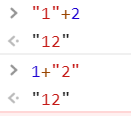
1. 博客
2. js异步编程的方法

回调函数、事件监听、发布\订阅、Promise对象

1. 解释语言的特性

非独立性、效率低

1. 字符串+数字



+会将数字转换为字符串，如果+前后有字符串

1. RegExp

reg.test()测试这个字符串是否满足这个正则表达式 返回布尔值

reg.exec()返回匹配这个表达式的子字符串

reg.compile 编译或修改正则表达式

str.match()字符串是否匹配这个正则表达式

var str='Hello World';

var reg=new RegExp("World");

console.log(reg.test(str));//true

console.log(reg.exec(str));//["World"]

console.log(reg.compile(str));///Hello World/

console.log(str.match(reg));//["Hello World", index: 0, input: "Hello World", groups: undefined]

var str="Every man in the world! Every woman on earth!";

var patt = /man/g;

str2 = str.replace(patt, "person");

console.log(str2);//Every person in the world! Every woperson on earth!

patt = /(wo)?man/g;

patt.compile(patt);

str2 = str.replace(patt, "person");

console.log(str2);//Every person in the world! Every person on earth!

1. setTimeout是window对象的方法，不是全局函数。
2. 函数参数按值传递

var a=1;

var obj={b:2}

function fn(x){

x++;

}

fn(a);

fn(obj.b);

console.log(a);//1

console.log(obj);//{b:2}

1. js执行

* js执行分为编译阶段和执行阶段。编译阶段分为：词法分析、语法分析、生成可执行代码。执行阶段分为创建一个全局对象、创建一个执行环境栈、创建全局执行上下文、全局上下文推入执行环境栈底、代码从上往下执行生成函数的执行上下文推入栈中、函数执行完其执行上下文出栈、全局执行上下文出栈。
* 词法分析：将由字符组成的字符串分解成有意义的代码块(语法单元)。
* 语法分析：将词法单元流组成抽象语法树(AST)。
* 生成可执行代码：将AST转换为可执行代码。

var a = 2;

function bar() {

var b = 2;

function foo() {

var c = 2;

}

foo();

}

bar();

* 1. 创建一个全局对象

全局对象的属性在任何地方都可以访问，它的存在伴随着程序的整个生命周期。全局对象在创建时，将Math,String,Date,document 等常用的JS对象作为其属性。由于这个全局对象不能通过名字直接访问，因此还有另外一个属性window,并将window指向了自身，这样就可以通过window访问这个全局对象了。

伪代码：

//创建一个全局对象

var globalObject = {

Math:{},

String:{},

Date:{},

document:{}, //DOM操作

...

window:this //让window属性指向了自身

}

* 1. Js引擎会创建一个执行环境栈

栈是新进后出

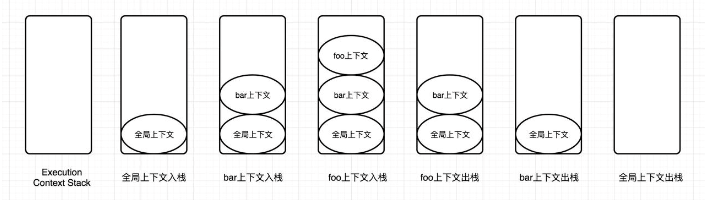
* 1. 创建全局执行上下文

js中有3种作用域：全局环境、函数环境、eval、with等(不推荐使用)

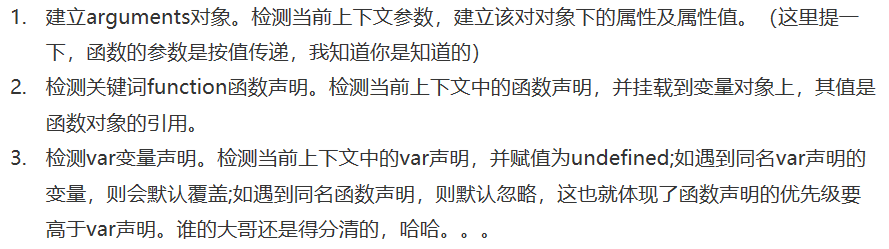
一个作用域中可能有多个执行上下文

作用域和上下文的区别：作用域在定义时确定、像一个地盘、除了全局作用域，每个函数都要创建一个作用域、上下文是在调用时创建的、上下文中包含着定义的变量。

* 1. 全局上下文推入执行环境栈底
  2. 代码从上到下执行，当执行到bar()，生成bar执行上下文，推入栈中
  3. 代码执行到foo()，生成foo执行上下文，推入栈中
  4. foo()执行完，foo执行上下文出栈
  5. bar()执行完，bar执行上下文出栈
  6. 全局上下文出栈



* 创建变量对象



* 变量对象(Variable Object)和活动对象(Active Object)的区别

他们其实都是同一个对象，只是处于执行上下文的不同生命周期。在执行阶段变量对象变为活动对象。不过只有处于函数调用栈栈顶的执行上下文中的变量对象，才会变成活动对象。

1. js的继承

<https://segmentfault.com/a/1190000014212372?utm_source=tag-newest>

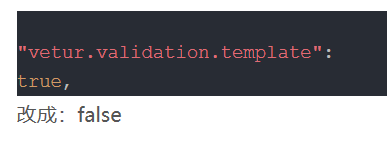
1. vue
2. vetur插件提示

[vue-language-server] Elements in iteration expect to have 'v-bind:key' directives.

Renders the element or template block multiple times based on the source data

原因：Vue 2.2.0+的版本里，当在组件中使用v-for时，key是必须的

方法：更改vetur配置 vscode->首选项->设置->搜索（vetur）



1. vue组件之间的调用



components:[] 应该是中括号，不是大括号

1. 组件之间的通信-props



app.vue调用header.vue 父组件向子组件传递信息

1. 组件名称要从小驼峰转换为-形式
2. 自定义事件 用于组件间通信

$on()监听事件

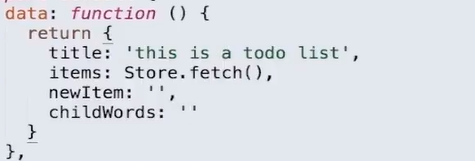
$emit()在它上面触发事件

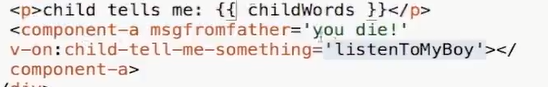
$dispatch()派发事件，事件沿父链冒泡

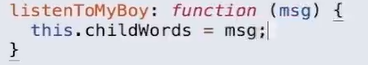
$broadcast()广播事件，向下传导给所有后代

子组件向父组件传值：

parent.vue:

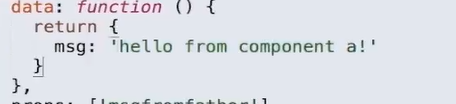


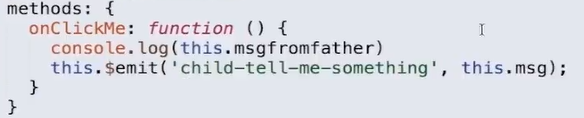




child.vue







子组件向父组件传值可以不用而用$ispatch

父组件向子组件传值可以不用emit而用$boardcast