对象

掌握对象的创建和使用方法

对象

掌握对象的基本语法

- **概念**
- 对象字面量
- 其他创建对象的方式
- 对象遍历

为什么要有对象？

- 如果有一组相关的数据，松散的存储不利于使用，存入数组中受下标限制又必须有固定的顺序，而对象可以自定义名称存储一系列无序的相关数据。

什么是对象？

现实生活中的对象：万物皆对象，对象是一个具体的事物，一个具体的事物就会有行为和特征。

举例： 一部车，一个手机

车是一类事物，门口停的那辆车才是对象

特征：红色、四个轮子

行为：驾驶、刹车

什么是对象？

- JavaScript 中的对象：

  JavaScript 中的对象其实就是生活中对象的一个抽象。

  JavaScript 的对象是无序属性的集合。
- 其属性可以包含基本值、对象或函数。对象就是一组没有顺序的值。我们可以把 JavaScript 中的对象想象成键值对，其中值可以是数据和函数。
- 对象的行为和特征：

  特征---在对象中用属性表示

  行为---在对象中用方法表示

对象

掌握对象的基本语法

- 概念
- **对象字面量**
- 其他创建对象的方式
- 对象遍历

对象字面量

- 创建一个对象最简单的方式是使用对象字面量赋值给变量。类似数组。
- 对象字面量语法：{}
- 内部可以存放多条数据，数据与数据之间用逗号分隔，最后一个后面不要加逗号。
- 每条数据都是有属性名和属性值组成，键值对写法：k: v
- k: 属性名
- v：属性值，可以是任意类型的数据，比如简单类型数据、函数、对象。

语法

```
var obj = {
    k: v,
    k: v,
    k: v
};
```

案例

```
var o = {
    name: 'zs',
    age: 18,
    sex: true,
    sayHi: function () {
      console.log("你好");
    }
};
```

区分属性和方法

- 属性：对象的描述性特征，一般是名词，相当于定义在对象内部的变量。
- 方法：对象的行为和功能，一般是动词，定义在对象中的函数。

调用对象内部属性和方法的语法

- 用对象的变量名打点调用某个属性名，得到属性值。
- 在对象内部用 this 打点调用属性名。this 替代对象。
- 用对象的变量名后面加 [] 调用，[] 内部是字符串格式的属性名。
- 调用方法时，需要在方法名后加 () 执行。

更改对象内部属性和方法的语法

- 更改属性的属性值方法：先调用属性，再等号赋值。

  o.age = 19;

- 增加新的属性和属性值：使用点语法或者[]方法直接定义新属性，等号赋值。

  o.height = 180;

- 删除一条属性：使用一个 delete 关键字，空格后面加属性调用。

  delete o.sex;

对象

掌握对象的基本语法

- 概念
- 对象字面量
- **其他创建对象的方式**
- 对象遍历

new Object() 创建对象

- Object() 构造函数，是一种特殊的函数。主要用来在创建对象时初始化对象， 即为对象成员变量赋初始值，总与 new 运算符一起使用在创建对象的语句中。

1. 构造函数用于创建一类对象，首字母要大写。
2. 构造函数要和 new 一起使用才有意义。

new 在执行时会做四件事情

- new 会在内存中创建一个新的空对象
- new 会让 this 指向这个新的对象
- 执行构造函数 目的：给这个新对象加属性和方法
- new 会返回这个新对象

工厂函数创建对象

- 如果要创建多个类似的对象，可以将 new Object() 过程封装到一个函数中，将来调用函数就能创建一个对象，相当于一个生产对象的函数工厂，用来简化代码。

工厂函数创建对象

```
function createPerson(name, age, job) {
  var person = new Object();
  person.name = name;
  person.age = age;
  person.job = job;
  person.sayHi = function () {
    console.log('Hello,everyBody');
  }
  return person;
}
var p1 = createPerson('张三', 22, 'actor');
var p2 = createPerson('李四', 23, 'doctor');
```

