



## 一篇深度剖析DOM元素真实面目



jCodeLife

0.119

2020.08.15 06:21:43 字数 1,558 阅读 47

编辑文章

我理解的DOM文档对象模型，说白了就是一系列用于表示文档的东西，这些东西本质是对象，这些对象表示了文档的各个不同部位。

例如DOM中DOM对象也叫DOM节点，节点有分document节点、元素节点、属性节点、文本节点、注释节点等，分别表示文档中的整个文档、某个元素、某个元素的属性、某个元素里面的文本、某个元素里面的注释等。

节点对象中定义了一系列的属性和方法，这些属性和方法跟我们平时自己写的js并没有什么不同，只不过这一系列的接口由浏览器厂商定义，用于更好、更方便的表示和操作文档。

而DOM本质就是这一系列对象的继承关系，由于这继承关系复杂，导致我们不好理解。特写这篇文章梳理一下，看看DOM里面到底有什么鬼！

那从哪里开始看呢？

没有思路的话，就以我们熟悉的document为例子吧

document对象代表整个HTML文档的，我们看看它的原型及它是谁构造出来的

```
> document.__proto__
< HTMLDocument {Symbol(Symbol.toStringTag): "HTMLDocument"}
> document.__proto__.constructor
< function HTMLDocument() { [native code] }
```

document的构造函数的原型

document的构造函数

可以看到document对象是由构造函数HTMLDocument(){}构造出来的，该构造函数的原型是HTMLDocument{}

document可以继承其原型HTMLDocument{}上的属性和方法

不妨一探究竟，看看原型里面到底定义里什么，看有没有我们眼熟的。

### 推荐阅读

Vue干货小技巧——学会再也不做加班狗

阅读 1,314

学会这6个强大的CSS选择器，将真正帮你写出干净的CSS代码！

阅读 258

vue优化页面

阅读 818

想成为一个好的前端，这篇文章你必须看

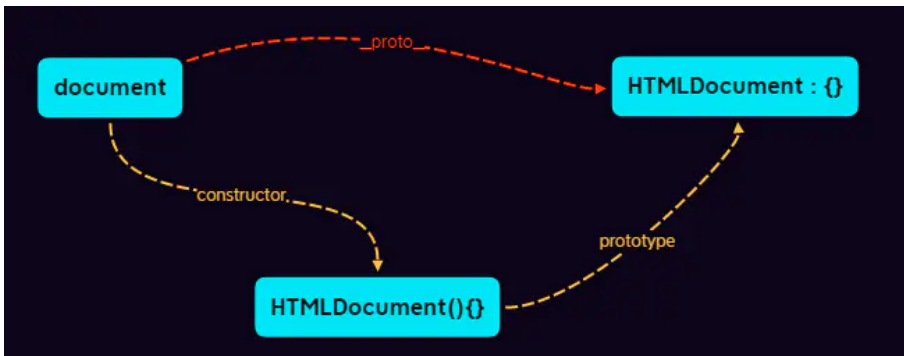
阅读 128

Vue-Router 原理

阅读 241

发现HTMLDocument.prototype上居然定义了多少属性和方法，我画了一下上篇文章有讲到的一些用于遍历节点的方法。

看看目前的关系：



接着，我们来看看

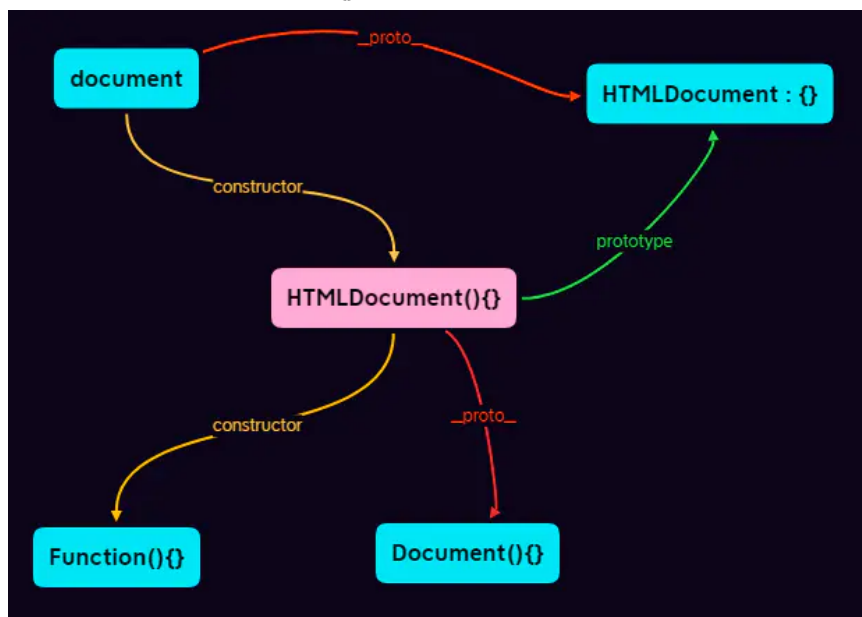
1. 构造函数HTMLDocument()
2. 原型对象HTMLDocument: {}

