

JavaScript

JavaScript（简称“JS”）是一种具有函数优先的轻量级，解释型或即时编译型的编程语言。



2021.10

目录

[常用知识点 2](#_Toc85816741)

[数据类型 2](#_Toc85816742)

[类型判断 2](#_Toc85816743)

[JavaScript 参考手册 2](#_Toc85816744)

[Array 2](#_Toc85816745)

[构造器 2](#_Toc85816746)

[静态属性 2](#_Toc85816747)

[静态方法 2](#_Toc85816748)

[实例属性 3](#_Toc85816749)

[实例方法 3](#_Toc85816750)

[Boolean 布尔值 8](#_Toc85816751)

[Classe 8](#_Toc85816752)

[Date 时间日期 8](#_Toc85816753)

[Error 错误 8](#_Toc85816754)

[Global 全局 8](#_Toc85816755)

[JSON 8](#_Toc85816756)

[Math 算数 8](#_Toc85816757)

[属性 8](#_Toc85816758)

[方法 9](#_Toc85816759)

[Number - 数值 12](#_Toc85816760)

[RegExp 12](#_Toc85816761)

[构造函数 12](#_Toc85816762)

[静态属性 12](#_Toc85816763)

[实例属性 12](#_Toc85816764)

[实例方法 12](#_Toc85816765)

[String – 字符串 13](#_Toc85816766)

[运算符 13](#_Toc85816767)

[语句 13](#_Toc85816768)

[HTML DOM 参考手册 13](#_Toc85816769)

[Attributes 13](#_Toc85816770)

[Document 13](#_Toc85816771)

[Element 13](#_Toc85816772)

[Events 13](#_Toc85816773)

[Event 13](#_Toc85816774)

[HTMLCollection 对象 13](#_Toc85816775)

[Location 13](#_Toc85816776)

[Navigator 13](#_Toc85816777)

[Screen 13](#_Toc85816778)

[Style 13](#_Toc85816779)

[Window 13](#_Toc85816780)

[Web API 14](#_Toc85816781)

[Console – 控制台 14](#_Toc85816782)

[Geolocation 14](#_Toc85816783)

[History – 操作历史 14](#_Toc85816784)

[Storage – 存储 14](#_Toc85816785)

# 常用知识点

### 数据类型

Null

Undefined

Array

Boolean

Date

String

Number

Object

Function

Bigint

### 类型判断

### 类型转换

### this

### 闭包（closure）

一个函数和对其周围状态（lexical environment，词法环境）的引用捆绑在一起（或者说函数被引用包围），这样的组合就是闭包（closure）。

也就是说，闭包让你**可以在一个内层函数中访问到其外层函数的作用域**。在 JavaScript 中，每当创建一个函数，闭包就会在函数创建的同时被创建出来。

### 作用域

### 变量提升

### new

### call、apply、bind

#### call

#### apply

#### bind

#### 对比

#### 实现

### 原型与原型链

### Class

### 继承

### 模块化

### Promise

### 迭代器与生成器

### async await

#### async

async函数是使用async关键字声明的函数。 async函数是AsyncFunction构造函数的实例， 并且其中允许使用await关键字。async和await关键字让我们可以用一种更简洁的方式写出基于Promise的异步行为，而无需刻意地链式调用promise。

#### await

await 操作符用于等待一个Promise 对象。它只能在异步函数 async function 中使用。

### 事件循环机制（event loop）

### 防抖与节流

### 函数柯里化

### 垃圾回收

### 事件冒泡、捕获、委托

#### 事件冒泡

#### 事件捕获

#### 事件委托

### 其他零散但重要的知识点

# 基本概念

# JavaScript 参考手册

## Array

### 构造器

#### Array()

创建一个新的 Array 对象

### 静态属性

#### get Array[@@species]

返回 Array 的构造函数

### 静态方法

#### Array.from()

从类数组对象或者可迭代对象中创建一个新的数组实例

#### Array.isArray()

用来判断某个变量是否是一个数组对象

#### Array.of()

根据一组参数来创建新的数组实例，支持任意的参数数量和类型

### 实例属性

#### Array.prototype.length

数组中的元素个数

#### Array.prototype[@@unscopables]

包含了所有 ES2015 (ES6) 中新定义的、且并未被更早的 ECMAScript 标准收纳的属性名。这些属性被排除在由 with 语句绑定的环境中

### 实例方法

#### Array.prototype.at() (en-US)

Returns the array item at the given index. Accepts negative integers, which count back from the last item.

