<el-input  
 v-model.trim="projectInfo.DmsProjectNo"  
 :maxlength="10"  
 clearable  
 readonly  
 @click.native="clearValue(***$event***)"  
 :placeholder="computedTip"  
 @change="dmsProjectNoChange"  
 >  
 <i v-if="!isBrowse" slot="suffix" class="el-icon-circle-close cursor-pointer"></i>  
 <el-button v-if="!isBrowse" type="primary" slot="append" size="mini" @click="chooseProject">选择</el-button>  
</el-input>

clearValue(e) {  
 if(e.target.className.indexOf('el-icon-circle-close') !== -1) {  
 this.projectInfo.DmsProjectNo = ''  
 }  
}

target定义：

target 属性：触发事件的对象的引用（某个dom元素）。当事件处理程序在事件的冒泡或捕获阶段被调用。

target 事件属性可返回事件的目标节点（触发该事件的节点），如生成事件的元素、文档或窗口。

语法：

event.target

event.target.nodeName 　　//获取事件触发元素标签名（li，p，div，img，button…）

event.target.id　　　　　　//获取事件触发元素id

event.target.className　　//获取事件触发元素classname

event.target.innerHTML　 //获取事件触发元素的内容（li）

示例：

// Make a list

var ul = document.createElement('ul');

document.body.appendChild(ul);

var li1 = document.createElement('li');

var li2 = document.createElement('li');

ul.appendChild(li1);

ul.appendChild(li2);

function hide(e){

// e.target 引用着 <li> 元素

// 不像 e.currentTarget 引用着其父级的 <ul> 元素.

e.target.style.visibility = 'hidden';

}

// 添加监听事件到列表，当每个 <li> 被点击的时候都会触发。

ul.addEventListener('click', hide, false);

jquery:this 与 e.target区别:

js中事件是会冒泡的，故this会冒泡而 e.target 不冒泡，它就是指向事件触发的dom。所以this是可以变化的，但event.target不会变化，它永远是直接接受事件的目标DOM元素。

————————————————

版权声明：本文为CSDN博主「bellediao」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/bellediao/article/details/107370438>

Vue获取当前点击的元素、子元素、父元素、上一个元素、下一个元素、兄弟元素

1.想要获取当前点击的元素，以及父元素、子元素、兄弟元素，第一步我们需要先绑定事件

例如： <li class="del" @click="del($event)">删除</li>

//当前点击的元素

e.target

//绑定事件的元素

e.currentTarget

//获得点击元素的前一个元素

e.currentTarget.previousElementSibling.innerHTML

//获得点击元素的第一个子元素

e.currentTarget.firstElementChild

//获得点击元素的下一个元素

e.currentTarget.nextElementSibling

//获得点击元素中id为string的元素

e.currentTarget.getElementById("string")

//获得点击元素的class属性

e.currentTarget.getAttributeNode('class')

