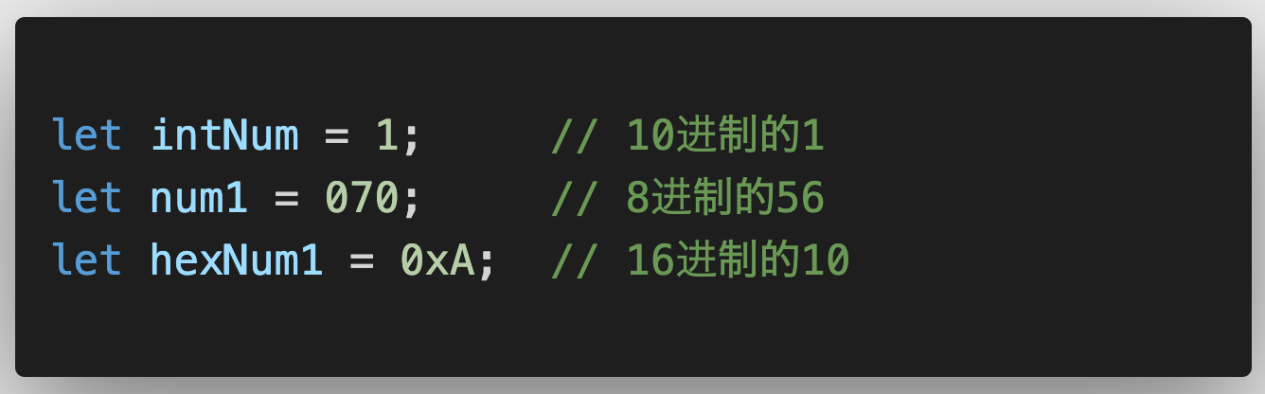
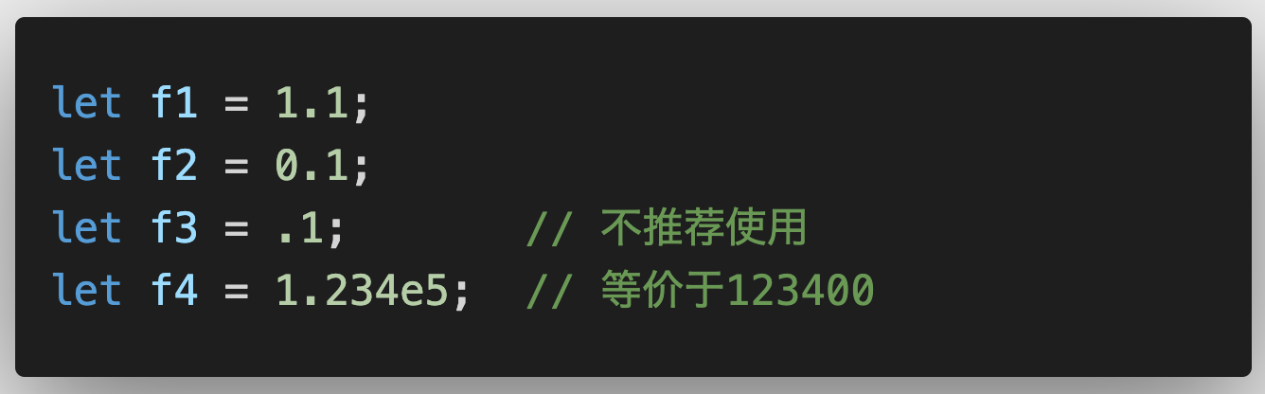
# JavaScript

