کوتاه بنویسید تا سریع‌تر اجرا شوند

این 18 تکنیک کوتاه‌سازی کدهای جاوا اسکریپت، شما را شگفت‌زده می‌کند

هیچ‌چیز به‌اندازه کوتاه‌سازی و مهم‌تر از آن شفاف‌سازی کدها در دنیای برنامه‌نویسی اهمیت ندارد. جاوا اسکریپت نیز از این قاعده مستثنا نیست و شما به‌عنوان یک طراح جاوا اسکریپت مجبور هستید از تکنیک‌هایی برای کوتاه کردن کدهای خود استفاده کنید. برنامه‌نویسان در طول دوران کاری خود یاد می‌گیرند که چگونه می‌توانند کدهای طولانی را فشرده و خلاصه کنند. این کار نه‌تنها به درک بهتر کدها کمک می‌کند، بلکه باعث می‌شود تا کدهای بلند کوتاه شوند و همچنین توانایی شما در خلق برنامه‌های منطبق با متدولوژی‌های روز دنیای نرم‌افزار را به‌خوبی نشان می‌دهد. بر همین اساس، در این مقاله به تشریح تکنیک‌هایی پرداخته‌ایم که به شما در کوتاه کردن کدهای جاوا اسکریپت مبتنی بر ES6 کمک می‌کنند.

این مطلب یکی از مقالات پرونده ویژه«جاوااسکریپت آینده است» شماره 198 ماهنامه شبکه است. علاقه‌مندان می‌توانند کل این پرونده ویژه را از روی سایت شبکه دانلود کنند.

1- عملگر سه‌جانبه

این تکنیک به‌شکل قابل توجهی در کدهای شما صرفه‌جویی می‌کند. به‌ویژه زمانی که در نظر دارید از ترکیب دستورات if...else در یک خط استفاده کنید.

کد بلند

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

کد کوتاه شده

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

2- کوتاه کردن ارزیابی‌ها

زمانی که مقدار یک متغیر به متغیر دیگری اختصاص داده می‌شود، باید اطمینان حاصل کنیم که متغیر مبدأ به null

تعریف نشده یا خالی اشاره نکرده باشد.
شما می‌توانید از یک دستور if بلند با چند دستور شرطی یا از یک عبارت ارزیابی کوتاه شده استفاده کنید.

کد بلند

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

کد کوتاه شده

```
const variable2 = variable1  || ‘new’;
```

3- کوتاه کردن تعریف متغیرها

تمرین خوبی است که فرآیند تعریف و مقداردهی متغیرها را در ابتدای یک تابع انجام دهید. این تکنیک به‌میزان قابل توجهی در زمان و فضای شما به‌ویژه زمانی که در نظر دارید چند متغیر از یک نوع را تعریف کنید صرفه‌جویی می‌کند.

کد بلند

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

کد کوتاه شده

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

مطلب پیشنهادی

شرکت اوراکل و برنامه‌های آینده
مسیر پیش روی Java 9: انتشار نسخه‌های شش ماهه

4- کوتاه کردن دستورات ارزیابی

زمانی که یک دستور if را برای یک ارزیابی ساده به کار می‌برید، اپراتورهای تخصیص داده شده را می‌توانید حذف کنید.

کد بلند

```
if (likeJavaScript === true)
```

کد کوتاه شده

```
if (likeJavaScript)
```

5- کوتاه کردن حلقه‌های جاوا اسکریپت

این تکنیک واقعاً مفید است، به‌ویژه اگر در نظر دارید تنها از جاوا اسکریپت استفاده کنید و به کتابخانه‌های خارجی همچون جی‌کوئری یا lodash اعتماد ندارید.

کد بلند

```
for (let i = 0; i < allImgs.length; i++)
```

کد کوتاه شده

```
for (let index of allImgs)
```

کوتاه شده برای Array.forEach

```
function logArrayElements(element, index, array) {
  console.log("a[" + index + "] = " + element);
}
[2, 5, 9].forEach(logArrayElements);
// logs:
// a[0] = 2
// a[1] = 5
// a[2] = 9
```

به‌جای نوشتن شش خط کد برای اختصاص یک مقدار پیش‌فرض، اگر پارامتر مورد نظر صفر یا نامشخص است، این توانایی را داریم تا از یک اپراتور منطقی ساده استفاده و آن چند خط کد را در یک خط خلاصه کنیم

6- کوتاه کردن ارزیابی‌ها

به‌جای نوشتن شش خط کد برای اختصاص یک مقدار پیش‌فرض، اگر پارامتر مورد نظر صفر یا نامشخص است، این توانایی را داریم تا از یک اپراتور منطقی ساده استفاده و آن چند خط کد را در یک خط خلاصه کنیم.