// 获得点击元素的父级元素

e.currentTarget.parentElement

// 获得点击元素的前一个元素的第一个子元素的HTML值

e.currentTarget.previousElementSibling.firstElementChild.innerHTM

————————————————

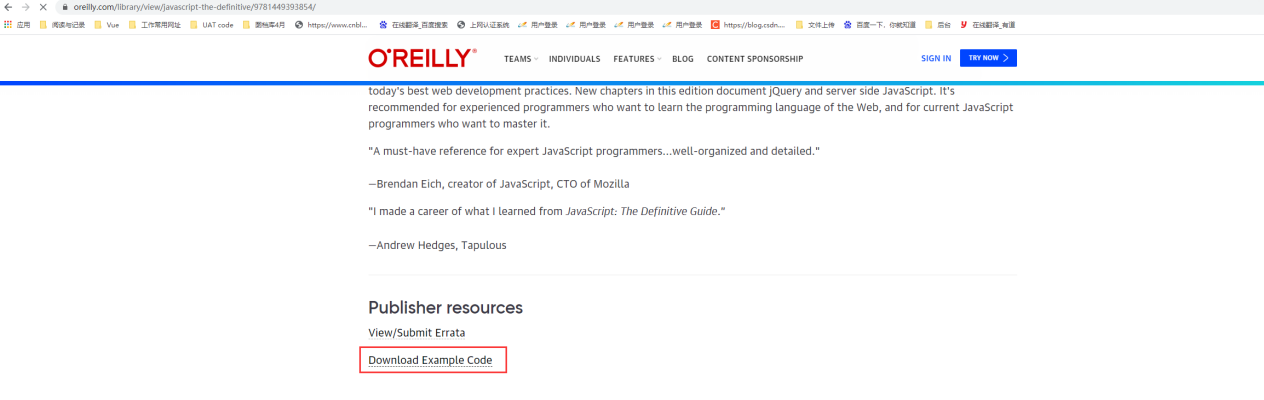
版权声明：本文为CSDN博主「半兽先生」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_43770056/article/details/123512950

<https://gitmind.cn/app/doc/07900b0c428c1ecbf1041e1891e48050>

示例代码

<http://oreilly.com/catalog/9780596805531/>



本书要讲述的内容涵盖JavaScript语言本身，以及Web浏览器所实现的JavaScriptAPI

对**JavaScript语言和Web平台**的理解和掌握再上一个台阶

JavaScript这门语言以及**运行它的各种平台环境**

第一部分主要讲述JavaScript这门语言。

核心章节都涵盖全新的内容（对象、数组、函数以及类)

第二部分主要讲述客户端JavaScript：HTML5和相关标准定义的JavaScript API以及Web浏览器实现的API。

脚本化文档与事件处理

第三部分是JavaScript语言核心部分的参考手册。

第四部分是客户端JavaScript的参考手册

第1章JavaScript概述

JavaScript解释器

前端与编译原理——用JS写一个JS解释器

<https://juejin.cn/post/6844903730135957517>

描述网页内容的HTML

描述网页样式的CSS

描述网页行为的JavaScript

高端的、动态的、弱类型的

面向对象和函数式的

**一等函数（first-class function）**

**原型（prototype-based）**

**(Java/Scheme/Self）**

**健壮性、高效性和通用性**

JavaScript的语法源自Java，

它的**一等函数（first-class function）**来自于Scheme，  
它的基于**原型（prototype-based）**的继承来自于Self。但使用本书学习JavaScript不必去了解那些（Java/Scheme/Self）语言或熟悉那些术语。

网景（Netscape）公司

Sun Microsystem公司（现在的Oracle）的注册商标

商标冲突

* 1. JavaScript语言核心 **概览**

第1章基础知识

第2章JavaScript注释、分号和Unicode字符集

第3章JavaScript变量和赋值

类的特性

“动态可继承”（dynamically extendable）

它弱化了对象的类型，强化了对象的功能。

诗家总爱西昆好，独恨无人作郑笺。

# **为什么要读源代码，如何阅读源代码**

<https://blog.csdn.net/hbwang_716/article/details/106165644>

<https://blog.csdn.net/ajian005/article/details/7921507>

C:\Program Files\JetBrains\WebStorm 2021.1.3\plugins\

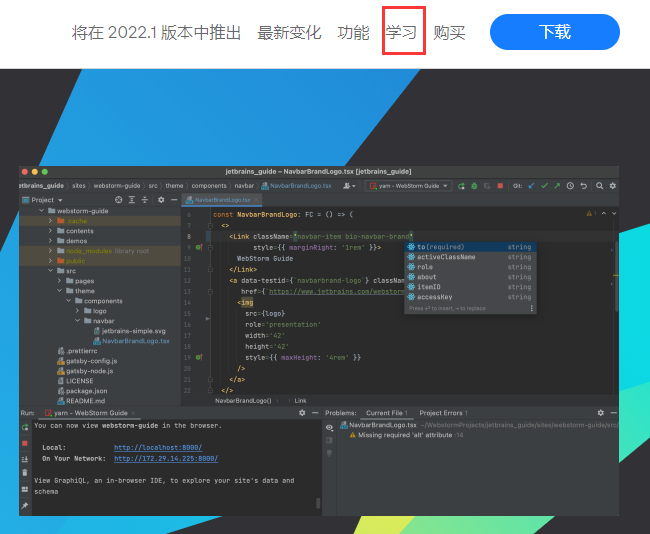
**JavaScriptLanguage**\jsLanguageServicesImpl\external\lib.dom.d.ts