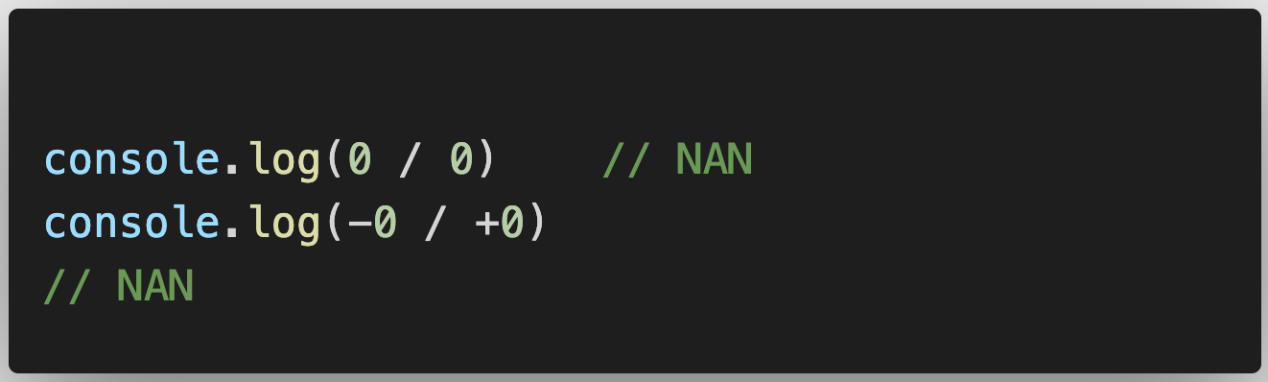
## 谈谈JavaScript中的数据类型，以及存储上的差别？

### 基本类型

Number







Undefined



String

字符串可以用双引号（"）、单引号（'）或反引号（`）表示。

字符串是不可变的，一旦创建，不可更改。



Null

逻辑上讲，null值表示一个空对象指针。

undefined由null派生而来。

null可作为变量填充值。



Boolean

Boolean有true和false两个字面值。

通过Boolean可以将其他类型的数据转化为布尔值。



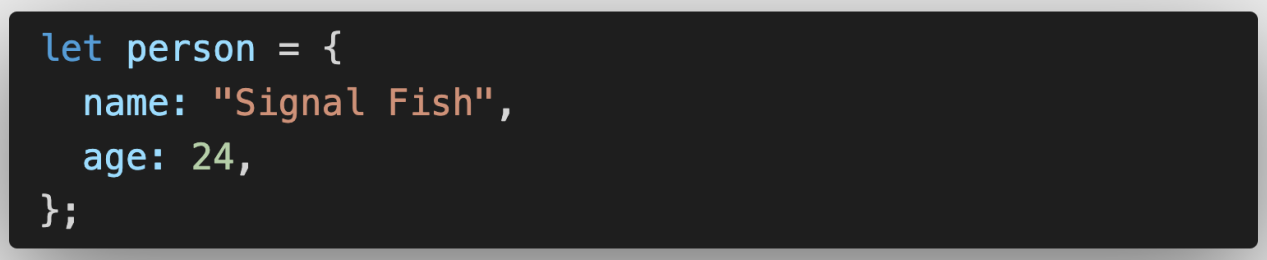
Symbol

Symbol是原始值，且符号实例是唯一、不可变的。Symbol的用途是确保对象属性使用唯一标识符，不会发生属性冲突的危险。



### 引用类型

Object



Array

在JavaScript中，数组是一组有序的数据。

数组可以存储任意类型的数据。

数组会随着数据的添加而动态扩容。

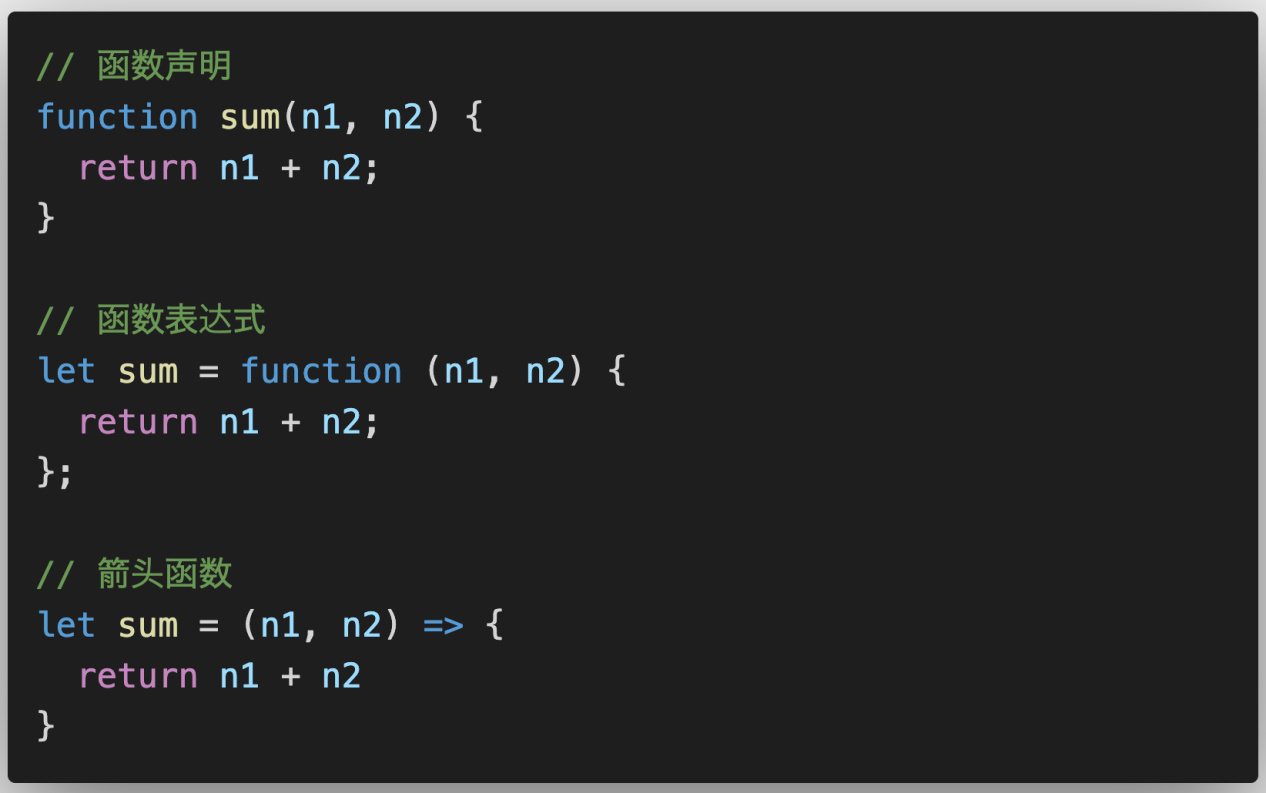


Function

函数是对象，每个函数都是Function类型的实例。

Function也有属性和方法。

函数存在三种常见的表达式：函数声明，函数表达式，以及箭头函数。



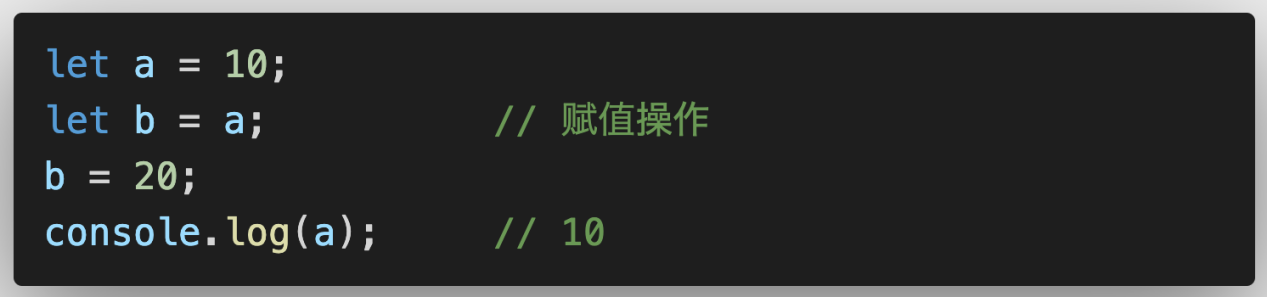
### 其他引用类型

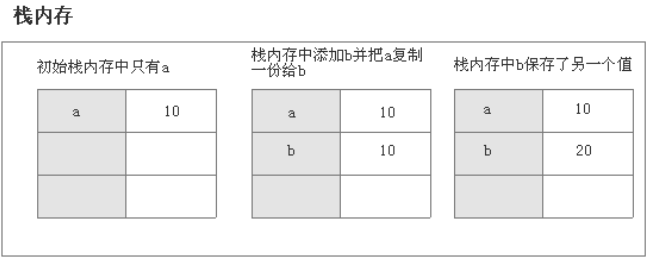
Date、RegExp、Map、Set等

### 存储区别

基本类型

基本类型数据存储在栈中，引用类型数据存储在堆中。

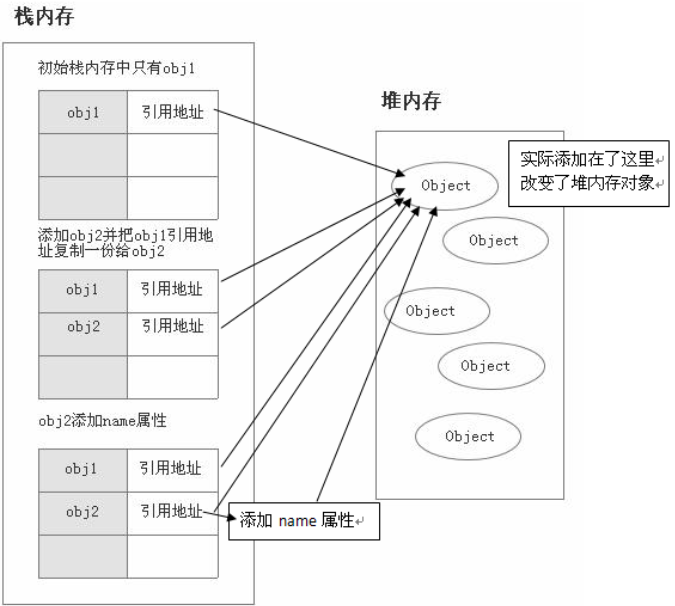




引用类型

引用类型数据存储在堆内存中，每个堆内存中有一个引用地址，该引用地址存储在栈中。





## 数组常用的方法有哪些？

### 增

push()、unshift()、splice()、concat()

push、unshift、和splice操作会对原生数组造成影响，concat操作不会对原生数组造成影响。

### 删

pop()、shift()、splice()、slice()

pop、shift、splice操作会对原生数组造成影响，slice操作不会对原生数组造成影响。

### 改

splice()

### 查

indexOf()、includes()、find()

### 排序

reverse()、sort()

### 转换

join()

### 迭代

map()、filter()、forEach()、every()、some()

## 字符串的常用方法有哪些？

### 增

+、${}、concat()

### 删

创建字符串的一个副本，再进行操作。

slice()、substr()、substring()

