# 正则表达式

正则表达式（英语：Regular Expression，在代码中常简写为regex、regexp或RE）使用单个字符串来描述、匹配一系列符合某个句法规则的字符串搜索模式。搜索模式可用于文本搜索和文本替换。

什么是正则表达式？

正则表达式是由一个字符序列形成的搜索模式。

当你在文本中搜索数据时，你可以用搜索模式来描述你要查询的内容。

正则表达式可以是一个简单的字符，或一个更复杂的模式。

正则表达式可用于所有文本搜索和文本替换的操作。

## 语法

/正则表达式主体/修饰符(可选)

实例解析：

/runoob/i 是一个正则表达式。

runoob 是一个正则表达式主体 (用于检索)。

i 是一个修饰符 (搜索不区分大小写)。

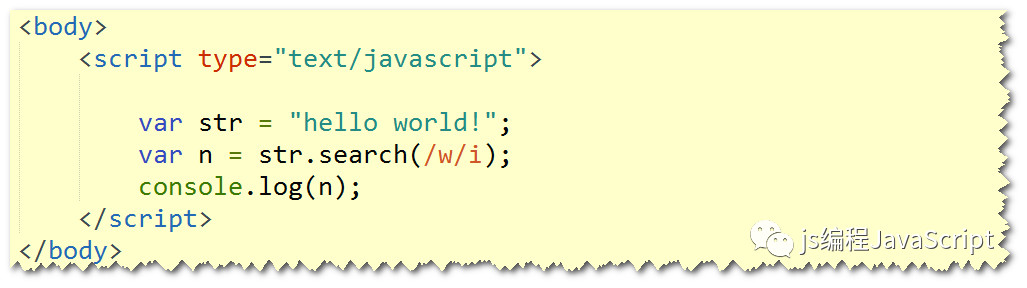
## 使用字符串方法

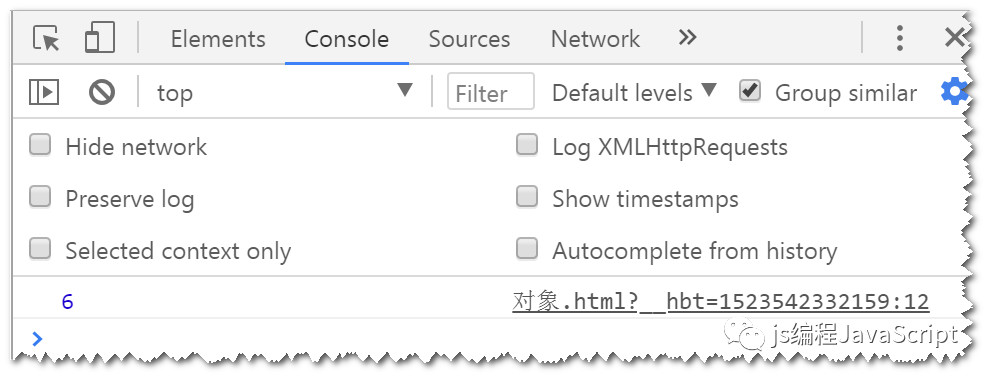
在 JavaScript 中，正则表达式通常用于两个字符串方法 : search() 和 replace()。