#### Array.prototype.concat()

用于合并两个或多个数组。此方法不会更改现有数组，而是返回一个新数组

#### Array.prototype.copyWithin()

浅复制数组的一部分到同一数组中的另一个位置，并返回它，不会改变原数组的长度

#### Array.prototype.entries()

返回一个新的 Array Iterator 对象，该对象包含数组中每个索引的键/值对

#### Array.prototype.every()

测试一个数组内的所有元素是否都能通过某个指定函数的测试。它返回一个布尔值

#### Array.prototype.fill()

用一个固定值填充一个数组中从起始索引到终止索引内的全部元素

#### Array.prototype.filter()

创建一个新数组, 其包含通过所提供函数实现的测试的所有元素

#### Array.prototype.find()

返回数组中满足提供的测试函数的第一个元素的值。否则返回 undefined

#### Array.prototype.findIndex()

返回数组中满足提供的测试函数的第一个元素的索引。若没有找到对应元素则返回 -1

#### Array.prototype.flat()

按照一个可指定的深度递归遍历数组，并将所有元素与遍历到的子数组中的元素合并为一个新数组返回

#### Array.prototype.flatMap()

使用映射函数映射每个元素，然后将结果压缩成一个新数组

#### Array.prototype.forEach()

对数组的每个元素执行一次给定的函数

#### Array.prototype.includes()

判断一个数组是否包含一个指定的值，如果包含则返回 true，否则返回 false

#### Array.prototype.indexOf()

返回在数组中可以找到一个给定元素的第一个索引，如果不存在，则返回 -1

#### Array.prototype.join()

将一个数组的所有元素连接成一个字符串并返回这个字符串

#### Array.prototype.keys()

返回一个包含数组中每个索引键的 Array Iterator 对象

#### Array.prototype.lastIndexOf()

返回指定元素在数组中的最后一个的索引，如果不存在则返回 -1

#### Array.prototype.map()

返回一个新数组，其结果是该数组中的每个元素是调用一次提供的函数后的返回值

#### Array.prototype.pop()

从数组中删除最后一个元素，并返回该元素的值

#### Array.prototype.push()

将一个或多个元素添加到数组的末尾，并返回该数组的新长度

#### Array.prototype.reduce()

对数组中的每个元素执行一个由您提供的reducer函数（升序执行），将其结果汇总为单个返回值

#### Array.prototype.reduceRight()

接受一个函数作为累加器（accumulator）和数组的每个值（从右到左）将其减少为单个值

#### Array.prototype.reverse()

将数组中元素的位置颠倒，并返回该数组。该方法会改变原数组

#### Array.prototype.shift()

从数组中删除第一个元素，并返回该元素的值

#### Array.prototype.slice()

提取源数组的一部分并返回一个新数组

#### Array.prototype.some()

测试数组中是不是至少有一个元素通过了被提供的函数测试

#### Array.prototype.sort()

对数组元素进行原地排序并返回此数组

#### Array.prototype.splice()

通过删除或替换现有元素或者原地添加新的元素来修改数组,并以数组形式返回被修改的内容

#### Array.prototype.toLocaleString()

返回一个字符串表示数组中的元素。数组中的元素将使用各自的 Object.prototype.toLocaleString() 方法转成字符串

#### Array.prototype.toString()

返回一个字符串表示指定的数组及其元素。数组中的元素将使用各自的 Object.prototype.toString() 方法转成字符串

#### Array.prototype.unshift()

将一个或多个元素添加到数组的头部，并返回该数组的新长度

#### Array.prototype.values()

返回一个新的 Array Iterator 对象，该对象包含数组每个索引的值

#### Array.prototype[@@iterator]()

返回一个新的 Array Iterator 对象，该对象包含数组每个索引的值

## Boolean 布尔值

## Classe

## Date 时间日期

## Error 错误

## Global 全局

## JSON

## Math 算数

### 属性

#### Math.E

欧拉常数，也是自然对数的底数，约等于 2.718。

#### Math.LN2

2 的自然对数，约等于 0.693。

#### Math.LN10

10 的自然对数，约等于 2.303。

#### Math.LOG2E

以 2 为底的 E 的对数，约等于 1.443。