### 改

创建字符串的一个副本，再进行操作。

trim()、trimLeft()、trimRight()、repeat()、padStart()、padEnd()、toLowerCase()、toUpperCase()

### 查

charAt()、indexOf()、startWith()、includes()

### 转换

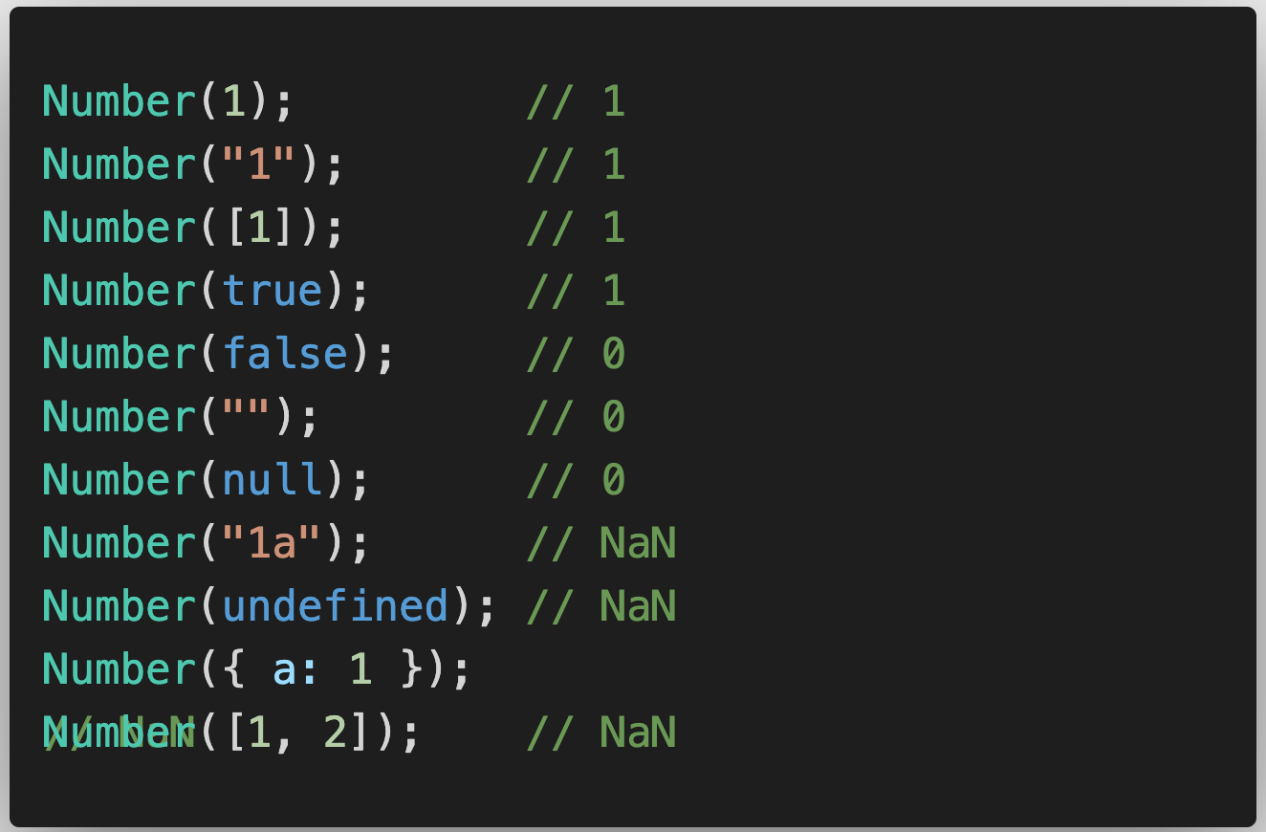
split()

### 模板匹配

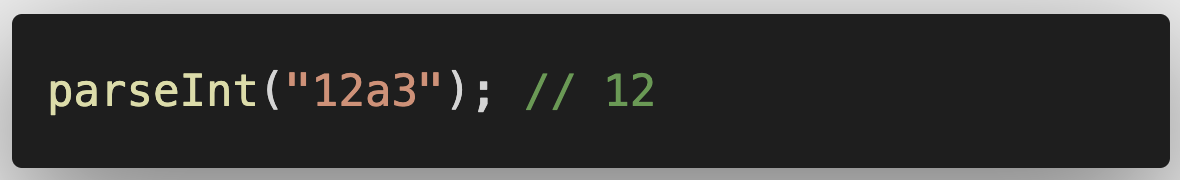
match()、search()、replace()

## JavaScript中的类型转换机制

### Number()



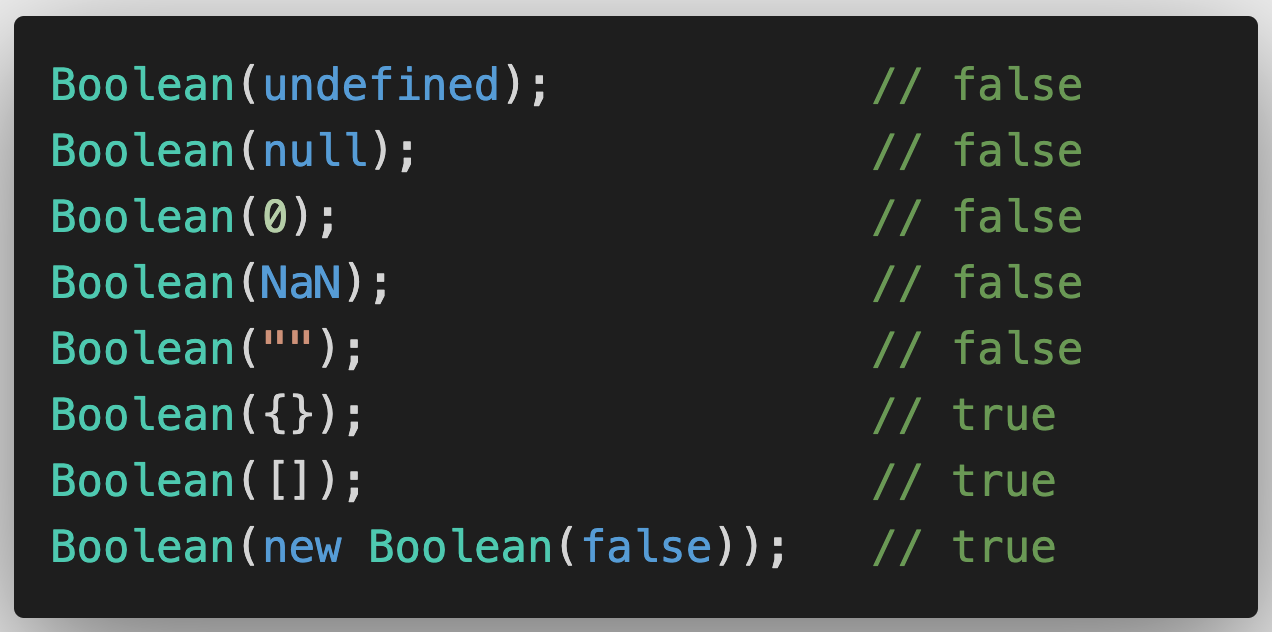
### parseInt()