search() 方法 用于检索字符串中指定的子字符串，或检索与正则表达式相匹配的子字符串，并返回子串的起始位置。

replace() 方法 用于在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子串。

search() 方法使用正则表达式



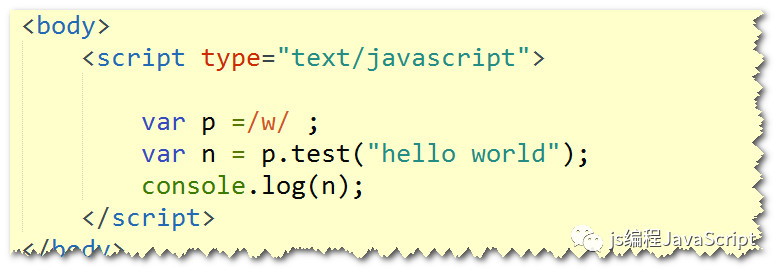


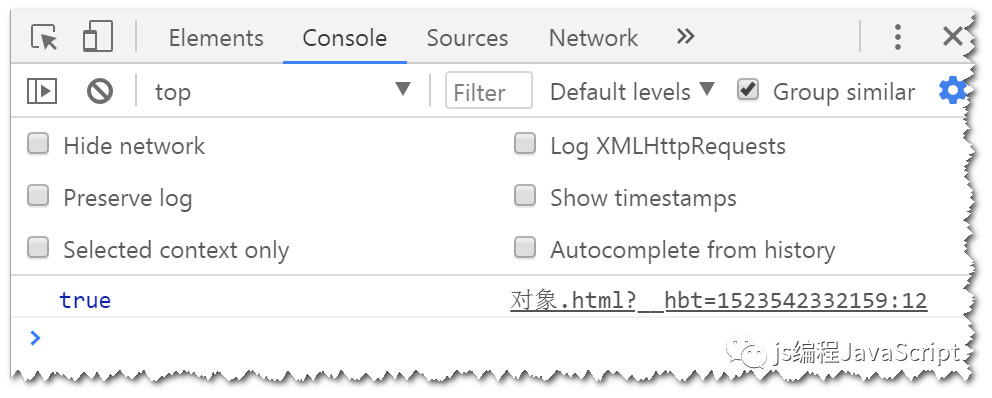
i 执行对大小写不敏感的匹配。



## 使用 test()

test() 方法是一个正则表达式方法。





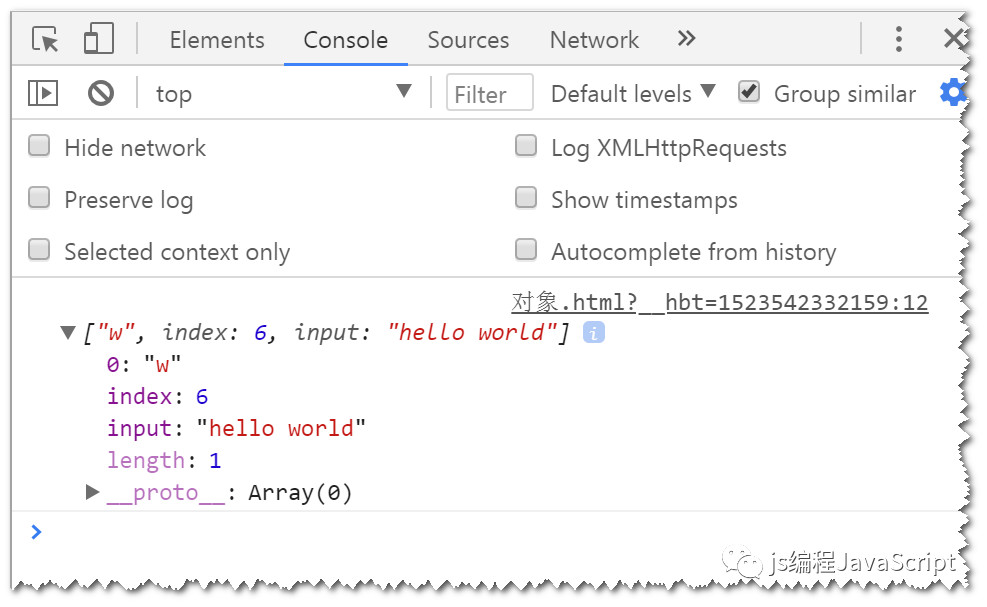
test() 方法用于检测一个字符串是否匹配某个模式，如果字符串中含有匹配的文本，则返回 true，否则返回 false。

