

電腦 網路 網路網路 的 網路網路 網路

網路網路網路 網路網路 網路網路 網路網路網路網路 網路網路 **18** 網路
網路網路 網路網路網路 網路 網路



網路 .網路網路 網路網路 網路網路網路網路 網路網路 的 網路 網路網路 的 的 網路 的 網路網路網路 網路網路網路 網路網路
的 網路網路 網路網路 網路網路 網路 網路 的 網路網路 網路 的 網路 網路網路 網路網路 網路 的 網路 網路網路
網路 網路 網路 網路 的 網路網路網路網路 .網路 網路網路 網路 網路網路 網路 網路網路 網路 網路網路 網路
網路 網路 網路 的 網路網路 網路 網路 .網路 網路 的 網路 的 網路網路 網路網路 網路網路 網路 的 網路網路
網路網路網路 網路 的 網路 網路網路 網路網路 的 網路 網路網路 網路 網路網路 網路 網路網路 網路 網路
的 網路網路 網路 的 網路網路 網路 的 .網路網路 網路 網路網路 的 網路網路 網路 網路網路網路網路 的 網路
網路 ES6 的 網路網路 網路網路 網路 網路 網路 網路 的 網路 的 網路 網路網路網路 網路網路網路 網路
.網路網路

網路網路 **198** 網路網路 «[網路-網路網路-網路網路網路網路](#)»網路 網路網路 網路網路 的 網路 網路 網路
網路網路 [網路-網路](#) 網路 的 的 網路 網路網路 網路 的 網路網路網路 網路網路網路 .網路 網路
.網路

.if...else

```
const x = 20;
let answer;
if (x > 10) {
  answer = 'greater than 10';
} else {
  answer = 'less than 10';
}
```

```
const answer = x > 10 ? 'greater than 10' : 'less than 10';
```

-2

if

```
if (variable1 !== null || variable1 !== undefined || variable1 !== '') {
  let variable2 = variable1;
}
```

```
const variable2 = variable1 || 'new';
```

-3

let

```
let x;
let y;
let z = 3;
```

```
let x, y, z=3;
```

The Java logo, featuring a stylized coffee cup with steam rising from it, followed by the word "java" in a lowercase, sans-serif font. The logo is white and set against a dark background.

☐☐☐☐ ☐☐ ☐☐☐☐☐☐ ☐☐☐☐☐☐ :Java 9 ☐☐☐ ☐☐☐ ☐☐☐☐

[illegible]

□ □ □ □ □ □ □ □ □ □

00 0 0000 00000000 00000000 0000 00 0000 0000 00 00 000000 0000 0000 000000 0000 00
 .000000 000000 lodash 00 00000000 000000 000000 000000000000

□ □ □ □ □ □ □ □ □ □

Array.forEach

□□□□□□ □□ □□□□□□□□ □□□□ □□ □□□□□□ □□□□ □□ □□ □□ □□□□ □□□□

이제 이 코드를 실행하면, process.env.DB_HOST가 undefined가 아니라, process.env.DB_HOST가 localhost로 설정된다는 것을 확인할 수 있습니다.

타입스크립트 -6

이제 이 코드를 실행하면, process.env.DB_HOST가 undefined가 아니라, process.env.DB_HOST가 localhost로 설정된다는 것을 확인할 수 있습니다.

이제 이 코드를 실행하면, process.env.DB_HOST가 undefined가 아니라, process.env.DB_HOST가 localhost로 설정된다는 것을 확인할 수 있습니다.

```
let dbHost;
if (process.env.DB_HOST) {
  dbHost = process.env.DB_HOST;
} else {
  dbHost = 'localhost';
}
```

이제 이 코드를 실행하면, process.env.DB_HOST가 undefined가 아니라, process.env.DB_HOST가 localhost로 설정된다는 것을 확인할 수 있습니다.

```
const dbHost = process.env.DB_HOST || 'localhost';
```

Decimal 타입 -7

이제 이 코드를 실행하면, process.env.DB_HOST가 undefined가 아니라, process.env.DB_HOST가 localhost로 설정된다는 것을 확인할 수 있습니다.