首先，我们来看看构造函数HTMLDocument()的原型和构造函数

```
> HTMLDocument.constructor
< function Function() { [native code] }
> HTMLDocument.__proto__
< function Document() { [native code] }
```

发现

1. HTMLDocument()的构造函数Function()，我们知道所有方法都是都new Function产生的，所以这里已经到尽头，无需说再往下看Function()
2. HTMLDocument()作为对象时，它的原型\_\_proto\_\_是Document()；而它作为function时，原型指向是HTMLDocument: {}，这里别搞混了。



接着来看，原型对象HTMLDocument{}即document.\_\_proto\_\_

## 推荐阅读

Vue干货小技巧——学会再也不做加班狗

阅读 1,314

学会这6个强大的CSS选择器，将真正帮你写出干净的CSS代码！

阅读 258

vue优化页面

阅读 818

想成为一个好的前端，这篇文章你必须看

阅读 128

Vue-Router 原理

阅读 241

发现

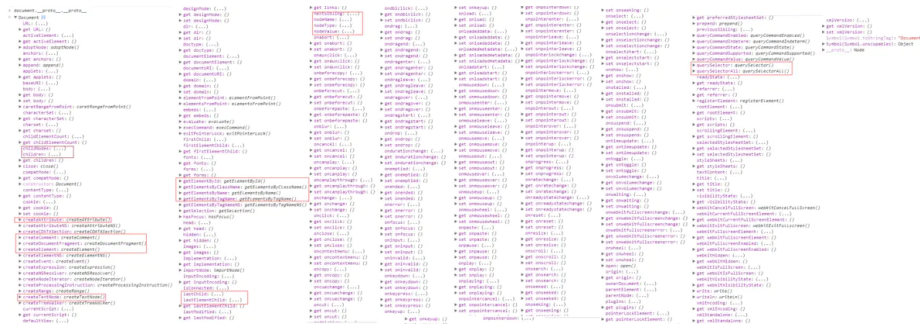
`document.__proto__` 的构造函数是 `HTMLDocument()`{}; 这就是上面一直说的那个 `HTMLDocument()`{}。

这点其实很容易理解，构造函数上有个 `prototype` 属性指向原型，原型有个属性 `constructor` 指向该构造函数

还发现：

`document.__proto__` 往上，即：`document.__proto__.__proto__` 是继承自 `Document()`{}

我们在来看看这个原型对象里面有什么鬼：



我去，这都是什么鬼

有空可以点开认真看看，其实我想主要表达的是：为什么 `document` 能使用 `getElementsByClassName` 这些东西，其实就是因为它顺着它的原型链往上找，找到这里定义的 `getElementsByClassName` 方法。

我在这里小结一下吧：

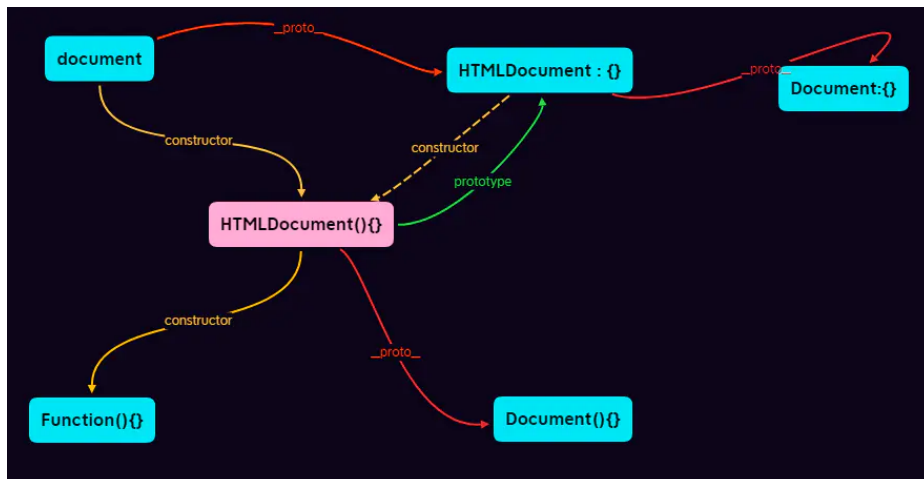
1. `document` 并非由 `Document()`{}，而是由 `HTMLDocument()`{} 构造
2. `document` 的原型是 `HTMLDocument: {}`，这个 `HTMLDocument` 对象里面定义了很多方法和属性，如：