## 使用 exec()

exec() 方法是一个正则表达式方法。



exec() 方法用于检索字符串中的正则表达式的匹配。

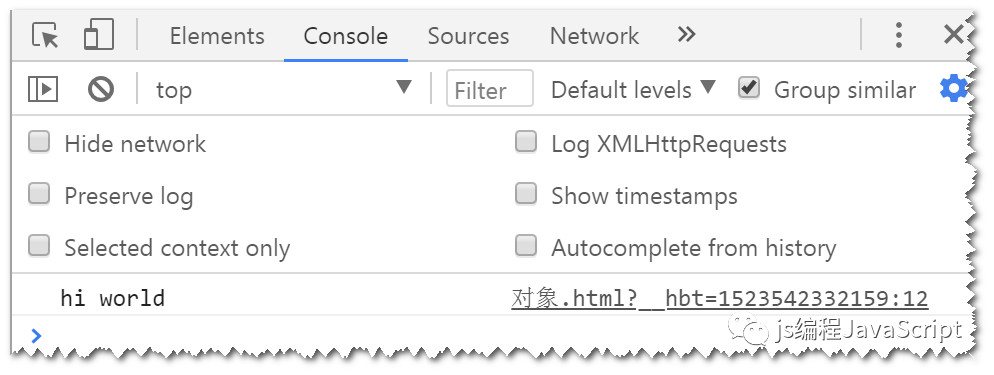


该函数返回一个数组，其中存放匹配的结果。如果未找到匹配，则返回值为 null。

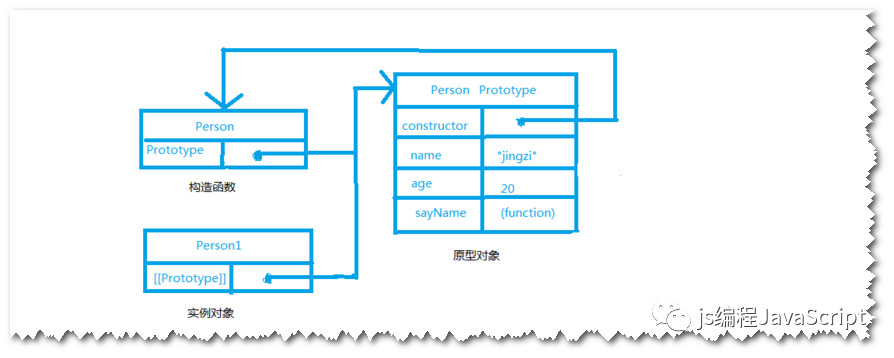
## replace()