کد بلند

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

کد کوتاه شده

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

7- مقدار پایه Decimal

به‌جای آنکه اعداد را با لشگری از صفرها در یک دستور بنویسید، این قابلیت را در اختیار دارید تا از شکل نمایی استفاده کنید. به طور مثال، 1e7 به‌معنای 1 همراه با 7 صفر است.

کد بلند

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

کد کوتاه شده

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

همه ارزیابی‌های زیر صحیح هستند.

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

مطلب پیشنهادی

یادگیری قدرتمند با اتکا بر رفتار هوشمندانه
برای یادگیری برنامه‌نویسی به این 13 نکته دقت کنید

8- کوتاه کردن فرآیند تخصیص اشیا

تعریف object در جاوا اسکریپت زندگی را بیش از پیش ساده کرده است. ES6 حتی یک راهکار ساده‌تر برای اختصاص خاصیت‌ها به اشیا را پیشنهاد داده است. اگر نام یک خاصیت یکسان با نام کلیدی است، شما می‌توانید از مزیت کوتاه کردن کدها استفاده کنید.

کد بلند

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

کد کوتاه شده

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

9- کوتاه کردن Implicit Return

Return یک کلمه کلیدی است که اغلب در پایان یک تابع از آن استفاده می‌کنیم. یک تابع arrow با یک پارامتر تکی به طور ضمنی نتیجه ارزیابی خود را باز می‌گرداند. برای بازگشت به یک دستور متشکل از چند دستور، ضروری است که از () به‌جای {} در بدنه تابع خود استفاده کنید. برای آنکه بتوان کلمه return را حذف کرد در بدنه تابع نباید از {} استفاده کرد. این تکنیک تضمین می‌کند که کد به‌عنوان یک دستور تکی ارزیابی شده است.

کد بلند

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

کد کوتاه شده

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

از به‌کارگیری عملگر + برای اتصال چند متغیر به یکدیگر داخل یک رشته خسته شده‌اید؟ بهتر نیست از راهکار ساده‌تری برای این منظور استفاده کنید؟

10- مقدار اولیه پارامترها

شما می‌توانید از دستور if برای تعریف مقدار پیش‌فرض برای پارامترهای تابع استفاده کنید. در ES6 شما می‌توانید

مقادیر پیش‌فرض را در خود تعریف تابع مشخص کنید.

کد بلند

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

کد کوتاه شده

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

مطلب پیشنهادی

موتور اجرای برنامه گوگل
برنامه‌نویسی را بر فراز ابرها تجربه کنید

Template Literals -11

از به‌کارگیری عملگر + برای اتصال چند متغیر به یکدیگر داخل یک رشته خسته شده‌اید؟ بهتر نیست از راهکار ساده‌تری برای این منظور استفاده کنید؟ اگر می‌توانید از ES6 استفاده کنید، قادر هستید از ${} برای محصور کردن و خلاصه‌سازی کدهای خود استفاده کنید.

کد بلند

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

کد کوتاه شده

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

12- انتساب کوتاه شده

اگر با هریک از چهارچوب‌های محبوب وب در حال کار هستید، شانس زیادی وجود دارد که از آرایه‌ها یا داده‌ها در قالب شکل خاصی به‌منظور ارسال اطلاعات به مؤلفه‌ها یا API استفاده کنید. هنگامی که یک شیء داده‌ای به یک مؤلفه می‌رسد، در ادامه مجبور خواهید بود آن را باز کنید.

کد بلند

```
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;
```

کد کوتاه شده

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

13- کوتاه کردن رشته چند خطی

اگر در کدنویسی خود با رشته‌های چند خطی روبه‌رو شدید، نیازی نیست از عملگر + و جفت‌های \n\t استفاده کنید.

کد بلند

```
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'
```

کد کوتاه شده

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

14- کوتاه کردن عملگر گسترشی

عملگر گسترشی اولین بار همراه با ES6 معرفی شد. تا به امروز کاربردهای مختلفی برای این عملگر در جاوا اسکریپت از سوی توسعه‌دهندگان شناسایی شده است. این عملگر می‌تواند جایگزین توابع آرایه‌ای شود. عملگر گسترش مجموعه‌ای از سه نقطه است!