#### Math.LOG10E

以 10 为底的 E 的对数，约等于 0.434。

#### Math.PI

圆周率，一个圆的周长和直径之比，约等于 3.14159。

#### Math.SQRT1\_2

二分之一 ½ 的平方根，同时也是 2 的平方根的倒数 12，约等于 0.707。

#### Math.SQRT2

2 的平方根，约等于 1.414。

### 方法

#### Math.abs(x)

返回一个数的绝对值。

#### Math.acos(x)

返回一个数的反余弦值。

#### Math.acosh(x)

返回一个数的反双曲余弦值。

#### Math.asin(x)

返回一个数的反正弦值。

#### Math.asinh(x)

返回一个数的反双曲正弦值。

#### Math.atan(x)

返回一个数的反正切值。

#### Math.atanh(x)

返回一个数的反双曲正切值。

#### Math.atan2(y, x)

返回 y/x 的反正切值。

#### Math.cbrt(x)

返回一个数的立方根。

#### Math.ceil(x)

返回大于一个数的最小整数，即一个数向上取整后的值。

#### Math.clz32(x)

返回一个 32 位整数的前导零的数量。

#### Math.cos(x)

返回一个数的余弦值。

#### Math.cosh(x)

返回一个数的双曲余弦值。

#### Math.exp(x)

返回欧拉常数的参数次方，Ex，其中 x 为参数，E 是欧拉常数（2.718...，自然对数的底数）。

#### Math.expm1(x)

返回 exp(x) - 1 的值。

#### Math.floor(x)

返回小于一个数的最大整数，即一个数向下取整后的值。

#### Math.fround(x)

返回最接近一个数的单精度浮点型表示。

#### Math.hypot([x[, y[, …]]])

返回其所有参数平方和的平方根。

#### Math.imul(x, y)

返回 32 位整数乘法的结果。

#### Math.log(x)

返回一个数的自然对数（㏒e，即 ㏑）。

#### Math.log1p(x)

返回一个数加 1 的和的自然对数（㏒e，即 ㏑）。

#### Math.log10(x)

返回一个数以 10 为底数的对数。

#### Math.log2(x)

返回一个数以 2 为底数的对数。

#### Math.max([x[, y[, …]]])

返回零到多个数值中最大值。

#### Math.min([x[, y[, …]]])

返回零到多个数值中最小值。

#### Math.pow(x, y)

返回一个数的 y 次幂。

#### Math.random()

返回一个 0 到 1 之间的伪随机数。

#### Math.round(x)

返回四舍五入后的整数。

#### Math.sign(x)

返回一个数的符号，得知一个数是正数、负数还是 0。

#### Math.sin(x)

返回一个数的正弦值。

#### Math.sinh(x)

返回一个数的双曲正弦值。

#### Math.sqrt(x)

返回一个数的平方根。

#### Math.tan(x)

返回一个数的正切值。

#### Math.tanh(x)

返回一个数的双曲正切值。

#### Math.toSource()

返回字符串 "Math"。

#### Math.trunc(x)

返回一个数的整数部分，直接去除其小数点及之后的部分。

## Number - 数值

## RegExp

### 构造函数

#### RegExp()

创建一个新的 RegExp 对象。

### 静态属性

#### get RegExp[@@species]

该构造函数用于创建派生对象。

#### RegExp.lastIndex

该索引表示从哪里开始下一个匹配

### 实例属性

#### RegExp.prototype.flags

含有 RegExp 对象 flags 的字符串。

#### RegExp.prototype.dotAll

. 是否要匹配新行（newlines）。

#### RegExp.prototype.global

针对字符串中所有可能的匹配项测试正则表达式，还是仅针对第一个匹配项。

#### RegExp.prototype.ignoreCase

匹配文本的时候是否忽略大小写。

#### RegExp.prototype.multiline

是否进行多行搜索。

#### RegExp.prototype.source

正则表达式的文本。

#### RegExp.prototype.sticky

搜索是否是 sticky。

#### RegExp.prototype.unicode

Unicode 功能是否开启。

### 实例方法

#### RegExp.prototype.compile()

运行脚本的期间（重新）编译正则表达式。

#### RegExp.prototype.exec()

在该字符串中执行匹配项的搜索。

#### RegExp.prototype.test()

该正则在字符串里是否有匹配。

#### RegExp.prototype[@@match]()

对给定字符串执行匹配并返回匹配结果。

#### RegExp.prototype[@@matchAll]()

对给定字符串执行匹配，返回所有匹配结果。