自定义构造函数

- 比工厂方法更加简单。
- 自定义一个创建具体对象的构造函数，函数内部不需要 new 一个构造函数的过程，直接使用 this 代替对象进行属性和方法的书写，也不需要 return 一个返回值。
- 使用时，利用 new 关键字调用自定义的构造函数即可。
- 注意：构造函数的函数名首字母需要大写，区别于其他普通函数名。

自定义构造函数

```
function Person(name, age, job) {
  this.name = name;
  this.age = age;
  this.job = job;
  this.sayHi = function () {
    console.log('Hello,everyBody');
  }
}
var p1 = new Person('张三', 22, 'actor');
```

对象

掌握对象的基本语法

- 概念
- 对象字面量
- 其他创建对象的方式
- **对象遍历**

对象遍历

- for in 循环也是循环的一种，专门用来遍历对象，内部会定义一个 k 变量， k 变量在每次循环时会从第一个开始接收属性名，一直接收到最后一条属性名，执行完后跳出循环。
- 简单的循环遍历：输出每一项的属性名和属性值。

```
// 循环遍历输出每一项
for(var k in obj){
        console.log(k + "项的属性值是" + obj[k]);
}
```

简单类型和复杂类型

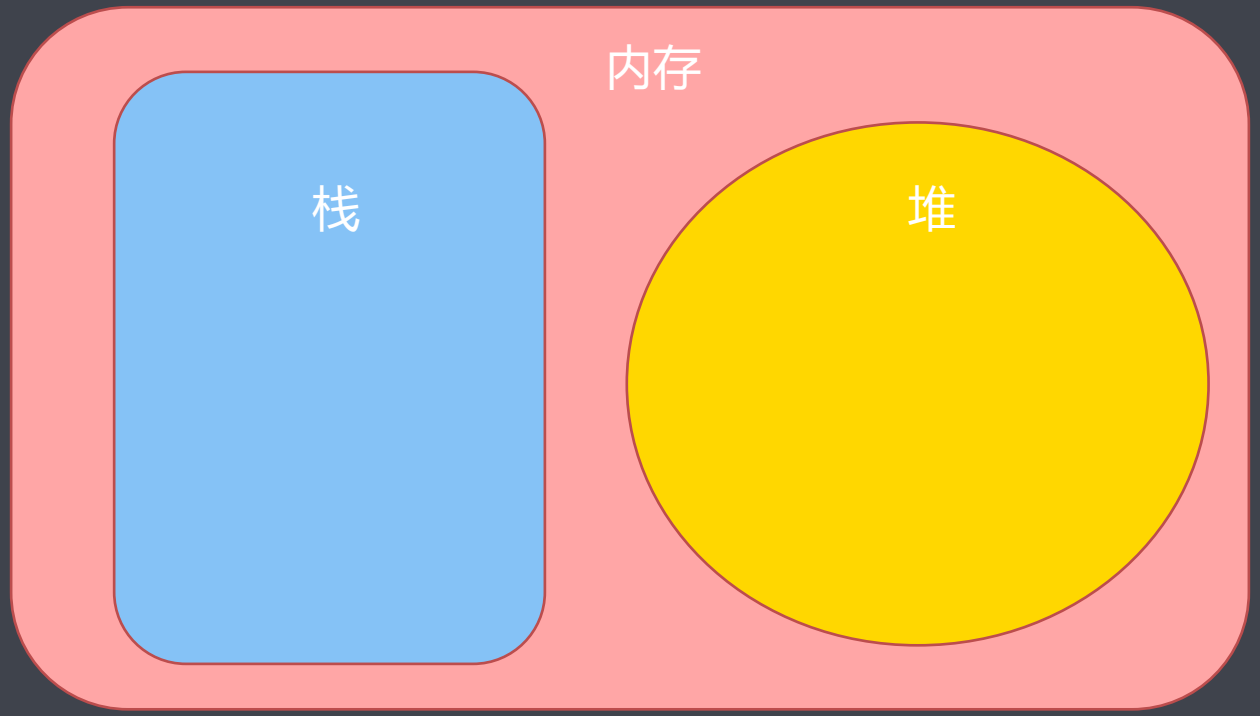
掌握简单类型和复杂类型的区别

简单类型和复杂类型的区别

- 我们已经学习过简单类型数据和一些复杂类型的数据，现在来看一下他们之间的区别有哪些。
- 基本类型又叫做值类型，复杂类型又叫做引用类型
- 值类型：简单数据类型，基本数据类型，在存储时，变量中存储的是值本身，因此叫做值类型。
- 引用类型：复杂数据类型，在存储时，变量中存储的仅仅是地址（引用），因此叫做引用数据类型。

堆和栈

- JavaScript 中没有堆和栈的概念，此处我们用堆和栈来讲解，目的是方便理解和方便以后的学习。



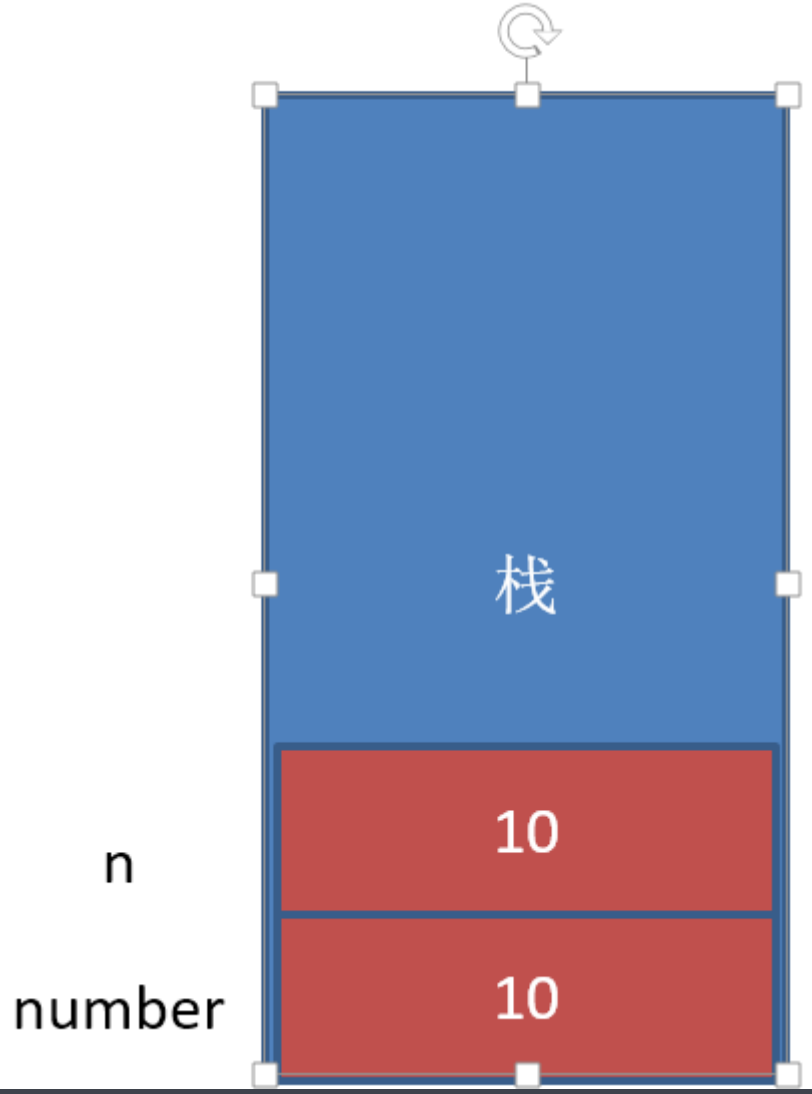
堆和栈

- 堆栈空间分配区别：

1. 栈（操作系统）：由操作系统自动分配释放，存放函数的参数值，局部变量的值等。
2. 堆（操作系统）：存储复杂类型(对象)，一般由程序员分配释放，若程序员不释放，由垃圾回收机制回收。