이제 이 코드를 실행하면, process.env.DB_HOST가 undefined가 아니라, process.env.DB_HOST가 localhost로 설정된다는 것을 확인할 수 있습니다.

```
for (let i = 0; i < 10000; i++) {}
```

이제 이 코드를 실행하면, process.env.DB_HOST가 undefined가 아니라, process.env.DB_HOST가 localhost로 설정된다는 것을 확인할 수 있습니다.

```
for (let i = 0; i < 1e7; i++) {}
```

이제 이 코드를 실행하면, process.env.DB_HOST가 undefined가 아니라, process.env.DB_HOST가 localhost로 설정된다는 것을 확인할 수 있습니다.

```
1e0 === 1;
1e1 === 10;
1e2 === 100;
1e3 === 1000;
1e4 === 10000;
1e5 === 100000;
```

이제 이 코드를 실행하면, process.env.DB_HOST가 undefined가 아니라, process.env.DB_HOST가 localhost로 설정된다는 것을 확인할 수 있습니다.



return statement is used to return the value of the function call.

return statement is used to return the value of the function call.

return statement is used to return the value of the function call -8

return statement is used to return the value of the function call. In ES6, the return statement is used to return the value of the function call. The return statement is used to return the value of the function call. The return statement is used to return the value of the function call.

return statement is used to return the value of the function call.

```
const obj = { x:x, y:y };
```

return statement is used to return the value of the function call.

```
const obj = { x, y };
```

Implicit Return statement is used to return the value of the function call -9

return statement is used to return the value of the function call. In ES6, the return statement is used to return the value of the function call. The return statement is used to return the value of the function call. The return statement is used to return the value of the function call.

return statement is used to return the value of the function call.

```
function calcCircumference(diameter) {  
  return Math.PI * diameter  
}
```

return statement is used to return the value of the function call.

```
calcCircumference = diameter => (  
  Math.PI * diameter;  
)
```

return statement is used to return the value of the function call. In ES6, the return statement is used to return the value of the function call. The return statement is used to return the value of the function call. The return statement is used to return the value of the function call.

return statement is used to return the value of the function call -10

return statement is used to return the value of the function call. In ES6, the return statement is used to return the value of the function call. The return statement is used to return the value of the function call. The return statement is used to return the value of the function call.

.function volume(l, w, h) {

function volume(l, w, h) {

```
function volume(l, w, h) {
  if (w === undefined)
    w = 3;
  if (h === undefined)
    h = 4;
  ; return l * w * h
}
```

function volume(l, w, h) {

```
volume = (l, w = 3, h = 4) => (l * w * h);
volume(2) //output: 24
```

function volume(l, w, h) {



function volume(l, w, h) {

function volume(l, w, h) {

Template Literals -11

function volume(l, w, h) {

function volume(l, w, h) {

```
const welcome = `You have logged in as ` + first + ` ` + last + `.`
const db = `http://${host}:${port}/${database}`;
```

function volume(l, w, h) {

```
const welcome = `You have logged in as ${first} ${last}`;
const db = `http://${host}:${port}/${database}`;
```

function volume(l, w, h) { -12

function volume(l, w, h) {

```
const observable = require('mobx/observable');
const action = require('mobx/action');
const runInAction = require('mobx/runInAction');
```

```
const store = this.props.store;
const form = this.props.form;
const loading = this.props.loading;
const errors = this.props.errors;
const entity = this.props.entity;
```

```
import { observable, action, runInAction } from 'mobx';
const { store, form, loading, errors, entity } = this.props;
```

000 000 0000 0000 00000 -13

0000000 n\t\ 000000 0 + 00000 00 0000 000000 00000 0000000 000 000 00000000 00 000 00000000 00 0000

```
const lorem = 'Lorem ipsum dolor sit amet, consectetur\n\t'
  + 'adipisicing elit, sed do eiusmod tempor incididunt\n\t'
  + 'ut labore et dolore magna aliqua.\n\t'
```

```
const lorem = `Lorem ipsum dolor sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.`
```

