Nguyễn Văn Quy @ruacondepzaj Follow
★ 2.4K 84 30

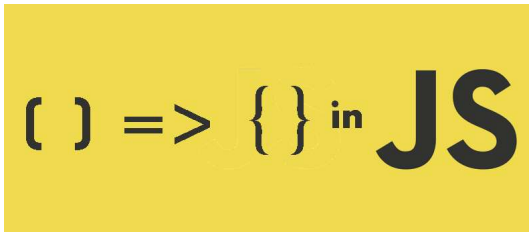
Published Aug 15th, 2018 8:39 AM - 5 min read
808 2 1

Thế nào là Arrow Function? Cách thức sử dụng Arrow Function và những điều cần chú ý

JavaScript arrow function

1. Giới thiệu

`Arrow functions` là một trong những tính năng mới rất hay của ES6. Việc sử dụng đúng cách arrow function giúp code trở nên ngắn gọn và dễ hiểu hơn. Vì vậy, trong bài viết này mình sẽ giúp bạn hiểu rõ hơn về arrow function trong JavaScript, cũng như biết cách sử dụng và những điều cần lưu ý khi sử dụng nó.



`Arrow function` được định nghĩa là:

> Arrow functions – also called “fat arrow” functions, from CoffeeScript (a transcompiled language) — are a more concise syntax for writing function expressions.

2. Cách sử dụng

Cú pháp cơ bản với nhiều tham số

```
// ES5
var multiTestEs5 = function(a, b) {
  return a * b;
};

// ES6
const multiTestEs6 = (a, b) => { return a * b };
```

Cú pháp cơ bản với một tham số

```
//ES5
var phraseSplitterEs5 = function phraseSplitter(phrase) {
  return phrase.split(' ');
};

//ES6
const phraseSplitterEs6 = phrase => phrase.split(" ");

console.log(phraseSplitterEs6("Framgia AwesomeTrip"));  // ["Framgia", "AwesomeTrip"]
```

`Arrow functions` đúng như những gì mà người ta nói về nó. Sử dụng `Arrow functions` giải quyết bài toán với một tham số, code khá là gọn, đơn giản và dễ hiểu.

Không có tham số

```
// ES5
var docLogEs5 = function docLog() {
    console.log(document);
};

// ES6
var docLogEs6 = () => { console.log(document); };
docLogEs6();
```

Cú pháp Object literals

```
//ES5
var setNameIdsEs5 = function setNameIds(id, name) {
  return {
    id: id,
    name: name
  };
};

// ES6
var setNameIdsEs6 = (id, name) => ({ id: id, name: name });

console.log(setNameIdsEs6 (4, "Van Quy"));   // Object {id: 4, name: "Van Quy"}
```

Promises and Callbacks

```
// ES5
aAsync().then(function() {
  returnbAsync();
}).then(function() {
  returncAsync();
}).done(function() {
  finish();
});

// ES6
aAsync().then(() => bAsync()).then(() => cAsync()).done(() => finish);
```

Có thể thấy rõ với Promises and Callbacks khi sử dụng arrow function, thay vì phải viết quá nhiều `function()` giờ chỉ cần viết `() =>` là đủ rồi =))

3. Những điều cần chú ý khi sử dụng arrow function

Qua các cách sử dụng ở trên chúng ta thấy được `Arrow functions` ngắn gọn, dễ hiểu. Tuy nhiên, không phải lúc nào cũng sử dụng tùy tiện được. Dưới đây là một vài trường hợp không nên sử dụng `Arrow functions` .

- **Click handle**

Đầu tiên. mình có một button lớn, có thể đặt tên cho nó là "push me":

```
<button id="pushy">Push me</button>

<style>
    button { font-size: 100px; }
    .on { background: #ffc600; }
</style>
```

Khi ai đó `click` vào nút, mình muốn nút này chuyển (thay đổi) class thành `on` và nút sẽ chuyển sang màu vàng.

```
const button = document.querySelector('#pushy');
button.addEventListener('click', () => {
    this.classList.toggle('on');
});
```

Nhưng nếu bạn `click` vào nút này thì bạn sẽ nhận được lỗi trả về là:

```
Uncaught TypeError: Cannot read property 'toggle' of undefined
```

Vậy điều này có nghĩa là gì? `this` trong trường hợp này, nó là thuộc tính `window` của trình duyệt. Bạn có thể sử dụng `console.log` để confirm nó.

```
const button = document.querySelector('#pushy');
button.addEventListener('click', () => {
    console.log(this); // Window!
    this.classList.toggle('on');
});
```

Hãy nhớ rằng: Chúng ta đang nói về việc sử dụng `Arrow functions` , vậy nên `this` không bị ràng buộc bởi button này. Ngược lại, nếu chúng ta sử dụng function thông thường, `this` sẽ bị ràng buộc vào button mà bạn `click` vào đó.

```
const button = document.querySelector('#pushy');
button.addEventListener('click', function() {
    console.log(this);
    this.classList.toggle('on');
});
```

Hiển nhiên khi `console.log(this)` kết quả trả về chính là button mà chúng ta vừa `click` , và button ban đầu sẽ chuyển sang màu vàng như ý muốn.