### String()



### Boolean()



### 隐式转换

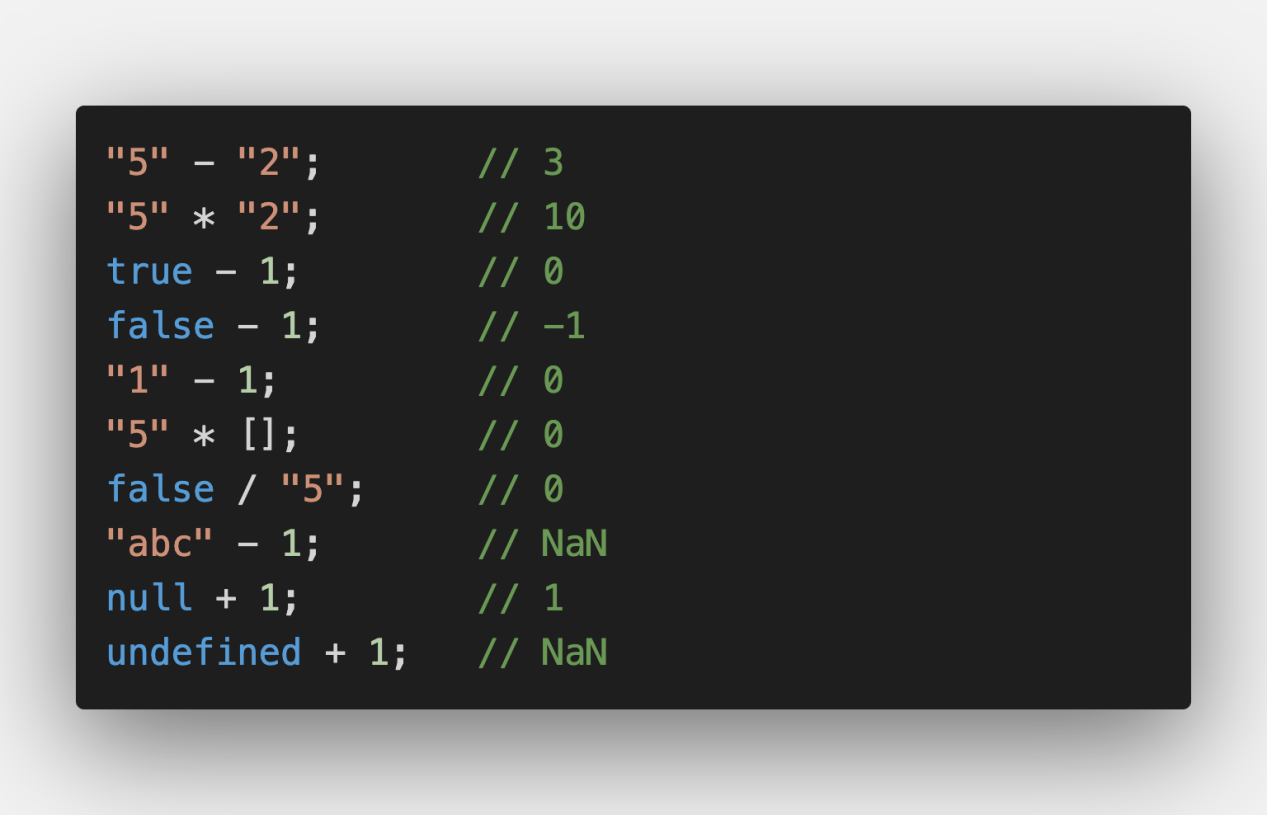
自动转换为布尔值

undefined、null、false、+0、-0、NaN、””会被自动转换为false

自动转换为字符串



自动转换为数值



## ==与===的区别

相等（==）操作符会做类型转换，再进行值得比较，全等（===）运算符不会做类型比较。

除了比较对象属性为null或者undefined的情况下，使用相等（==）操作符，其他情况一律使用全等（===）操作符。

## 浅拷贝与深拷贝的区别

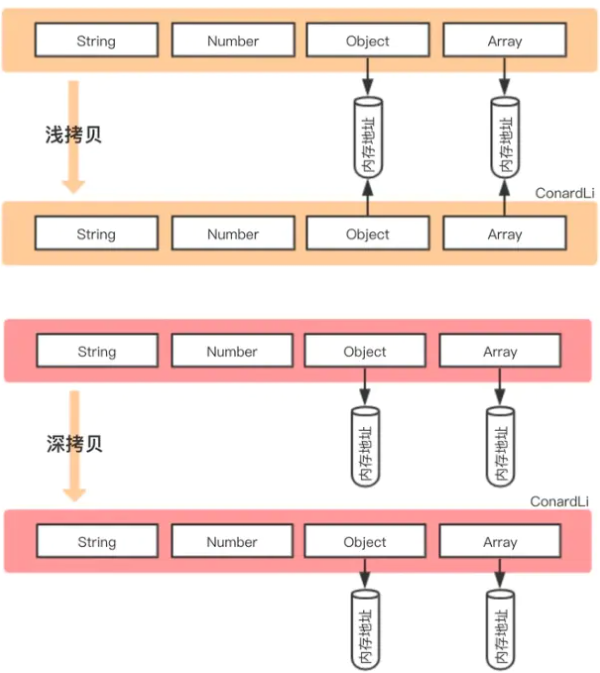
浅拷贝，指的是创建新的数据，这个数据有着原始数据属性值的一份精确拷贝。如果属性是基本类型，拷贝的就是基本类型的值。如果属性是引用类型，拷贝的就是内存地址。

常见的浅拷贝的方式有：Object.assign()、Array.prototype.slice()、Array.prototype.concat()、拓展运算符。

深拷贝会创建一个新的栈，两个对象属性完全相同，但是对应两个不同的地址，修改一个对象的属性，不会改变另一个对象的属性。

常见的深拷贝方式有：\_.cloneDeep()、jQuery.extend()、JSON.stringify()、手写循环递归

浅拷贝与深拷贝的区别：



## 闭包及其使用场景

闭包可以使一个内层函数访问到其外层函数的作用域。



使用场景：延长变量生命周期、柯里化函数

## 对作用域链的理解

作用域指变量和函数生效的区域。我们一般将作用域分为全局作用域、函数作用域和块级作用域。

### 词法作用域

词法作用域又叫静态作用域。变量被创建时就确定好，而非执行阶段确定。JavaScript遵循的是词法作用域。



var a = 2;

function foo() {

console.log(a);

}

function bar() {

var a = 3;

foo();

}

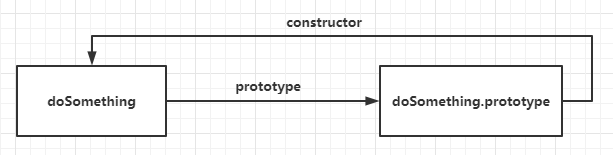
bar();

JavaScript访问变量时，遵循当前作用于-->全局作用域原则。

## 原型及原型链

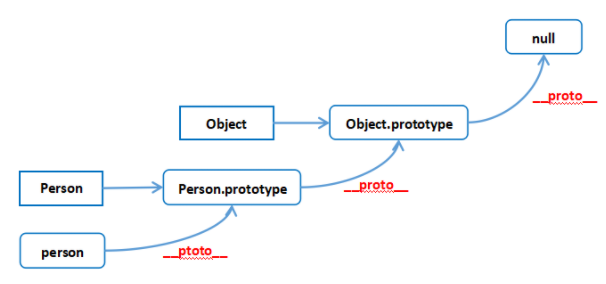
### 原型

# code



### 原型链







总结:

