); // resolve    fail
　}, 1000);
　return dtd;
}()).done(function(num) {
　console.log(num);
}).done(function() {
　console.log(2);
}).done(function() {
　console.log(2);
})

//ajax     deferred
$.when($.post("index.php", {
　name: "wscat",
　}), $.post("other.php"))
　.done(function(data1, data2) {
　　// ajax      done
　　console.log(data1, data2);
　}).fail(function(err) {
```

```
  // ajax    fail
  console.log(err)
})
```

event loop

||||

then	setTimeout

```
console.log('script start');
setTimeout(function() {
  console.log('setTimeout');
}, 0);//
Promise.resolve().then(function() {
  console.log('promise1');
}).then(function() {//then
  console.log('promise2');
});
console.log('script end');
//
//
```

script start, script end, promise1, promise2, setTimeout

```
console.log('script start');
setTimeout(function() {
  console.log('timeout1');
}, 10);
new Promise(resolve => {
    console.log('promise1');
    resolve();
    setTimeout(() => console.log('timeout2'), 10);
}).then(function() {
    console.log('then1')
})
console.log('script end');
```

```
script start, promise1, script end, then1, timeout1, timeout2
```

- Event Loop
- JavaScript

await/async

```
npm i -D babel-core babel-polyfill babel-preset-es2015 babel-preset-stage-0 babel-loader
```

`.babelrc`

```
{
    "presets": [
        "stage-0",
        "es2015"
    ]
}
```

inde `app.js` require b `await` p `async`

```
require("babel-core/register");
require("babel-polyfill");
require("./app.js");
```

Babel 6 regeneratorRuntime is not defined

await,async,promise

```
//async
((async() => {
　try {
　　//await                  await
　　const a = await (() => {
　　　return new Promise((resolve, reject) => {
　　　　setTimeout(() => {
　　　　　console.log(1)
　　　　　resolve(2);
　　　　　//reject(3)
```

```
      }, 1000)
    });
  })();
  const b = await (() => {
    return new Promise((resolve, reject) => {
      setTimeout(() => {
        console.log(1)
        //resolve(2);
        reject(3)
      }, 1000)
    });
  })();
  console.log(4)
  console.log(a)  //3
  return b;
} catch(err) {
  // try     reject
  console.log(err);
  return err;
}
})()).then((data) => {
 console.error(data)
}).catch((err) => {
 console.error(err)
})
//    213
```

- async   await   await   await   await
- await   promise   generator
- await   async
- await   Promise   rejected   await   try...catch

```
//  try...catch  Promise reject
async function ajax(data) {
 try {
  return await new Promise((resolve, reject) => {
   setTimeout(() => {
    console.log(data)
    resolve(data); //
   }, 2000);
```

```
  });
} catch(err) {}
}
async function io() {
 try {
  const response = await new Promise((resolve, reject) => {
   setTimeout(() => {
    reject("io"); //
   }, 1000);
  });
  //resolve          return
  return response
 } catch(err) {
  console.log(err);
 }
}
//
(async() => {
 await ajax("ajax1");
 await ajax("ajax2");
 await io();
})()


(async() => {
    let [ajax1, ajax2] = await Promise.all([ajax("ajax1"), ajax("ajax2"),io()]);
    return [ajax1,ajax2]
})()
```

worker

Web Worker   HTML5   API   JavaScript **UI**

```
<input id="btn" type="button" value="  " />
<script>
    let myWorker = new Worker('./worker.js');
    let button = document.querySelector('#btn');
    // myWorker.onmessage = function (event) { //
    //     console.log('        :' + event.data);
    //     myWorker.terminate(); //
```

```javascript
        // }
    myWorker.addEventListener('message', function (e) {
        console.log('        :' + event.data);
        myWorker.terminate(); //
    });
    //    error
    myWorker.addEventListener('error', function (e) {
        console.log('  ', e);
    });
    button.onclick = function () {
        myWorker.postMessage("        ");  //
    }
</script>
```

worker.js

```javascript
addEventListener('message', function (e) {
    postMessage('       : ' + e.data);
}, false);
```

blob

```javascript
let script = 'console.log("hello world!");'
let workerBlob = new Blob([script], { type: "text/javascript" });
let url = URL.createObjectURL(workerBlob);
let myWorker = new Worker(url);
```
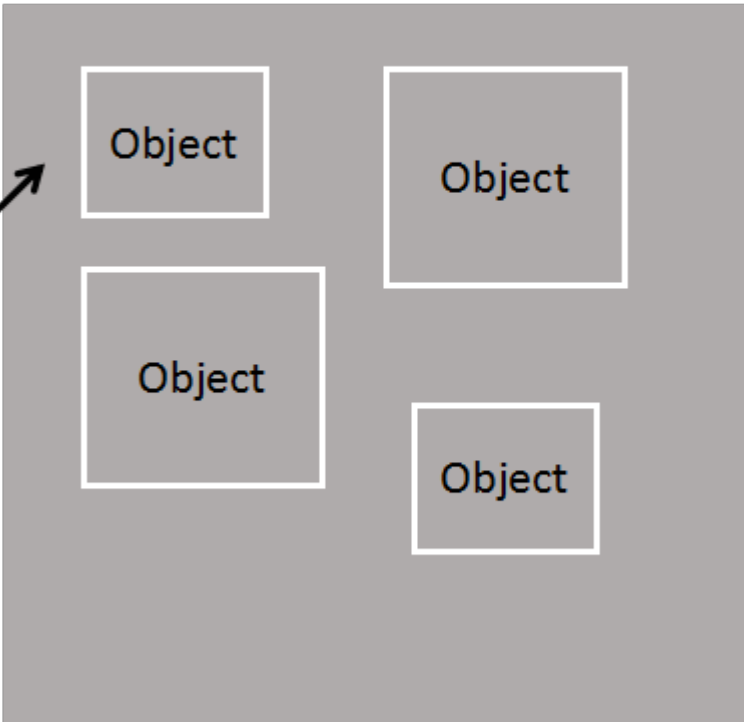
栈内存　　　　　　　　　　　堆内存

	(heap)	(stack)
	heap　　　　　　heap	stack　　　　　　stack
	: ,	Boolean Number String Undefined Null
	new	

new　　　　　　　　　　　new

```
var a = new String('123')
```

```
var b = String('123')
var c = '123'
console.log(a==b, a===b, b==c, b===c, a==c, a===c)
>>> true false true true true false
console.log(typeof a)
>>> 'object'
```

new String   null

```
var a = new String('123')
var b = new String('123')
console.log(a==b, a===b)
>>> false false
```

a b   null === null true

- 　　　　　　,　　　　.

JS

- 　　　　　　-ES7 Async/Await
- NodeJs　async/await
- 　JavaScript　async/await

Revision #1
Created 20 March 2020 16:06:11 by
Updated 23 March 2020 10:03:24 by