`children`、`childNodes`、`lastChild`、`lastElementChild`、`nextSibling`、`nodeName`、`nodeType`、`nodeValue`、`parentElement`、`parentNode` 等等

3. `document` 的原型的原型是 `Document: {}`。这个大的 `Document` 上方法和属性巨多，忍不住想说几个方法：

`createElement()`、`createTextNode()`、`createAttribute()`、`getElementById()`、`getElementsByClassName()`、`getElementsByName()`、`getElementsByTagName()`、`querySelector()`、`querySelectorAll()` 等等...

到这里可以在看下我们的图，捋一下他们的关系：



看到这里有心细的同学肯定会想 `Document: {}` 和 `Document(){}` 是不是有什么关系呢？

## 推荐阅读

[Vue干货小技巧——学会再也不做加班狗](#)

阅读 1,314

[学会这6个强大的CSS选择器，将真正帮你写出干净的CSS代码！](#)

阅读 258

[vue优化页面](#)

阅读 818

[想成为一个好的前端，这篇文章你必须看](#)

阅读 128

[Vue-Router 原理](#)

阅读 241

写下你的评论...

评论 0

赞 3

...

我再来看Document: {}的原型是谁

为了足够深入，我们还是得看看它指向的Node: {}里面又有些什么（别怕麻烦，一次搞清楚了，下次再也不怕了）

```

lastChild: (...)
▶ get lastChild: ()
▶ lookupNamespaceURI: LookupNamespaceURI()
▶ lookupPrefix: LookupPrefix()
nextSibling: (...)
▶ get nextSibling: ()
nodeName: (...)
▶ get nodeName: ()
nodeType: (...)
▶ get nodeType: ()
nodeValue: (...)
▶ get nodeValue: ()
▶ set nodeValue: ()
▶ normalize: normalize()
ownerDocument: (...)
▶ get ownerDocument: ()
parentElement: (...)
▶ get parentElement: ()
parentNode: (...)
▶ get parentNode: ()
previousSibling: (...)
▶ get previousSibling: ()
▶ removeChild: removeChild()
▶ replaceChild: replaceChild()
textContent: (...)
▶ get textContent: ()
▶ set textContent: ()
Symbol(Symbol.toStringTag): "Node"
__proto__: EventTarget

```

为了不乱，先画下图吧



```
▼ EventTarget ⓘ
  ▶ addEventListener: addEventListener()
  ▶ constructor: EventTarget()
  ▶ dispatchEvent: dispatchEvent()
  ▶ removeEventListener: removeEventListener()
  Symbol(Symbol.toStringTag): "EventTarget"
  ▶ __proto__: Object
```

这下开心了，终于看到了想看到的东西，就是我们最最熟悉的Object了

另外还发现原来给元素添加和移除事件的方法 `addEventListener()`、`removeEventListener()` 也定义在 `EventTarget: {}` 里头了