#### RegExp.prototype[@@replace]()

给定新的子串，替换所有匹配结果。

#### RegExp.prototype[@@search]()

在给定字符串中搜索匹配项，并返回在字符串中找到字符索引。

#### RegExp.prototype[@@split]()

通过将给定字符串拆分为子字符串，并返回字符串形成的数组。

#### RegExp.prototype.toString()

返回表示指定对象的字符串。重写Object.prototype.toString()方法。

## String

## 运算符

### 算术运算符

算术运算符用于在变量和/或值之间执行算术。

* +
* -
* \*
* /
* %
* ++
* --

### 赋值运算符

赋值运算符用于为 JavaScript 变量赋值。

### 字符串运算符

+ 运算符和 += 运算符还可用于连接（添加）字符串。

### 比较运算符

在逻辑语句中使用比较运算符来确定变量或值之间的相等性或差异。

### 条件（三元）运算符

条件运算符根据条件将值赋给变量。

### 逻辑运算符

逻辑运算符用于确定变量或值之间的逻辑。

### 位运算符

位运算符可处理 32 位数字。该运算中的任何数字操作数都将转换为 32 位数字。结果将转换回 JavaScript 数字。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算符 | 描述 | 例子 | 等同于 | 结果 | 十进制 |
| & | AND | x = 5 & 1 | 0101 & 0001 | 0001 | 1 |
| | | OR | x = 5 | 1 | 0101 | 0001 | 0101 | 5 |
| ~ | NOT | x = ~ 5 | ~0101 | 1010 | 10 |
| ^ | XOR | x = 5 ^ 1 | 0101 ^ | 0100 | 4 |
| << | Left shift | x = 5 << 1 | 0101 << 1 | 1010 | 10 |
| >> | Right shift | x = 5 >> 1 | 0101 >> 1 | 0010 | 2 |

### typeof运算符

typeof 运算符返回变量、对象、函数或表达式的类型：

### delete运算符

delete 运算符从对象中删除属性：

### in运算符

如果指定的属性在指定的对象中，则 in 运算符将返回 true，否则返回 false：

### instanceof运算符

如果指定对象是指定对象的实例，则 instanceof 运算符返回 true：

### void 运算符

void 运算符计算一个表达式并返回 undefined。该运算符通常用于使用 "void(0)" 来获取未定义的原始值（在计算表达式而不使用返回值时很有用）。

## 语句

### break

退出 switch 或循环。

### const

声明具有恒定值的变量。

### class

声明类。

### continue

如果出现指定条件，则（在循环中）中断一次循环，并继续循环中的下一次迭代。

### debugger

停止执行 JavaScript，并调用调试功能（如果有）。

### do ... while

执行一个语句块，并在条件为 true 时重复该语句块。

### for

循环遍历代码块多次.

### for ... in

循环遍历对象的属性。

### for ... of

循环遍历可迭代对象的值。

### function

声明函数。

### if ... else ... else if

标记根据条件执行的语句块。

### let

声明在方括号 {} 作用域内的变量。

### return

停止执行函数并从该函数返回值。

### switch

根据不同情况标记要执行的语句块。

### throw

抛出（生成）错误。

### try ... catch ... finally

标记在 try 块中发生错误时要执行的语句块，并实现错误处理。

### var

声明变量。

### while

标记条件为 true 时要执行的语句块。

# HTML DOM 参考手册

HTML 文档对象模型

## Attributes

## Document

## Element

## Events

## Event

## HTMLCollection 对象

# BOM

浏览器对象模型

## Window

## Location

## Navigator

## Screen

## Style

# Web API

## Console – 控制台

## Geolocation

## History – 操作历史

## Storage – 存储

# 参考

1. JavaScript | MDN (<https://developer.mozilla.org/zh-CN/docs/Web/JavaScript>)
2. JavaScript | W3school (<https://www.w3school.com.cn/js/index.asp>)
3. JavaScript高级程序设计（第3版）
4. JavaScript高级程序设计（第 4 版）
5. JavaScript权威指南
6. JavaScript 忍者秘籍（第 2 版）
7. 你不知道的 JS 三卷
8. JavaScript 悟道
9. JavaScript 语言精髓与编程实践（第 3 版）