一切对象都是继承Object对象，Object对象直接继承根源对象null；

一切的函数对象（包括Object对象），都是继承自Function对象；

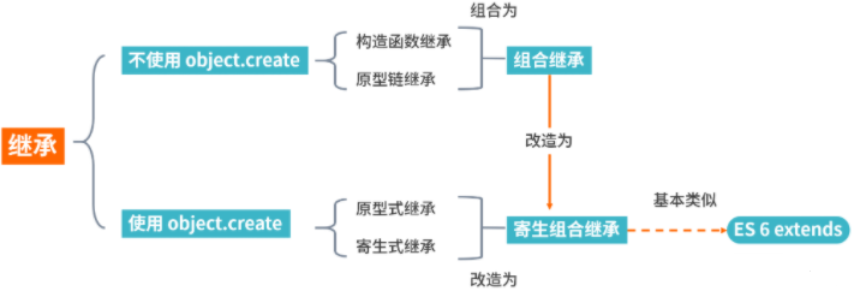
Object对象直接继承自Function对象；

Function对象的\_\_proto\_\_会指向自己的原型对象，最终还是继承自Object对象。

## JavaScript如何实现继承

JavaScript常见的继承方式：

原型链继承、构造函数继承（借助call）、组合继承、原型式继承、寄生式继承以及寄生组合式继承。



### 原型链继承



### 构造函数继承



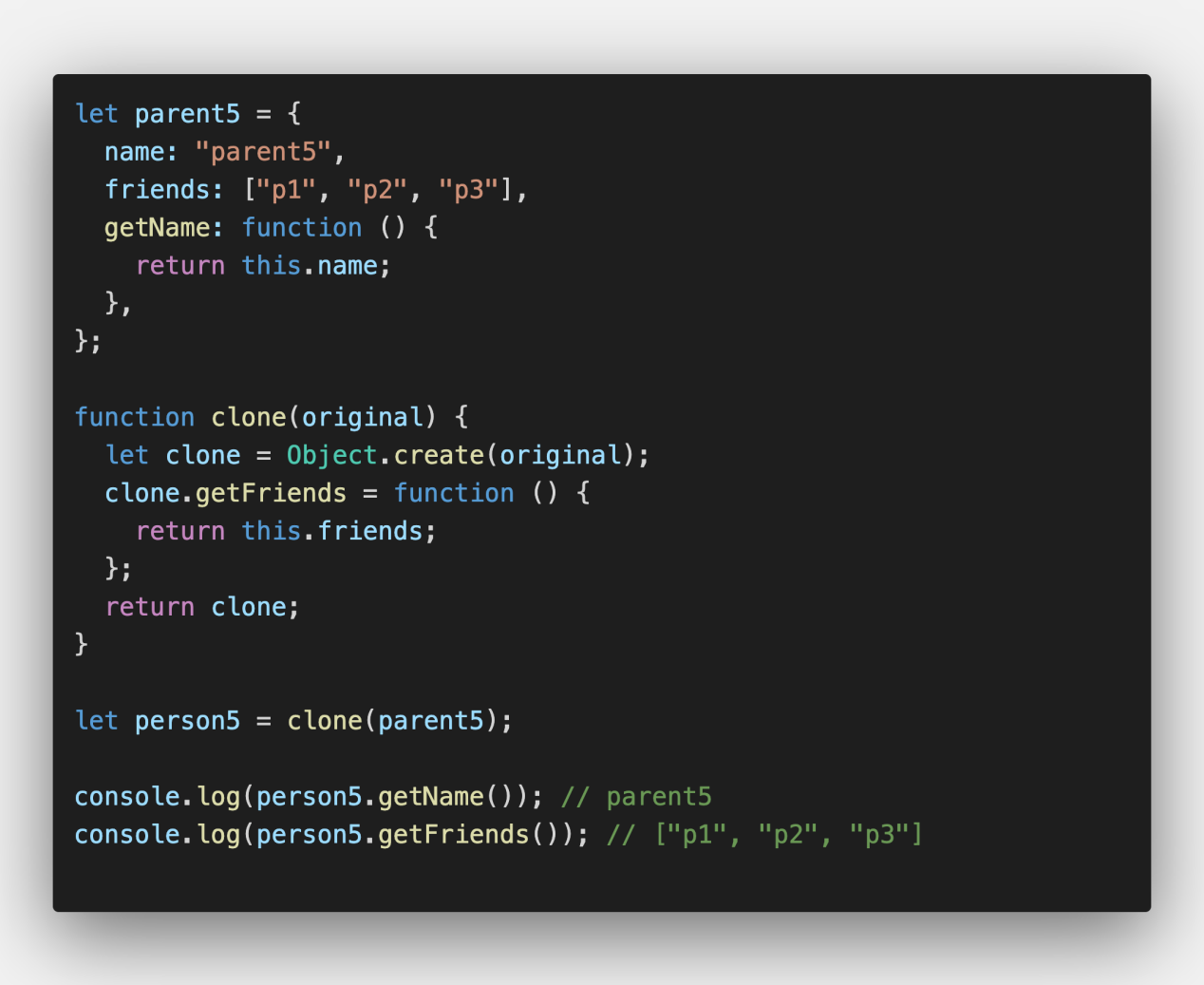
### 组合继承



### 原型式继承



### 寄生式继承



### 寄生组合式继承

寄生组合式继承是所有继承方式中相对最优的继承方式。





## 谈谈对this的理解

this关键字是函数运行时自动生成的一个内部对象，只能在函数内部使用，总指向调用它的对象。

this在函数执行过程中一经确定，就不可再更改。

### this绑定规则

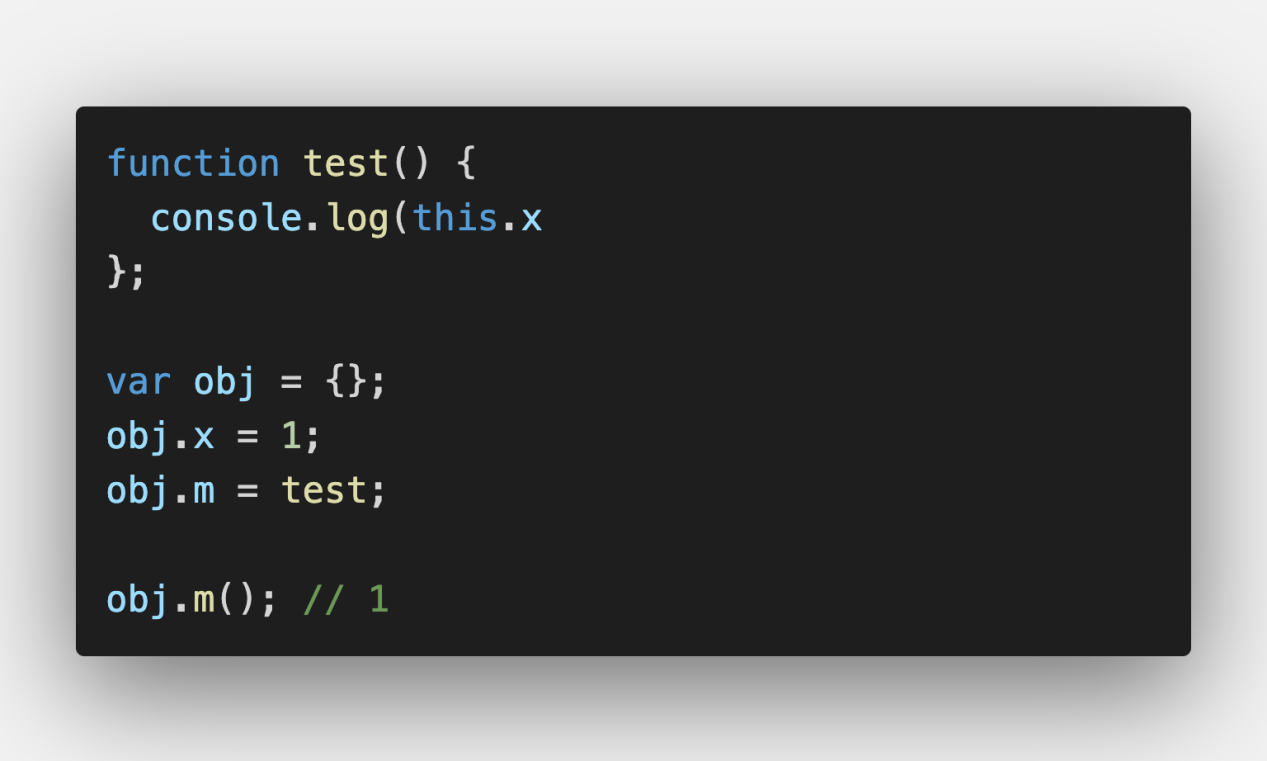