论诗三十首》是[元好问](https://baike.baidu.com/item/%E5%85%83%E5%A5%BD%E9%97%AE/586311" \t "https://baike.baidu.com/item/%E8%AE%BA%E8%AF%97%E4%B8%89%E5%8D%81%E9%A6%96/_blank)继[杜甫](https://baike.baidu.com/item/%E6%9D%9C%E7%94%AB/63508" \t "https://baike.baidu.com/item/%E8%AE%BA%E8%AF%97%E4%B8%89%E5%8D%81%E9%A6%96/_blank)之后运用绝句形式比较系统地阐发[诗歌理论](https://baike.baidu.com/item/%E8%AF%97%E6%AD%8C%E7%90%86%E8%AE%BA/7773174" \t "https://baike.baidu.com/item/%E8%AE%BA%E8%AF%97%E4%B8%89%E5%8D%81%E9%A6%96/_blank)的著名组诗。他评论了自汉魏至宋代的许多著名作家和流派，表明了他的文学观点，对后世有重要影响。

<http://www.dba.cn/>

[WebStorm](https://www.jetbrains.com/zh-cn/webstorm/)最智能的 JavaScript IDE





npm 是 JavaScript 的包管理器

[React](https://react.docschina.org/docs/getting-started.html) :一个 JavaScript 库

[React](https://react.docschina.org/docs/getting-started.html) 是一个 JavaScript 库,所以我们假设你对 JavaScript 语言已有基本的了解。如果你对自己的基础不自信，我们推荐**[通过 JavaScript 教程](https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript)**检查你的基础知识储备水平，使得你能够无压力的阅读本指南。这可能会花费你 30 分钟到 1 个小时的时间，但这样做的好处是你不会觉得同时在学习 React 和 JavaScript。

组件逻辑使用 JavaScript 编写而非模版，因此你可以轻松地在应用中传递数据，并使得状态与 DOM 分离。

## **[Babel](https://babeljs.io/) :**一个 JavaScript编译器

Vue.js作注之难



vm.\_self = vm

initLifecycle(vm)

initEvents(vm)

initRender(vm)

callHook(vm, 'beforeCreate')