000000 00000 0000 00000 -14

00 00000 000 00000 0000000000 000000 00 00 .00 000000 ES6 00 000000 000 000000 0000000 000000
00000000 000000 00000000 00000000 000000 0000 .000 0000 000000000 00000000000000000000 000 00 000000000 00000
!0000 00000 00 00 0000000000 000000 000000 .0000

```
// joining arrays
const odd = [1, 3, 5];
const nums = [2 ,4 , 6].concat(odd);

// cloning arrays
const arr = [1, 2, 3, 4];
const arr2 = arr.slice()
```



```
mandatory = () => {
  throw new Error('Missing parameter!');
}

foo = (bar = mandatory()) => {
  return bar;
}
```

上面代码中，mandatory 函数没有参数，foo 函数有一个参数 bar，它的默认值是 mandatory()。由于 mandatory 函数没有参数，调用它的时候，就抛出了一个错误。因此，foo 函数的默认值，就变成了 undefined。

Array.find 方法 -16

ES6 提供了 Array 对象的 find 方法，用于找出符合条件的元素。它的用法类似于 for 循环，但会返回符合条件的第一个元素。

ES6 代码

```
const pets = [
  { type: 'Dog', name: 'Max' },
  { type: 'Cat', name: 'Karl' },
  { type: 'Dog', name: 'Tommy' },
]

function findDog(name) {
  for(let i = 0; i<pets.length; ++i) {
    if(pets[i].type === 'Dog' && pets[i].name === name) {
      return pets[i];
    }
  }
}
```

ES6 代码

```
pet = pets.find(pet => pet.type === 'Dog' && pet.name === 'Tommy');
console.log(pet); // { type: 'Dog', name: 'Tommy' }
```

[key] 方法 -17

ES6 提供了 Object 对象的 key 方法，用于找出符合条件的属性。它的用法类似于 find 方法，但会返回符合条件的第一个属性名。

ES6 代码

```
function validate(values) {
  if(!values.first)
```

```
console.log(validate({first:'Bruce',last:'Wayne'})); // true
```

[illegible][illegible]

0000 0000 00 00000000 0000 00000000 0000 00 000 00 000000 0000000000 0000 00 0000 0000000000
 000000 0 0000 00 0000 000000 000 00 .00000000 00000000 0000 00 000 000 00 00000000 000 00000000
 0000 0000 00000000 0000 00 0000 000 00 0000 000 000000000 0000 00000000 0000 0000 000 00 00 000000
Math.floor() 0000 000000 000000000 000 00 000000000 000 .000 00000000 00000000 **NOT** 000000 000 00 00
 000000 000000000 000000 00 000 00000 000 00000 00 000 000 00 **NOT** 0000 000000 000 00000 .0000 00000000
 .000000

Math.floor(4.9) === 4 //true

~~4.9 === 4 //true

:شماره ۱۱۱
شماره ۱۱۱

:شماره ۱۱۱

Sitepoint

:شماره ۱۱۱

شماره ۱۱۱

شماره ۱۱۱

:شماره ۱۱۱

11:45 - 20/11/1396

:شماره ۱۱۱

- شماره ۱۱۱ - شماره ۱۱۱ - شماره ۱۱۱ - شماره ۱۱۱ - شماره ۱۱۱
شماره ۱۱۱ - شماره ۱۱۱ - شماره ۱۱۱ - شماره ۱۱۱ - شماره ۱۱۱

شماره ۱۱۱

<https://www.shabakeh-mag.com/cover-story/11198/%D8%A7%DB%8C%D9%86-18-%D8%AA%D%A9%D9%86%DB%8C%DA%A9-%DA%A9%D9%88%D8%AA%D8%A7%D9%87%E2%80%8C%D8%B3%D8%A7%D8%B2%DB%8C-%DA%A9%D8%AF%D9%87%D8%A7%DB%8C-%D8%AC%D8%A7%D9%88%D8%A7-%D8%A7%D8%B3%DA%A9%D8%B1%DB%8C%D9%BE%D8%AA%D8%8C-%D8%B4%D9%85%D8%A7-%D8%B1%D8%A7-%D8%B4%DA%AF%D9%81%D8%AA%E2%80%8C%D8%B2%D8%AF%D9%87-%D9%85%DB%8C%E2%80%8C%DA%A9%D9%86%D8%AF>