试试Object上面还有吗？

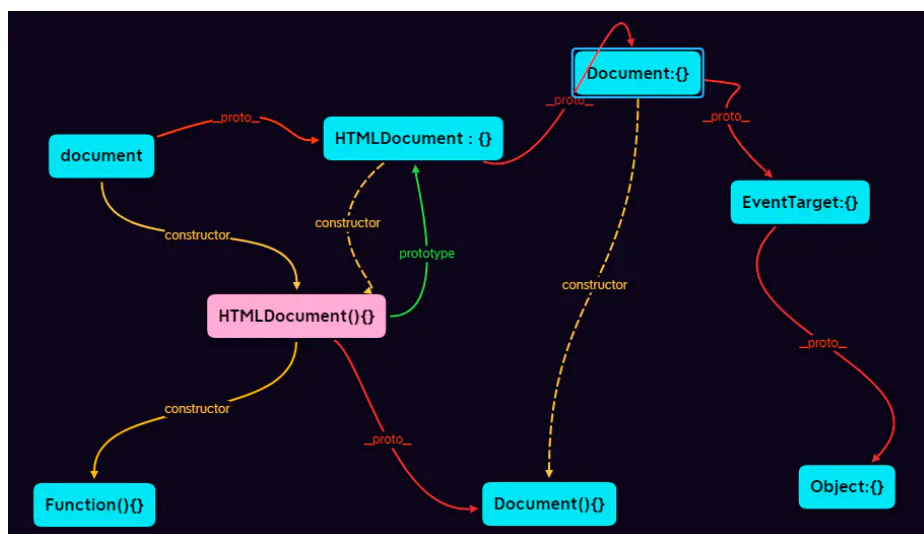
```
▼ Object ⓘ
  ▶ __defineGetter__: __defineGetter__()
  ▶ __defineSetter__: __defineSetter__()
  ▶ __lookupGetter__: __lookupGetter__()
  ▶ __lookupSetter__: __lookupSetter__()
  ▶ constructor: Object()
  ▶ hasOwnProperty: hasOwnProperty()
  ▶ isPrototypeOf: isPrototypeOf()
  ▶ propertyIsEnumerable: propertyIsEnumerable()
  ▶ toLocaleString: toLocaleString()
  ▶ toString: toString()
  ▶ valueOf: valueOf()
  ▶ get __proto__: __proto__()
  ▶ set __proto__: __proto__()
```

Object就到顶了，再上就会返回null。那这里就不用再看了。

```
> document.__proto__
< ▶ HTMLDocument {Symbol(Symbol.toStringTag): "HTMLDocument"}
> document.__proto__.__proto__
< ▶ Document {Symbol(Symbol.toStringTag): "Document", Symbol(Symbol.unscopables): Object}
> document.__proto__.__proto__.__proto__
< ▶ Node {ELEMENT_NODE: 1, ATTRIBUTE_NODE: 2, TEXT_NODE: 3, CDATA_SECTION_NODE: 4, ENTITY_REFERENCE_NODE: 5...}
> document.__proto__.__proto__.__proto__.__proto__
< ▶ EventTarget {Symbol(Symbol.toStringTag): "EventTarget"}
> document.__proto__.__proto__.__proto__.__proto__.__proto__
< ▶ Object {}
> document.__proto__.__proto__.__proto__.__proto__.__proto__.__proto__
< null
```