默认绑定

严格模式下，不能将全局对象用于默认绑定，this会绑定到undefined，只有函数运行在非严格模式下，默认绑定才能绑定到全局对象。



### 隐式绑定

函数作为对象方法调用时，这时this指向这个上级对象。



当函数中包含多个对象时，this指向它的上一级对象。

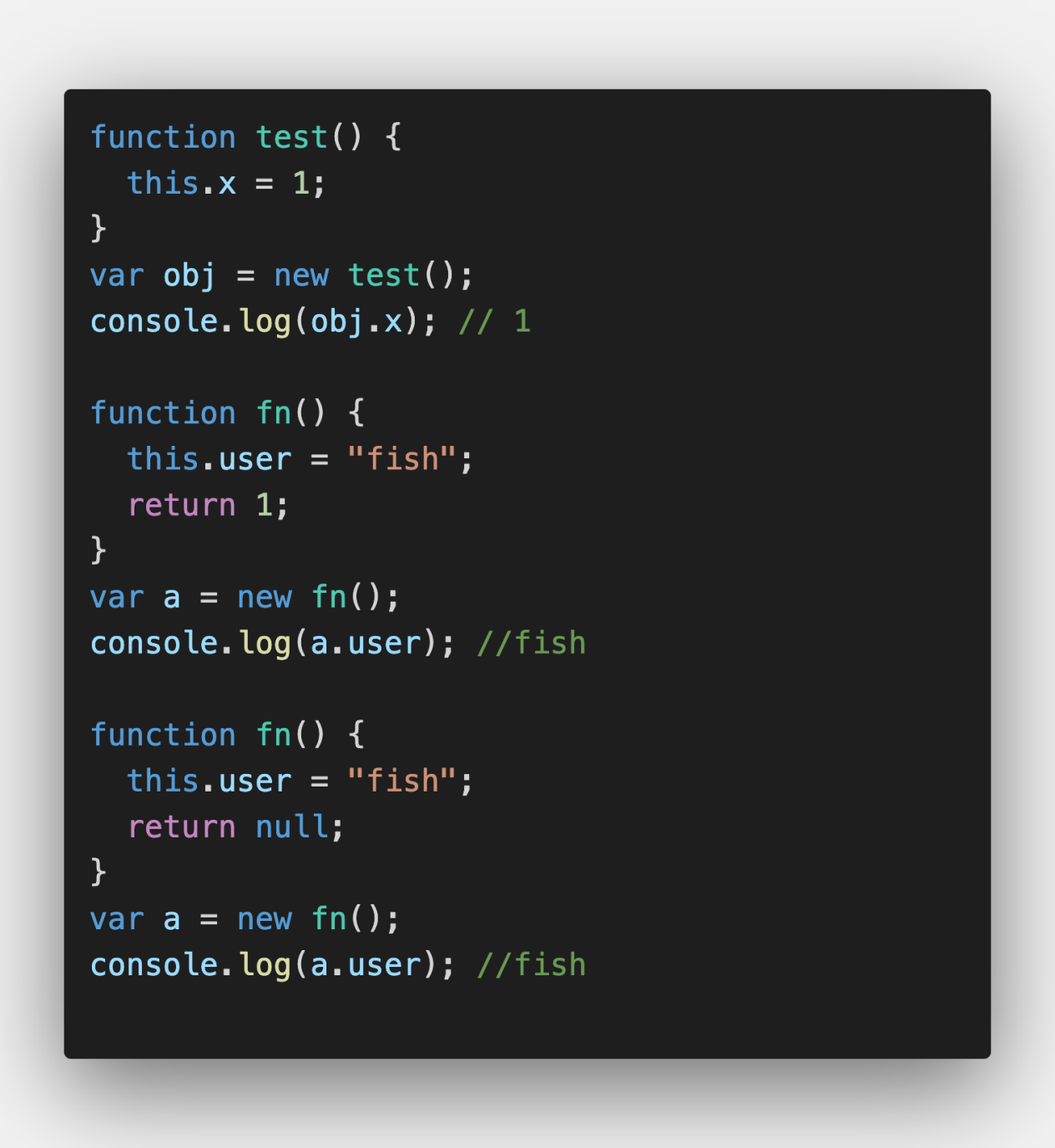


this永远指向调用它的对象



### new绑定

通过new关键字生成一个实例对象，此时this指向这个实例对象。



当new过程中遇到return一个对象，此时指向函数返回的对象。



### 显式修改



### 优先级

new绑定优先级 > 显示绑定优先级 > 隐式绑定优先级 > 默认绑定优先级

## 执行上下文与执行栈

执行上下文是对JavaScript代码执行环境的抽象概念。

执行上下文的类型分为三种：（1）全局上下文（只有一个，浏览器中的全局对象为window对象，this指向这个全局对象）；（2）函数执行上下文（存在无数个，只有在函数被调用时才会被创建，每次调用函数都会创建一个新的执行上下文）；（3）Eval函数执行上下文：指的是运行在eval函数中的代码，很少用。

