0、语雀：<https://www.yuque.com/fe9/basic/mvx83k>

1、主流浏览器内核：

|  |  |
| --- | --- |
| IE | Trident |
| Chrome | Webkit/blink |
| Firefox | Gecko |
| Opera | Presto |
| Safari | Webkit |

2、

// 第一种方式

function box (参数){

// 内容

};

// 第二种叫命名函数表达式

var box = function test(参数){

// 内容

}

// 在浏览器console上输出 box.name --> test

// 第二种写法function后面就成了表达式,有没有名字无所谓,所以延伸出了第三种写法

// 第三种叫匿名函数表达式-->函数表达式

var box = function (参数){

//内容

}

// 在浏览器console上输出 box.name --> box

//函数执行

box();

3、**JS 的映射规则**

function sum(a,b){

a = 2;

console.log(arguments[0]); // 打印出2，参数a改变了

}

sum(1,2);

function sum(a,b){

a = 2;

arguments[0] = 3

console.log(arguments[0]); // 打印出3,参数a改变了,可以相互改变

}

sum(1,2);

----------------------------------------------------------------

function sum(a,b){

//arguments[1]没值

b = 2;

console.log(arguments[1]);

}

sum(1);

* 打印出 undefined
* 实参列表传进来的时候他就有几个，就算我让b等于2，它也不会往 arguments 里面加了。
* 函数根本就没有，这个时候 b 就当一个变量用了。
* b 实参不映射，因为形参比实参多了一位 b，只有他们相等的时候他们才会有映射的规则，不相等的时候，形参多了他不对应实参了，他们之间不映射。

4、预编译：函数声明整体提升，变量声明（注意区分变量声明和变量赋值，这是两个步骤）提升。暗示全局变量：即任何变量，如果变量未经声明就赋值，该变量就为全局对象所有（在函数体内定义的全局变量，要在外面调用过一次之后，才能再次被访问到）。

var a = b = 10; b就是全局变量，因为b没有被声明就被赋值。

5、预编译正式：

• 创建AO对象

• 找形参和变量声明,将变量和形参名作为AO属性名,值为undefined

• 将实参值和形参统一

• 在函数体里面找函数声明，值赋予函数体

6、运行期上下文 ： 当函数执行前/时，会创建一个称为执行期上下文的内部对象。一个执行期上下文定义了一个函数执行时的环境，函数每次执行时对应的执行上下文都是独一无二的，所以多次调用一个函数会导致创建多个执行期上下文，当函数执行完毕，它所产生的执行上下文被销毁。

<https://www.yuque.com/fe9/basic/zd0d1h>

每个函数的scope chain里面的GO和非自身的AO都是同一个东西，非自身的AO是一个引用，而不是拷贝。

7、闭包：一个函数（outer）内部的另一个函数（inner）可以访问outer的变量。即：当内部函数被保存到外部时，将会生成闭包。闭包会导致原有作用域链不释放，造成内存泄露。

作用：

* 实现公有变量
* 可以做缓存（存储结构）
* 实现封装，属性私有化
* 模块化开发，防止污染全局变量

8、立即执行函数：适用于对数据初始化、整个程序只执行一次的函数，执行完函数的引用就销毁了，但销毁的意思不是函数不存在了，是其他地方再也找不到这个立即执行函数！（立即执行时忽略函数名）除了会立即销毁外，拥有所有普通函数的特性。

[var \*\* = ](function ([形参1],[形参2],[形参3]) {  
 //do something

[return \*\*;]  
}([实参1],[实参2],[实参3]))

9、关于函数的补充，注意区分函数的声明和函数表达式，只有函数表达式可以被执行（用“()”就能执行）

e.g

函数声明：

function () {  
 //do something  
}

函数表达式：

var a = function () {  
  
}

函数声明是无法执行的，但函数表达式，在后面加上括号就能执行：

var ***a*** = function () {  
 ***console***.log(123);  
}();