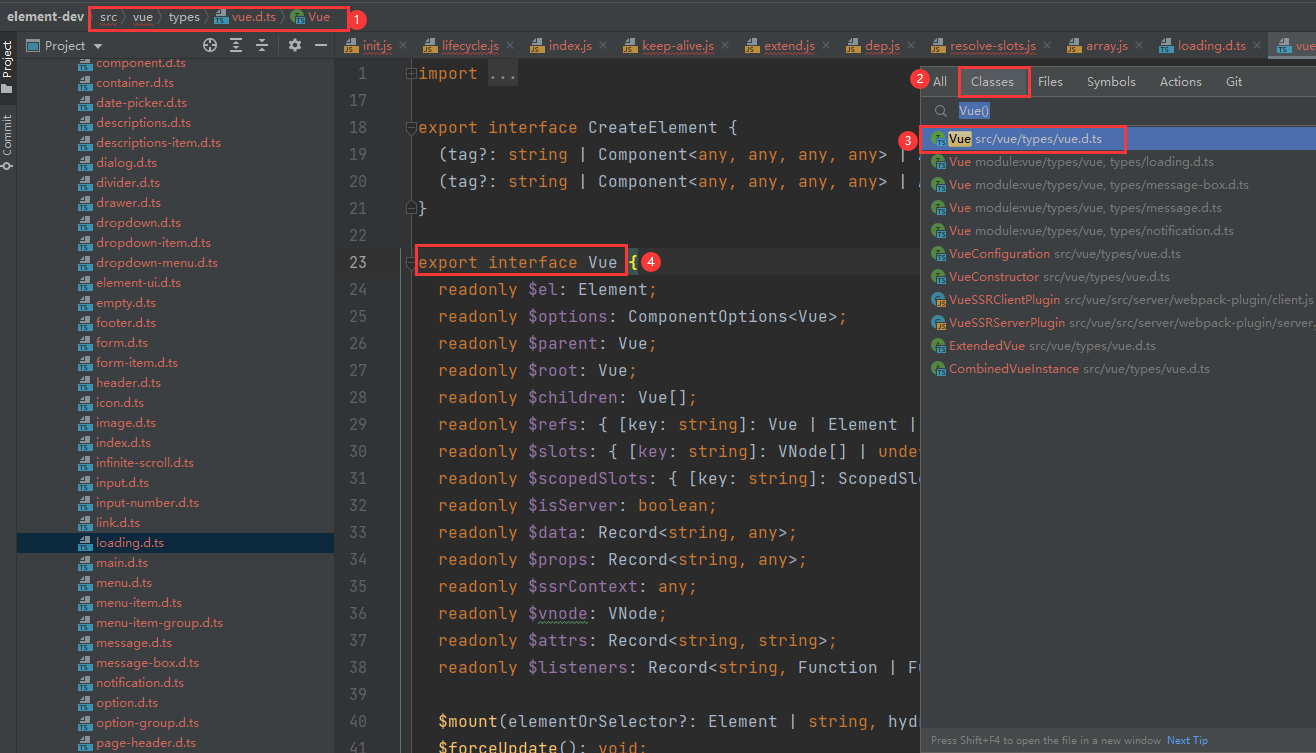
initInjections(vm) // resolve injections before data/props

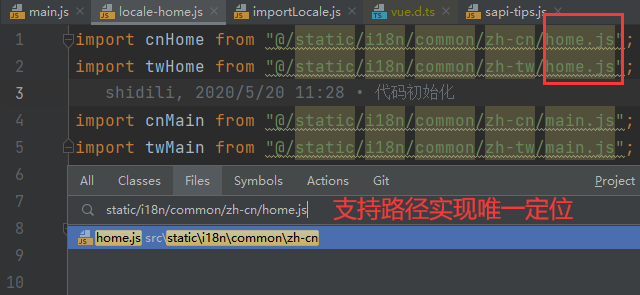
initState(vm)

initProvide(vm) // resolve provide after data/props

callHook(vm, 'created')

Vue在何处





为了防止属性名冲突， JavaScript 介绍了一种 symbols 的类型。在 2015 - 2019 中，symbols 提供一种方法去模拟私有属性。

* 在ES6之前,js的的基本数据类型有String Number Null Boolean undefined Object6中数据类型,Symbol是一种新增加的基本数据类型

## **特性**

* Symbol 不需要new关键字,他是原始类型,不是对象
* Symbol 不论参数是否相等,他返回值都不相等

js中的Object.create(null) 和 {} 的区别

前端js

console.log(Object.create(null))

console.log({})

结果图



根据结果我们不难发现，两者最大区别在于Object.create(null)创建的对象没有继承Object 所有Object.create(null)创建的对象也没有Object提供的方法，是一个非常纯净的对象

**ps: Object.create(obj)实际上表示创建一个对象，并给他指定原型对象**

————————————————

版权声明：本文为CSDN博主「前端打字员」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/a153052/article/details/102839698>

碎碎念

instanceof是[Java](https://baike.baidu.com/item/Java/85979" \t "https://baike.baidu.com/item/instanceof/_blank)、[php](https://baike.baidu.com/item/php/9337" \t "https://baike.baidu.com/item/instanceof/_blank)的一个二元操作符（运算符），**和==，>，<是同一类东西**