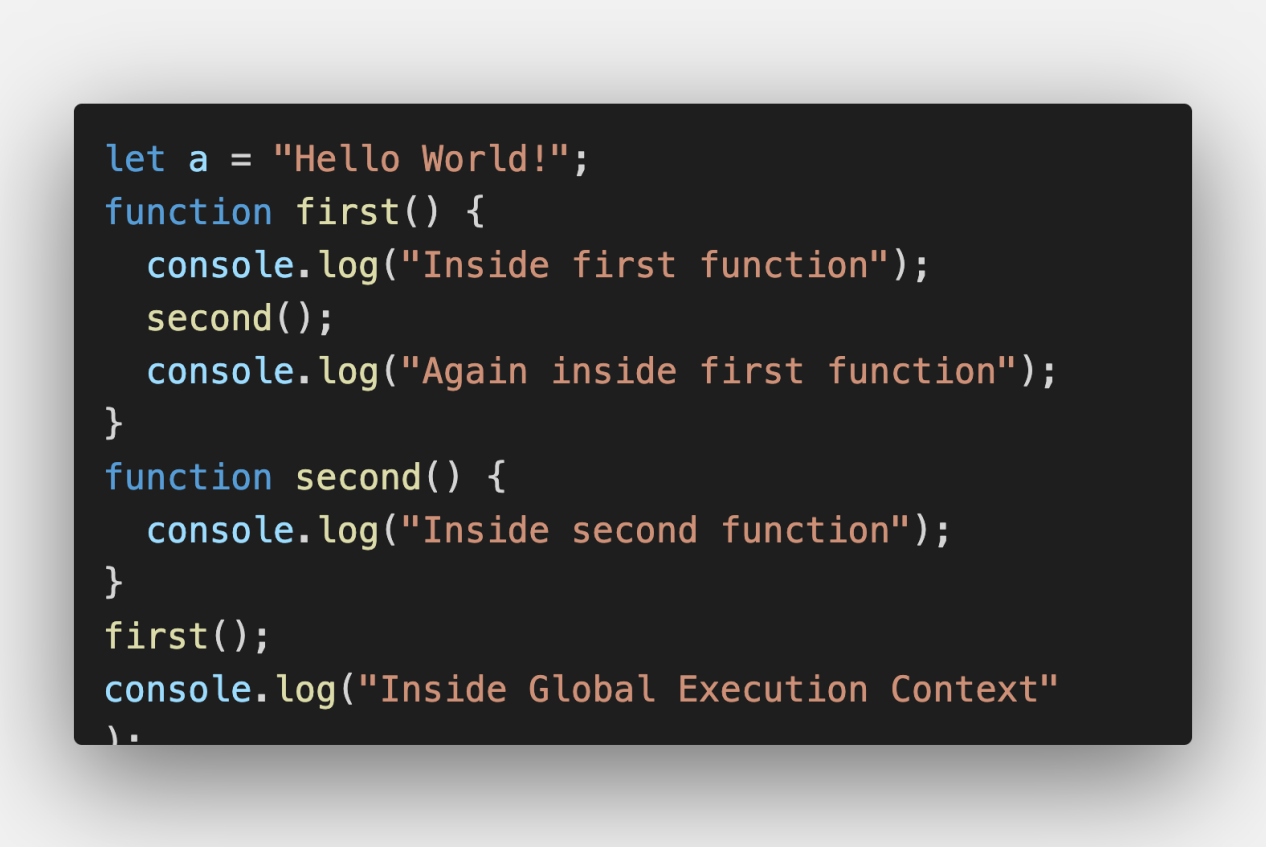
只有全局上下文（的变量）能被其他任何上下文访问。

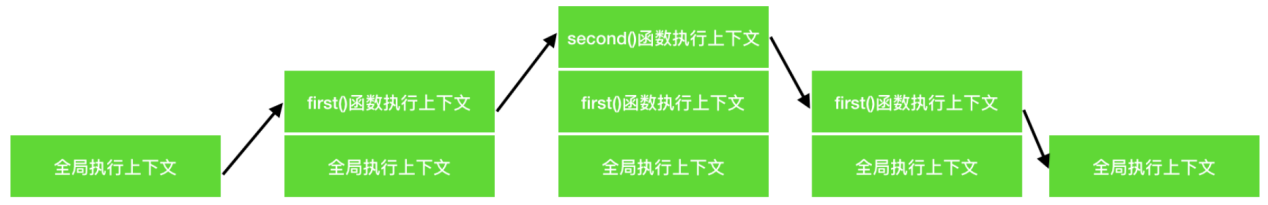
### 生命周期

执行上下文的生命周期包括三个阶段：创建阶段 --> 执行阶段 --> 回收阶段

### 执行栈

执行栈，也叫调用栈，具有 LIFO（后进先出）结构，用于存储在代码执行期间创建的所有执行上下文。





# CSS

## 谈谈你对盒子模型的理解

盒子由content（实际内容）、padding（内边距）、border（边框）、margin（外边距）四个部分组成。



盒子总宽度 = width + padding + border + margin

盒子总高度 = height + padding + border + margin

### Box-sizing

CSS中的box-sizing属性定义了引擎应该如何计算一个元素的总宽度和总高度。

content-box（默认值）: 元素的width/height不包含padding，border。

border-box：元素的 width/height 包含 padding，border。

inherit：指定 box-sizing 属性的值，应该从父元素继承。



由于box-sizing为border-box，因此盒子所占据的高度为100px，宽度为200px。

# HTTP

## 什么是HTTP，HTTP和HTTPS的区别

HTTP指超文本传输协议，是实现网络通信的一种规范。

HTTP与HTTPS区别如下：

1. HTTPS是HTTP协议的安全版本，HTTP协议的数据传输是明文的，不安全的，而HTTPS使用了SSL/TLS协议进行了加密处理，相对更安全。
2. HTTP和HTTPS使用连接方式不同，默认端口也不一样，HTTP是80，HTTPS是443。
3. HTTP由于需要设计加密以及多次握手，性能方面不如HTTP。

## UDP与TCP的区别

UDP（User Datagram Protocol）是一个简单的面向数据报的通信协议，即对应用层交下来的报文，不合并，不拆分，只是在其上面加上首部后就交给了下面的网络层。

特点：不可靠、面向无连接、丢包不重发、无顺序控制

TCP（Transmission Control Protocol）是一种可靠的、面向字节流的通信协议，把上面应用层交下来的数据看成是无结构的字节流来发送。

特点：可靠、面向有连接、丢包重发、顺序控制



# 操作系统

## 什么是操作系统

操作系统是运行在计算机上的软件程序，用于管理计算机硬件和软件资源。

## 什么是进程？什么是线程？两者区别？

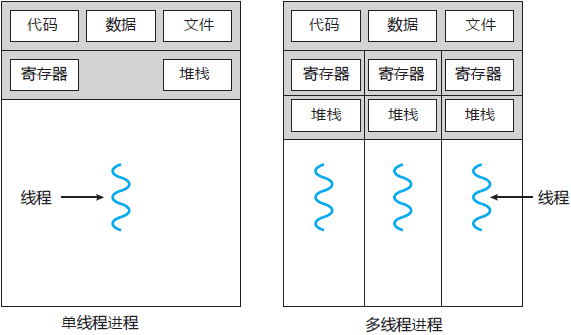
进程是程序的实例，是系统进行**资源分配**和调度的基本单位。

进程由程序、数据集合和进程控制块三部分组成：

1. 程序用于进程要完成的功能，是控制进程执行的指令集
2. 数据集合是程序在执行时所需要的数据和工作区
3. 程序控制块包含进程的描述信息和控制信息，是进程存在的唯一标志