输出123。

且函数表达式后面的括号里是可以加入实参，被函数表达式的形参调用的，用法参考立即执行函数（实际上就是立即执行函数的另一种写法），立即执行时会忽略该函数名。

可以在函数声明前面加上+/-/!转换为函数表达式，这样就可以在函数表达式后面加上括号用了。

善用立即执行函数来解决闭包造成的数据只有一个值的问题。

10、逗号运算符：

(A,B) 先运算A然后运算B，然后把B的运算结果返回出去。

11、运算符：

typeof：返回的是字符串。



delete：删除成功返回true

12、对象：

this：在对象内部使用this时就是对象本身的意思。

增：直接使用 对象名.属性 = 属性值

删：delete 对象名.属性

改：对象名.属性 = 新属性值

查：对象名.属性

创建方法：

1. var obj = {} 对象字面量/对象直接量
2. 构造函数：（命名规则，每个单词首字母大写）
3. 系统自带构造函数:var obj = new Object();
4. 自定义:

function Obj() {  
 this.value1 = "key1";  
 this.value2 = "key2";  
 this.value3 = "key3";  
}  
var ***obj*** = new Obj();

构造函数内部原理：

1. 在函数体最前面隐式的加上this={}
2. 执行this.xxx = xxx;
3. 隐式的返回this

注：返回值可以手动干预，可以return 函数，对象，数组等等，但不能返回原始值，例如123(number)，“qwe”(string)，true/false(Boolean),undifined，否则返回无效，仍然返回的是this。

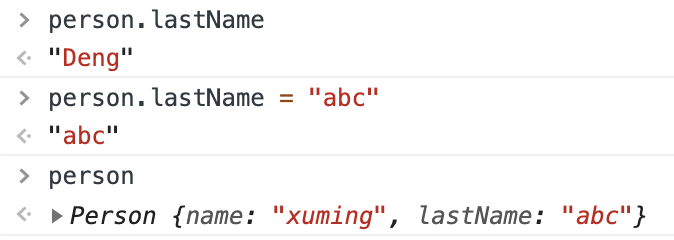
13、JavaScript 数据类型和数据结构

<https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Data_structures>

14、原型：实际上是一个对象

构造函数名. prototype = {} 构造函数名. prototype.属性值

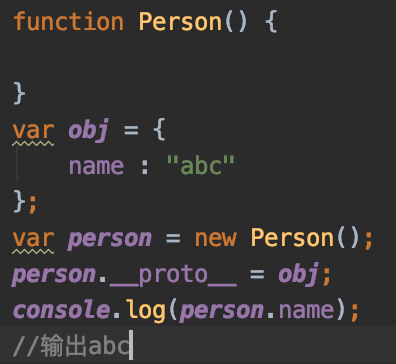
无法通过new出来的后代对象修改原型的值，只会在对象里增加一个属性，属性值就是赋值的。



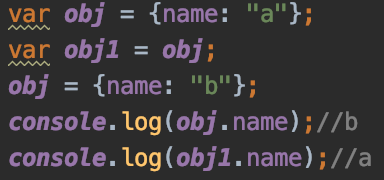
直接修改原型是可以的



在构造新对象时，内部原理的第一步“在函数体最前面隐式的加上this={}”，实际上this={}里面有一个\_\_proto\_\_:构造函数名.prototype，这里这个\_\_proto\_\_是可以重新给他赋值的，这样就能指向其他的对象。也正因为有\_\_proto\_\_:构造函数名.prototype ，new出来的对象才能沿着\_\_proto\_\_找到原型。



注意区分“构造函数名.prototype.属性 = 属性值”和“构造函数名.prototype = { 属性 = 属性值 }”，他们的过程跟普通的对象赋值一样，注意往堆栈里思考问题。后者那样赋值不是“地址”赋值，是重新的一个拷贝，再次访问“构造函数名.prototype.属性”的时候，还是原来的那个。这种情况只适用于在对象new出来之后，再“构造函数名.prototype = { 属性 = 属性值 }”。



在new之前的话，属性值会变。

object.prototype是大部分对象的最终原型。

例外出现在如下方法创建的对象中，为完全干净的一个对象。