**[] instanceof Array true**

**typeof [] "object"**

因为Array，Function，Date虽然是基于Object进行创建的，但是他们****继承的是Object.toString()，而不是Object.prototype.toString()。****

再加深一遍印象：

Object.toString()//"function Object() { [native code] }"

Object.prototype.toString()//"[object Object]"

if (!(key in vm)) {

defineComputed(vm, key, userDef)

}

'a' in { a: 3 } true

'b' in { a: 3 } false

Init.js文件

internal英 [ɪn'tɜːnl] 　 　 美 [ɪn'tɜːrnl]  adj. 国内的；内部的；身内的

export function initIternalComponent () {}

dedupe美 [dɪˈdup] 英 [dɪ'djup]

网络重复资料删除；重复数据删除技术；数据重复删除技术

function dedupe() {}

/\* istanbul ignore if \*/

这个软件以土耳其最大城市**伊斯坦布尔**命名，因为土耳其地毯世界闻名，而地毯是用来覆盖的。

[Istanbul](https://github.com/gotwarlost/istanbul" \t "https://www.techug.com/post/_blank) 是 JavaScript 程序的代码覆盖率工具

istanbul 提供[注释语法](https://github.com/gotwarlost/istanbul/blob/master/ignoring-code-for-coverage.md" \t "https://www.techug.com/post/_blank)，允许某些代码不计入覆盖率。

上面代码的 if 语句块，在计算覆盖率的时候会被忽略。

代码覆盖率工具 Istanbul 入门教程

<http://www.ruanyifeng.com/blog/2015/06/istanbul.html>

optimize

优化

if (***options*** && ***options***.\_isComponent) {  
 // optimize（优化） internal component instantiation（实例化）  
 // since dynamic （动态）options merging is pretty slow, and none of the  
 // internal component options needs special treatment（处理）.  
 initInternalComponent(vm, ***options***)  
}

vue.common.js

function Vue (options) {  
 if (***process***.env.NODE\_ENV !== 'production' &&  
 !(this instanceof Vue)  
 ) {  
 warn('Vue is a constructor and should be called with the `new` keyword');  
 }  
 this.\_init(options);  
}  
  
initMixin(Vue);  
stateMixin(Vue);  
eventsMixin(Vue);  
lifecycleMixin(Vue);  
renderMixin(Vue);

[JSDoc](http://www.dba.cn/book/jsdoc/)

/\*\*

 \*

 \* @param {\*} a

 \* @param {\*} b

 \*/

function aa(a, b) {

}

Intercept mutating methods and emit events

拦截变异的方法并发出事件

[JavaScript中文手册](http://www.dba.cn/book/javascript/)

Node.js 运行在服务端的 JavaScript。

防抖

解释定义

防抖是指在一定的时间内再次触发此事件,会清空上次的事件重新开始,如果制定的时间内没有再次触发,那么这个事件才会执行

例如: input输入信息,不可能每次按下都发起一个ajax请求,可以等一段时间内不输入了之后在发起请求

代码样例

function debounce(sayHi){

var timer = null

return function(){

clearInterval(timer)

timer = setTimeout(()=>{

sayHi.apply(this,arguments)

},1000)

}

}

function sayHi(){

console.log('防抖')

}

var inp = document.getElementById('inp');

inp.addEventListener('input', debounce(sayHi)); // 防抖

节流

解释定义

节流是指在一定的时间同一事件只会触发一次,只有超过了这个时间才会再次出发

例如: 验证码60秒内不可以再次触发(实际开发肯定是使用禁止,但是原理和验证码一样)

代码样例

function chufa(fn){

var flag = true

return function(){

if(!flag) return;

flag = false

setTimeout(()=>{

fn.apply(this,arguments)

flag = true

},1000)

}

}

function hansh(e){

console.log(e.target.innerWidth, e.target.innerHeight);

}

window.addEventListener('resize',chufa(hansh))

————————————————

版权声明：本文为CSDN博主「残梦a」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：<https://blog.csdn.net/qq_43268396/article/details/103486354>