document的原型链

接着我们回到图看下先



## 推荐阅读

Vue干货小技巧——学会再也不做加班狗

阅读 1,314

学会这6个强大的CSS选择器，将真正帮你写出干净的CSS代码！

阅读 258

vue优化页面

阅读 818

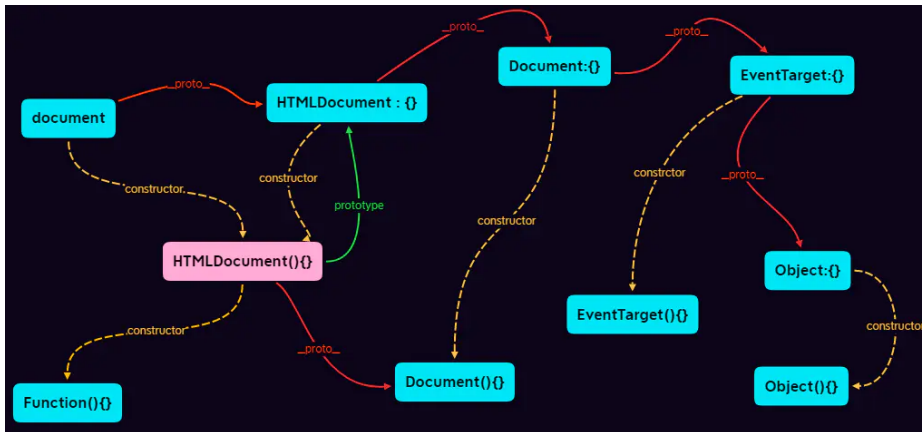
想成为一个好的前端，这篇文章你必须看

阅读 128

Vue-Router 原理

阅读 241

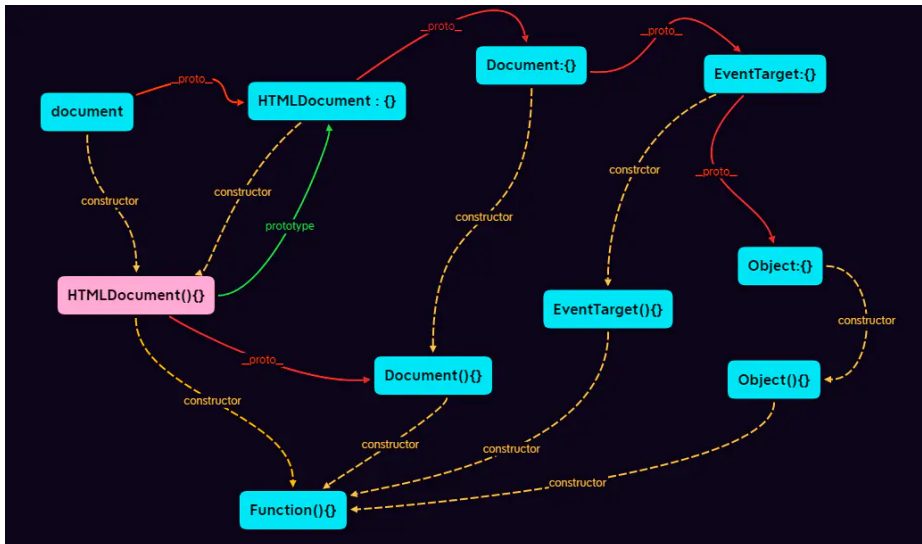
发现EventTarget:{}和Object:{}分别是由对应的构造函数EventTarget(){}和Object(){}构造出来的。



接着在来看Document(){}、EventTarget(){}、Object(){}是谁构建出来的

```
> document.__proto__.__proto__.__proto__.__proto__.__proto__.constructor.constructor
< function Function() { [native code] }
> document.__proto__.__proto__.__proto__.__proto__.__proto__.constructor.constructor
< function Function() { [native code] }
> document.__proto__.__proto__.constructor
< function Document() { [native code] }
> document.__proto__.__proto__.constructor.constructor
< function Function() { [native code] }
```

发现所有的function都是由function Function(){}构建出来的



那Function的构造函数呢？也会返回function Function(){}!!!

js中一切皆是对象，我们看看Function(){}作为对象时，它的原型指向是谁

```
> Function.__proto__
< function () {}
> Function.__proto__.__proto__
< Object {}
> Function.__proto__.__proto__.__proto__
< null
```

## 推荐阅读

Vue干货小技巧——学会再也不做加班狗

阅读 1,314

学会这6个强大的CSS选择器，将真正帮你写出干净的CSS代码！

阅读 258

vue优化页面

阅读 818

想成为一个好的前端，这篇文章你必须看

阅读 128

Vue-Router 原理

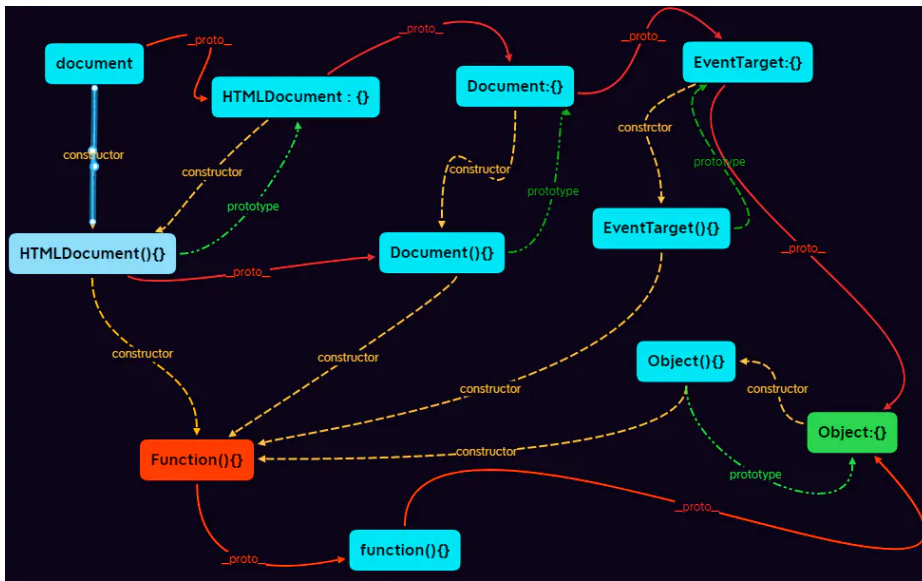
阅读 241



两进是别的对象\_\_proto\_\_指向我和构造函数prototype指向我;