方法使用正则表达式



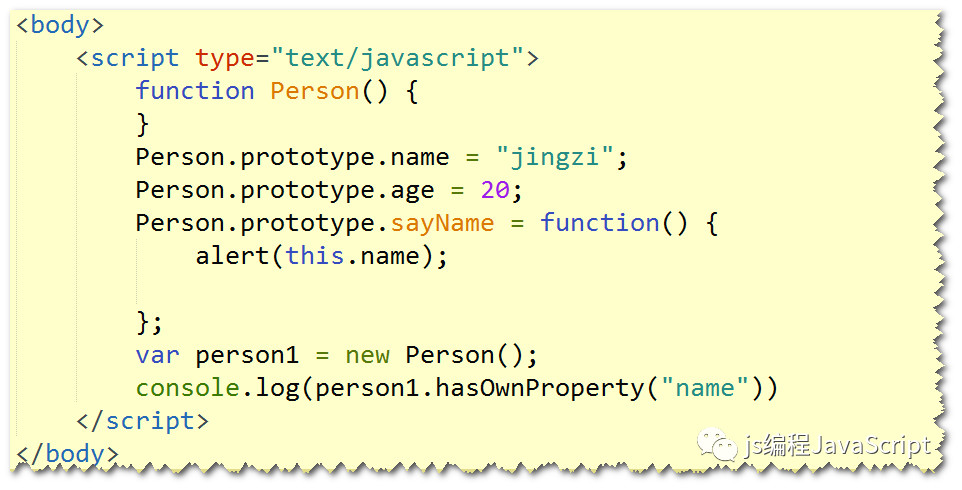


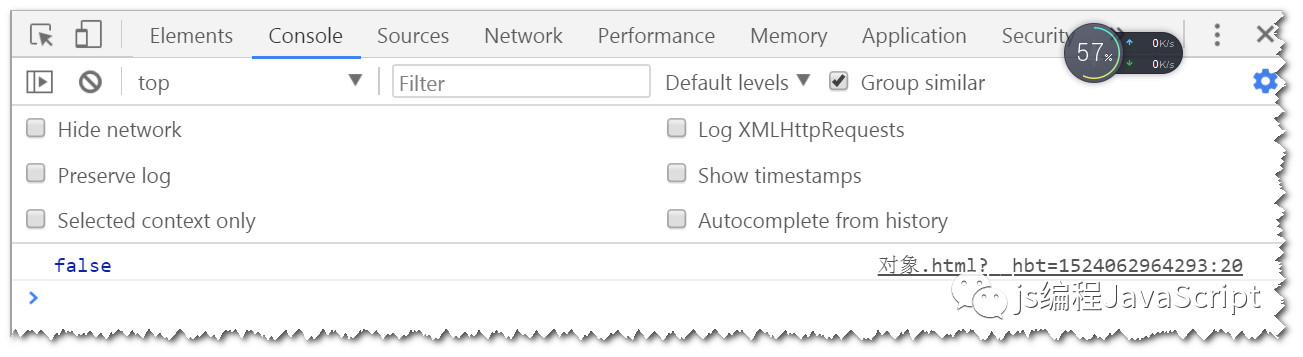
# 原型



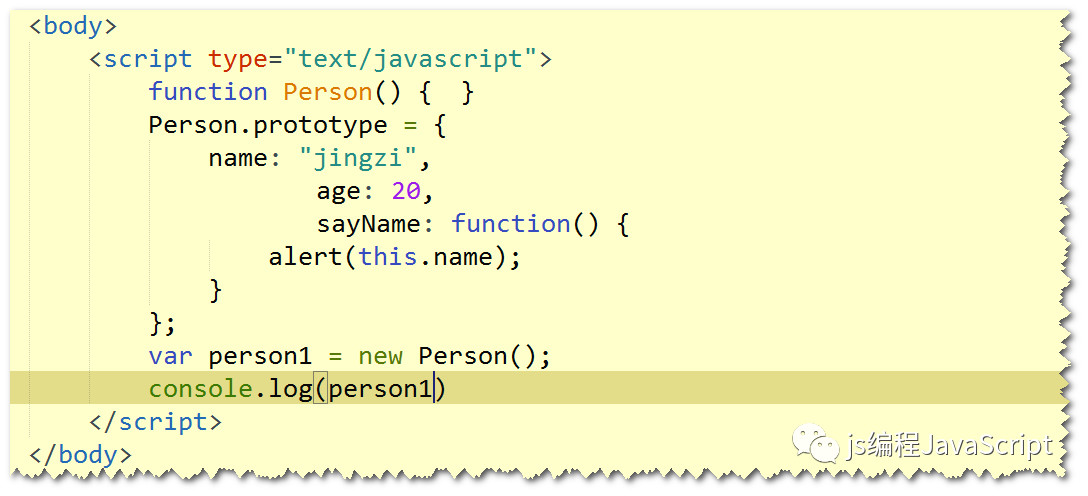
创建了一个Person函数，它就会拥有一个prototype属性，这个属性指向了Person Prototype原型对象，而这个原型对象拥有一个constructor属性，其指针指向了Person。prototype属性会指向其原型对象而不是直接指向其构造函数Person。

使用 hasOwnProperty()方法可以检测一个属性是存在于实例中，还是存在于原型中





前面例子中每添加一个属性和方法就要敲一遍 Person.prototype。为减少不必要的输入，也为了从视觉上更好地封装原型的功能，更常见的做法是用一个包含所有属性和方法的对象字面量来重写整个原型对象，如下面的例子所示



实例

function Person(gender){

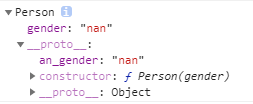
this.gender = gender;

Person.prototype.an\_gender = "nan";

}

var person = new Person("nan");

console.log(person);



## prototype和\_proto\_的区别