*/\*\* EventTarget is a DOM interface implemented by objects that can receive events and may have listeners for them. \*/*interface EventTarget {  
 */\*\*  
 \* Appends an event listener for events whose type attribute value is type. The callback argument sets the callback that will be invoked when the event is dispatched.  
 \*   
 \* The options argument sets listener-specific options. For compatibility this can be a boolean, in which case the method behaves exactly as if the value was specified as options's capture.  
 \*   
 \* When set to true, options's capture prevents callback from being invoked when the event's eventPhase attribute value is BUBBLING\_PHASE. When false (or not present), callback will not be invoked when event's eventPhase attribute value is CAPTURING\_PHASE. Either way, callback will be invoked if event's eventPhase attribute value is AT\_TARGET.  
 \*   
 \* When set to true, options's passive indicates that the callback will not cancel the event by invoking preventDefault(). This is used to enable performance optimizations described in § 2.8 Observing event listeners.  
 \*   
 \* When set to true, options's once indicates that the callback will only be invoked once after which the event listener will be removed.  
 \*   
 \* The event listener is appended to target's event listener list and is not appended if it has the same type, callback, and capture.  
 \*/* addEventListener(type: string, listener: EventListenerOrEventListenerObject | null, options?: boolean | AddEventListenerOptions): void;  
 */\*\*  
 \* Dispatches a synthetic event event to target and returns true if either event's cancelable attribute value is false or its preventDefault() method was not invoked, and false otherwise.  
 \*/* dispatchEvent(event: Event): boolean;  
 */\*\*  
 \* Removes the event listener in target's event listener list with the same type, callback, and options.  
 \*/* removeEventListener(type: string, callback: EventListenerOrEventListenerObject | null, options?: EventListenerOptions | boolean): void;  
}

**Apache ECharts**一个基于 JavaScript 的开源可视化图表库

[快速入门](https://echarts.apache.org/handbook/zh/get-started)[所有示例](https://echarts.apache.org/examples/zh/index.html)

<https://echarts.apache.org/zh/index.html>

[https://echarts.apache.org/examples/zh/index.html#chart-type-bar](https://echarts.apache.org/examples/zh/index.html" \l "chart-type-bar)

JavaScript教程 廖雪峰

每个.js文件都是一个模块

要在模块中对外输出变量，用：

**module**.exports = variable;

[脚本之家](https://www.jb51.net/) <https://www.jb51.net/>

跨域

<https://blog.csdn.net/yzx0xyz/article/details/83686549>

# 提高 web 应用性能之 JavaScript 性能调优(转)

<https://blog.csdn.net/weixin_34008933/article/details/89756884>

闭包

内存

<https://blog.csdn.net/weixin_44867717/article/details/121121269>





this.initProjectModel = ***Object***.assign(this.initProjectModel, res.FinishCarryExpense)