两出是我的属性constructor和\_\_proto\_\_

先完善这点



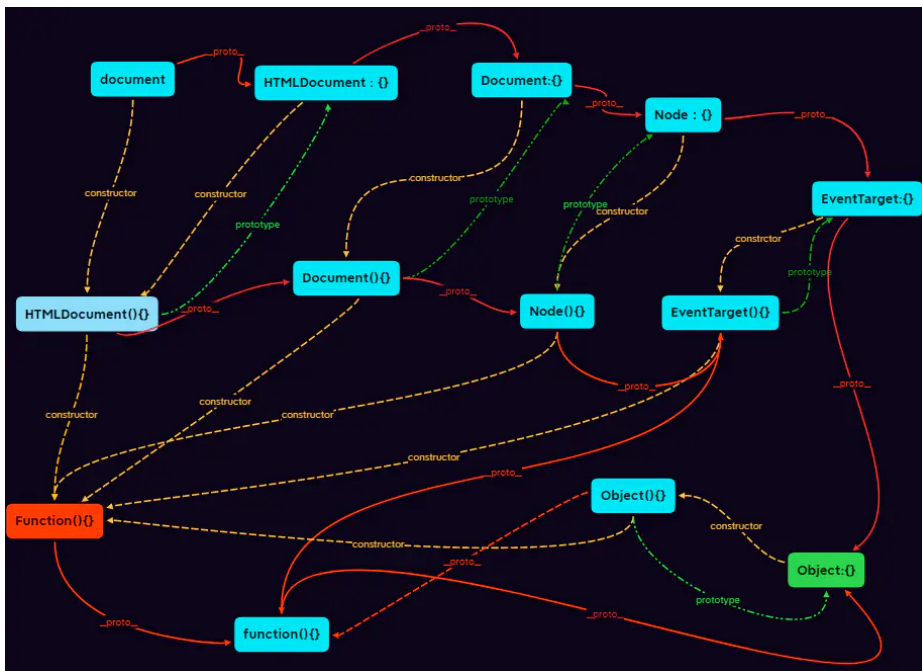
再看，一般的方法是两进三出：

两进是我生产的对象的`constructor`指向我，以及我的原型的`constructor`指向我。(这里大部分没讲到自己生产的对象，所以只有一个原型指向我)

三出是：1 我的`constructor`指向`Function`，2我作为`function`我的`prototype`指向谁，3我作为对象时我的 `__proto__` 指向谁

规律非绝对！

根据上面说的规则再来完善，另外发现上面漏了一个`Node: {}`,现在补上，对不住大家了哈：



我们发现：

基本所有的对象都是通过构造函数产生的，而所有的构造函数都是`Function`产生的

基本所有对象的原型都是最终继承自`Object`。

## 推荐阅读

Vue干货小技巧——学会再也不做加班狗

阅读 1,314

学会这6个强大的CSS选择器，将真正帮你写出干净的CSS代码！

阅读 258

vue优化页面

阅读 818

想成为一个好的前端，这篇文章你必须看

阅读 128

Vue-Router 原理

阅读 241

写下你的评论...

评论0

赞3

...

# 一篇深度剖析DOM元素真实面目

更加深入了解到其里面本来面目，让学习学的更加透彻！！

 3人点赞 >




 笔记 (1)



"小礼物走一走，来简书关注我"

赞赏支持


还没有人赞赏，支持一下



jCodeLife

书山有路勤为径，学海无涯苦作舟

总资产14 (约0.99元) 共写了12.3W字 获得231个赞 共23个粉丝



写下你的评论...

全部评论 0

只看作者

关闭评论

按时间倒序

按时间正序

被以下专题收入，发现更多相似内容

 投稿管理

+ 收入我的专题

 领跑计划笔记

## 推荐阅读

Vue干货小技巧——学会再也不做加班狗

阅读 1,314

学会这6个强大的CSS选择器，将真正帮你写出干净的CSS代码！

阅读 258

vue优化页面

阅读 818

想成为一个好的前端，这篇文章你必须看

阅读 128

Vue-Router 原理

阅读 241