- **Object Methods**

Kết thúc ví dụ về Click handle ở trên, tiếp đến chúng ta sẽ đi làm một ví dụ về object methods. Trước tiên, chúng ta cần một method để `bind` (ràng buộc) một object.

```
const person = {
    points: 23,
    score: () => {
        this.points++;
    }
}
```

Giả sử chúng ta có một object là `person` . Chúng ta có một method được gọi là `score` và có thể gọi nó ra bằng `person.score` . `score` có sẽ tăng dần `points` tùy vào số lần nó được `run` và `points` hiện tại là 23.

Khi sử dụng arrow function dù có run `person.score` bao nhiêu lần đi chăng nữa, `points` vẫn chỉ dừng lại ở con số 23.

Tại sao vậy???

Khi sử dụng arrow function `this` không bị ràng buộc bởi thứ gì cả. Nó chỉ kế thừa từ phạm vi cha mà trong trường hợp này là `window` .

Và khi dùng function thông thường nếu chúng ta chạy `person.score();` , trong một vài lần, `points` sẽ có giá trị > 23.

Bởi vậy chúng ta không nên dùng arrow function trong trường hợp này:

```
const person = {
    points: 23,
    score: function()  {
        this.points++;
    }
}
```

- **Prototype Methods**

Sau khi dừng lại 2 ví dụ ở trên, chắc rằng chúng ta cũng đã có thêm được những lưu ý cần thiết khi sử dụng arrow function rồi đúng không nhỉ! Tại ví dụ tiếp theo, hãy cùng tìm hiểu về `prototype methods` để có thêm cái nhìn rõ nét hơn.

```
class Car {
    constructor(make, colour) {
        this.make = make;
        this.colour = colour;
    }
}
```

Ở ví dụ này, chúng ta khởi tạo một `class` là car. Và để thêm các `new Car` ta sử dụng:

```
const beemer = new Car('BMW', 'blue');
const subie = new Car('Subaru', 'white');
```

Dễ dàng nhận thấy rằng đã có `subie` là một Car {make: "Subaru", colour: "white"}, và beemer là một Car {make: "BMW", colour: "blue"}. Sau đó thử thêm một prototype method:

```
Car.prototype.summarize = () => {
    return `This car is a ${this.make} in the colour ${this.colour}`;
};
```

Sau tất cả những gì chúng ta đã tạo ra, chúng ta có thể thêm được tất cả các methods vào tất cả chúng. Bởi vậy `Car.prototype.summarize` được thiết lập, vì vậy hãy viết vào console `subie.summarize`

Trường hợp này khi sử dụng arrow function `this.car` trả về `undefined` và `this.colour` cũng trả về `undefined`.

Bạn phải sử dụng `this` trong một function thông thường:

```
Car.prototype.summarize = function() {
    return `This car is a ${this.make} in the colour ${this.colour}`;
};
```

Kết luận

- `Arrow functions` với cú pháp rút gọn, dễ hiểu là sự chọn lựa thông minh cho coder.
- `Arrow functions` không ràng buộc `this`.

Tham khảo

- ES6 Arrow Functions.
- ES6 Arrow Functions: Fat and Concise Syntax in JavaScript.
- Arrow-function-no-no.

Related

Destructuring Assignment in ES6
Nguyễn Ngọc Vịnh
11 min read
1908 11 0 14

"this" trong JavaScript
manhhomienbienthuy
16 min read
3460 10 3 14

Javascript prototype
Nguyen Tuan Linh
5 min read
477 3 5 6

Modern JavaScript Cheatsheet (Part 1)
Nguyen Duc Tung
14 min read
177 2 0 3

More from Nguyễn Văn Quy

Đa ngôn ngữ cho website có sử dụng Nuxtjs
Nguyễn Văn Quy
4 min read
39 0 0 6

Một số tips nhỏ khi sử dụng nuxtjs
Nguyễn Văn Quy
6 min read
62 1 0 4

Một số tips nhỏ khi sử dụng vuejs
Nguyễn Văn Quy
5 min read

Tìm hiểu về Notifications trong Laravel
Nguyễn Văn Quy
4 min read

+10

Comments

Login to comment

Truong Dang @xdangminhtruongx Aug 15th, 2018 8:55 AM

(bantim)

+2 | Reply Share

Nguyễn Văn Quy @ruacondepzaj Aug 15th, 2018 8:56 AM

(danhnhau)

+1 | Reply Share