线程是操作系统能够进行**任务运算调度**的最小单位，是进程中的一个执行单元。

进程可以只有一个线程，也可以运行多个线程



# React

## React是什么？有哪些特性？

React是用于构建用户界面的JavaScript库，只提供了UI层面的解决方案。

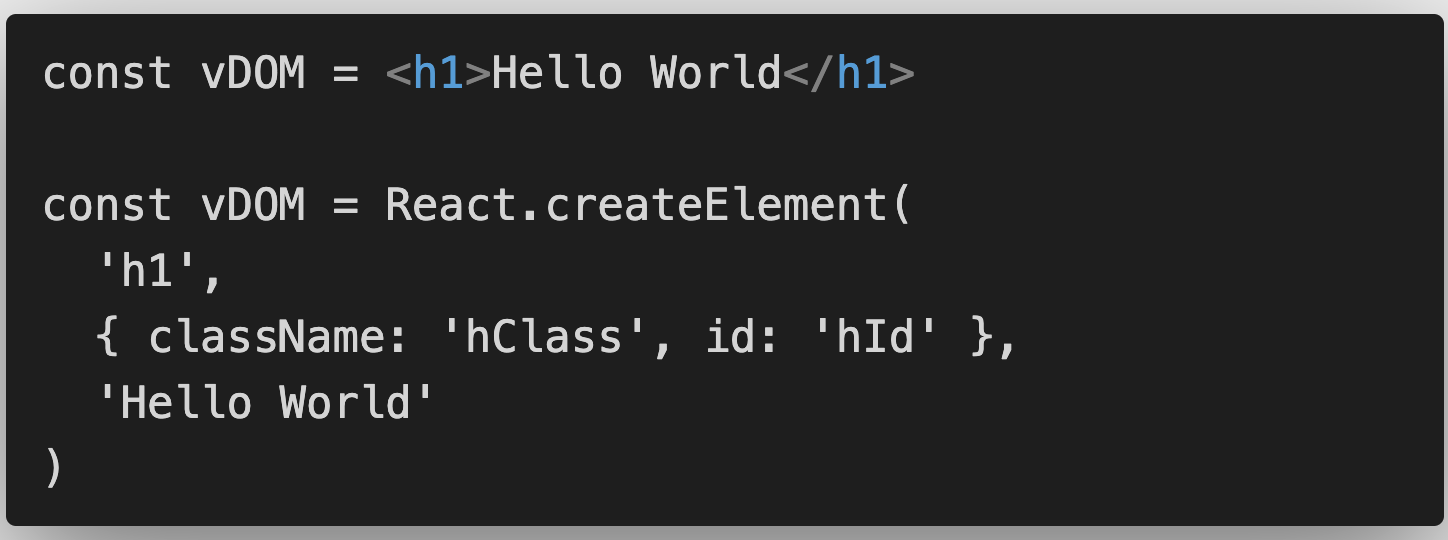
React特性：JSX语法、单向数据绑定，虚拟DOM、声明式编程、Component。

## Real DOM和Virtual DOM的区别？

Real DOM



Virtual DOM

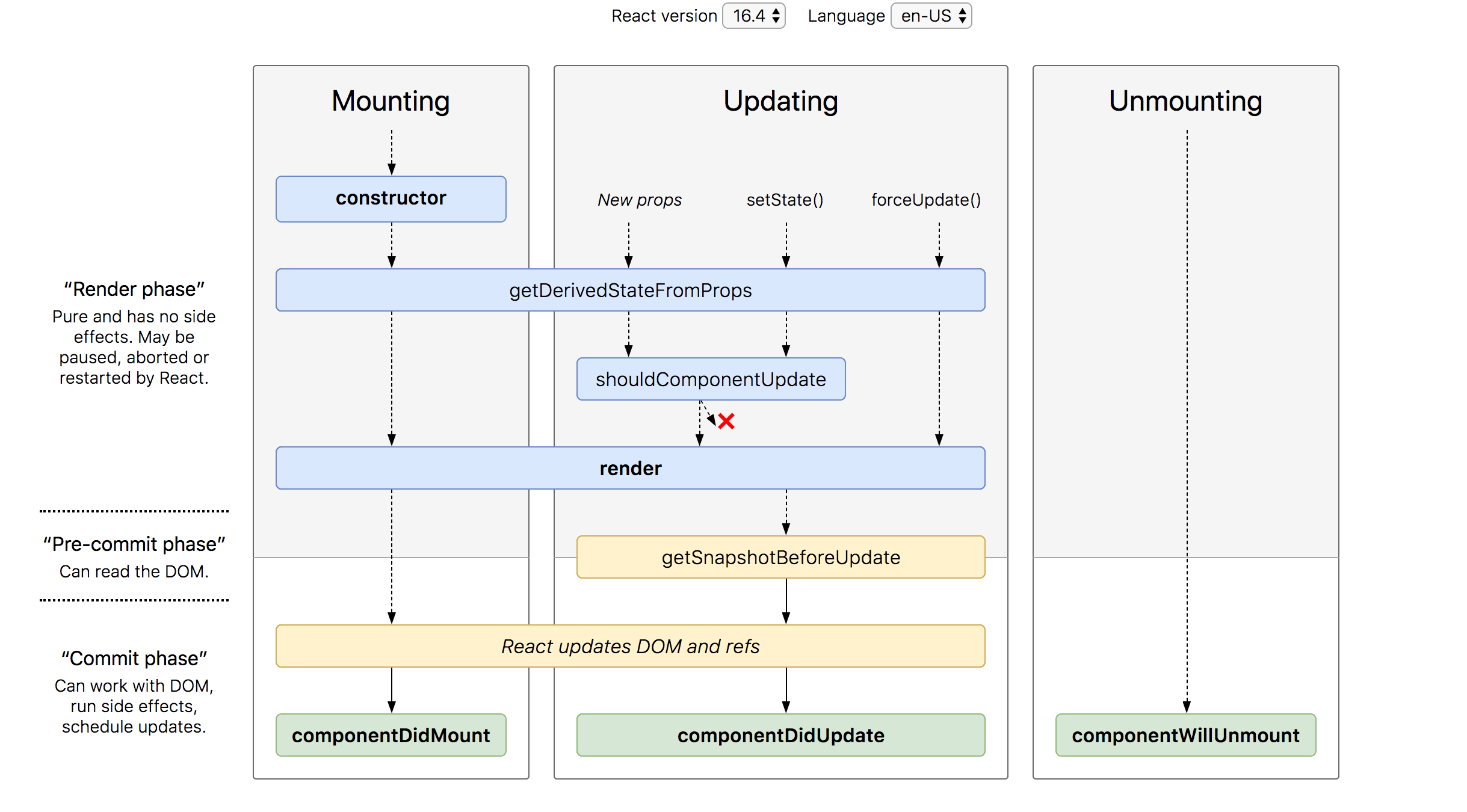


两者区别：

1. 虚拟DOM不会进行排版与重绘操作，而真实DOM会频繁重排与重绘
2. 虚拟DOM的总损耗是“虚拟DOM增删改+真实DOM差异增删改+排版与重绘”，真实DOM的总损耗是“真实DOM完全增删改+排版与重绘”

## React生命周期

React的生命周期可以分为三个阶段：创建阶段、更新阶段和卸载阶段。



创建阶段方法：constructor、getDerivedStateFromProps、render、componentDidMount

更新阶段方法：getDerivedStateFromProps、shouldComponentUpdate、render、getSnapshotBeforeUpdate、componentDidUpdate

卸载阶段方法：componentWillUnmount

## state和props的区别

相同点：

1. 两者都是JavaScript对象
2. 两者都是用于保存信息
3. props和state都能触发渲染更新

区别：

1. props由外部传递给组件，state由组件内部自行管理，一般在constructor中初始化
2. props在组件内部不可修改，state在组件内部可以修改

## super()？

super代替的是父类的构建函数，使用super(name)相当于调用sup.prototype.constructor.call(this.name)。

super()将父类中的this对象继承给子类的，没有super()子类就没有this对象。因此在子类constructor中，必须先调用super后才能引用this。