浅拷贝共享内存，深拷贝不共享内存

# **前端与编译原理——用JS写一个JS解释器**

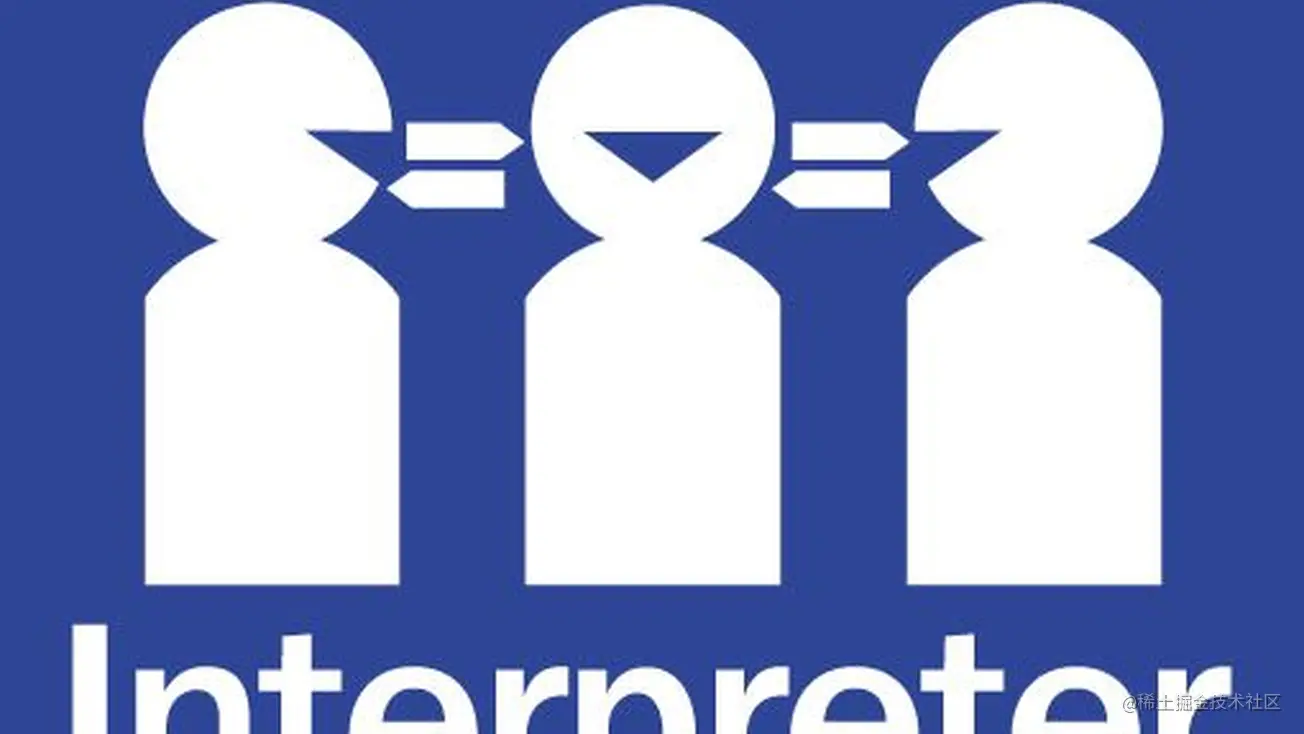
<https://juejin.cn/post/6844903730135957517>

[](https://juejin.cn/user/712139233835879)

[Jrainlau IMG_257](https://juejin.cn/user/712139233835879" \t "https://juejin.cn/post/_blank)

2018年12月03日 23:31 ·  阅读 5132

关注





说起编译原理，印象往往只停留在本科时那些枯燥的课程和晦涩的概念。作为前端开发者，编译原理似乎离我们很远，对它的理解很可能仅仅局限于“抽象语法树（AST）”。但这仅仅是个开头而已。编译原理的使用，甚至能让我们利用JS直接写一个能运行JS代码的解释器。

项目地址：[github.com/jrainlau/c.…](https://link.juejin.cn/?target=https://github.com/jrainlau/canjs" \o "https://github.com/jrainlau/canjs" \t "https://juejin.cn/post/_blank)

在线体验：[codepen.io/jrainlau/p.…](https://link.juejin.cn/?target=https://codepen.io/jrainlau/pen/YRgQXo" \o "https://codepen.io/jrainlau/pen/YRgQXo" \t "https://juejin.cn/post/_blank)点击预览

## 为什么要用JS写JS的解释器

接触过小程序开发的同学应该知道，小程序运行的环境禁止new Function，eval等方法的使用，导致我们无法直接执行字符串形式的动态代码。此外，许多平台也对这些JS自带的可执行动态代码的方法进行了限制，那么我们是没有任何办法了吗？既然如此，我们便可以用JS写一个解析器，让JS自己去运行自己。

在开始之前，我们先简单回顾一下编译原理的一些概念。

## 什么是编译器

说到编译原理，肯定离不开编译器。简单来说，当一段代码经过编译器的词法分析、语法分析等阶段之后，会生成一个树状结构的“抽象语法树（AST）”，该语法树的每一个节点都对应着代码当中不同含义的片段。

比如有这么一段代码：