کد بلند

```
// 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()
```

کد کوتاه شده

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

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

برعکس تابع ()concat، شما می‌توانید از عملگر گسترشی برای اضافه کردن یک آرایه به هر مکانی درون آرایه دیگری استفاده کنید.

```
const odd = [1, 3, 5 ];
const nums = [2, ...odd, 4 , 6];
```

شما همچنین می‌توانید عملگر گسترشی را با انتساب‌ها در ES6 ترکیب کنید.

```
const { a, b, ...z } = { a: 1, b: 2, c: 3, d: 4 };
console.log(a) // 1
console.log(b) // 2
console.log(z) // { c: 3, d: 4 }
```

مطلب پیشنهادی

این زبان‌ها به شما در درک بهتر برنامه‌نویسی کمک می‌کنند
ساده‌ترین زبان‌های برنامه‌نویسی ویژه افراد تازه‌کار

15- کوتاه کردن پارامتر اجباری

به طور پیش‌فرض، اگر هیچ پارامتری برای یک تابع مشخص نشده باشد، جاوا اسکریپت پارامترهای تابع را به undefined تنظیم می‌کند. زبان‌های دیگر یک خطا یا هشدار را تولید می‌کنند. برای اطمینان از اختصاص پارامتر به یک تابع می‌توانید اگر undefined تعریف شده است، از دستور if برای تولید یک خطا یا از مزیت کوتاه کردن پارامتر اجباری استفاده کنید.

کد بلند

```
function foo(bar) {
  if(bar === undefined) {
    throw new Error('Missing parameter!');
  }
  return bar;
}
```

کد کوتاه شده

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

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

به طور پیش‌فرض، اگر هیچ پارامتری برای یک تابع مشخص نشده باشد، جاوا اسکریپت پارامترهای تابع را به undefined تنظیم می‌کند. زبان‌های دیگر یک خطا یا هشدار را تولید می‌کنند

16- کوتاه کردن Array.find

اگر در زمان نوشتن پروژه‌ها نیاز داشتید تا یک تابع find را در جاوا اسکریپت بنویسید، احتمالاً از یک حلقه for استفاده می‌کردید. در ES6 تابع جدیدی به‌نام find معرفی شد.

کد بلند

```
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];
    }
  }
}
```

کد کوتاه شده

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

17- کوتاه کردن شیء [key]

از این موضوع اطلاع داشتید که Foo.bar را می‌توان به‌صورت Foo[‘bar’] نیز نوشت؟ در ابتدا، دلیلی پیدا نمی‌کنید که چرا باید به این شکل بنویسید. با این حال، این تکنیک به شما در ساخت بلوک‌هایی از کدها که قابلیت استفاده مجدد دارند کمک می‌کند. مثال زیر یک تابع اعتبارسنجی ساده را به شما نشان می‌دهد.

کد بلند

```
function validate(values) {
  if(!values.first)
    return false;
  if(!values.last)
    return false;
  return true;
}

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

کد کوتاه شده

این تابع کار خود را به‌شکل کاملی انجام می‌دهد. با این حال، یک سناریو را در نظر بگیرید که فرم‌های زیادی دارید که باید فرآیند اعتبارسنجی را در آن‌ها انجام دهید، اما این کار با فیلدها و قواعد مختلفی باید انجام شود. کار جالبی نخواهد بود که یک تابع اعتبارسنجی عمومی که بتوان در زمان اجرا آن را پیکربندی کرد را نوشت؟

```
// object validation rules
const schema = {
  first: {
    required:true
```

```
  },
  last: {
    required:true
  }
}
// universal validation function
const validate = (schema, values) => {
  for(field in schema) {
    if(schema[field].required) {
      if(!values[field]) {
        return false;
      }
    }
  }
  return true;
}
console.log(validate(schema, {first:'Bruce'})); // false
console.log(validate(schema, {first:'Bruce',last:'Wayne'})); // true
```

عملگرهای بیتی از جمله ویژگی‌هایی هستند که شما در آغاز یادگیری جاوا اسکریپت با آن‌ها آشنا می‌شوید، اما تقریباً در هیچ کجا از آن‌ها استفاده نمی‌کنید

18- کوتاه کردن جفت عملگر بیتی NOT

عملگرهای بیتی از جمله ویژگی‌هایی هستند که شما در آغاز یادگیری جاوا اسکریپت با آن‌ها آشنا می‌شوید، اما تقریباً در هیچ کجا از آن‌ها استفاده نمی‌کنید. چه کسی تمایل دارد با یک‌ها و صفرها زمانی که در هیچ کجا مورد استفاده قرار نمی‌گیرند کار کند؟ با این حال، یک مورد کاربردی وجود دارد که از جفت عملگر NOT می‌توان استفاده کرد. شما می‌توانید از این اپراتور به‌جای تابع Math.floor() استفاده کنید. مزیت جفت عملگر بیتی NOT در این است که همان کار تابع فوق را به‌شکل سریع‌تری انجام می‌دهد.

کد بلند

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

کد کوتاه شده

```
~~4.9 === 4   //true
```

منبع:

Sitepoint

تاریخ انتشار:
20 بهمن 1396

نشانی منبع:
https://www.shabakeh-mag.com/workshop/programming/11198/%D8%A7%DB%8C%D9%86-18-%D8%AA%DA%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