基本类型在内存中的存储

变量中如果存储的是简单类型的数据，那么变量中存储的是值本身，如果将变量赋值给另一个变量，是将内部的值复制一份给了另一个变量，两个变量之间没有联系，一个变化，另一个不会同时变化。



n

number

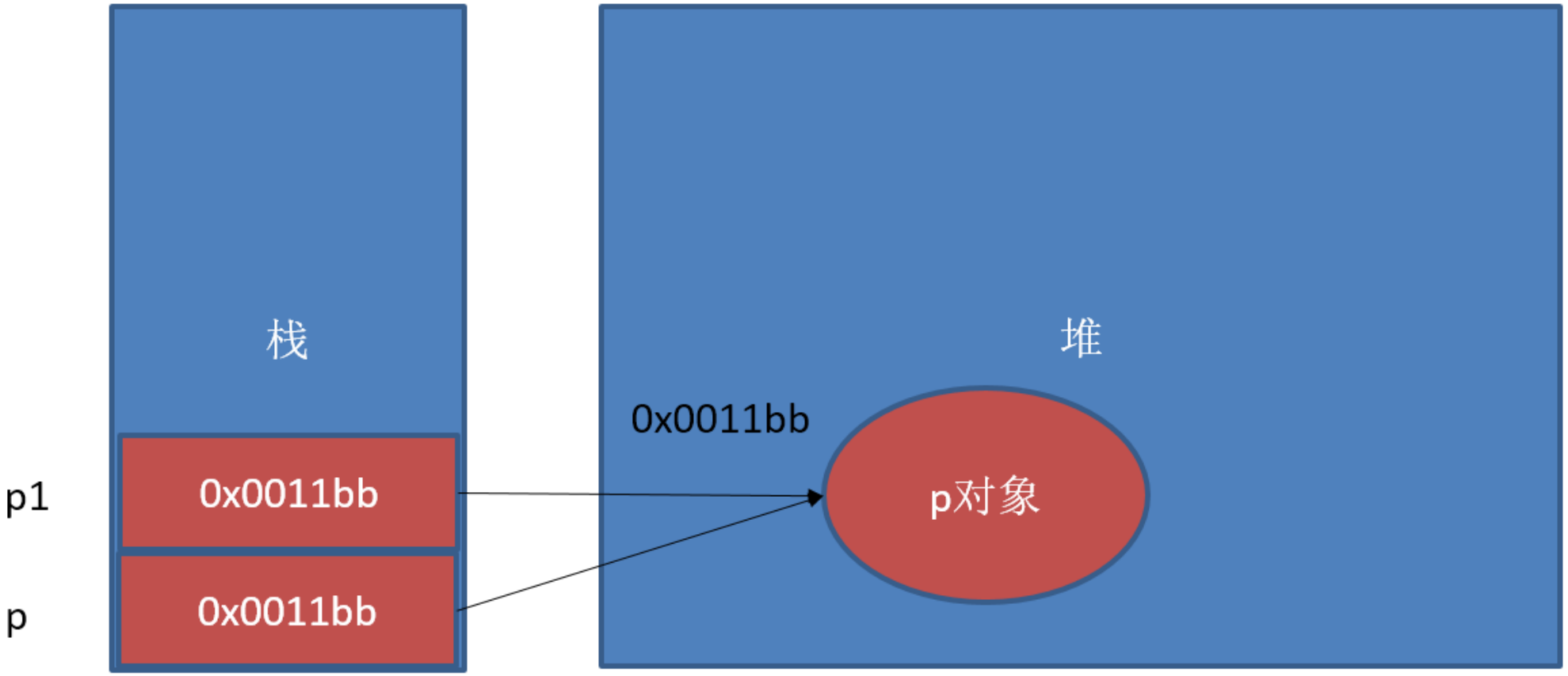
```
var number = 10;

var n = number;
```

复杂类型在内存中的存储

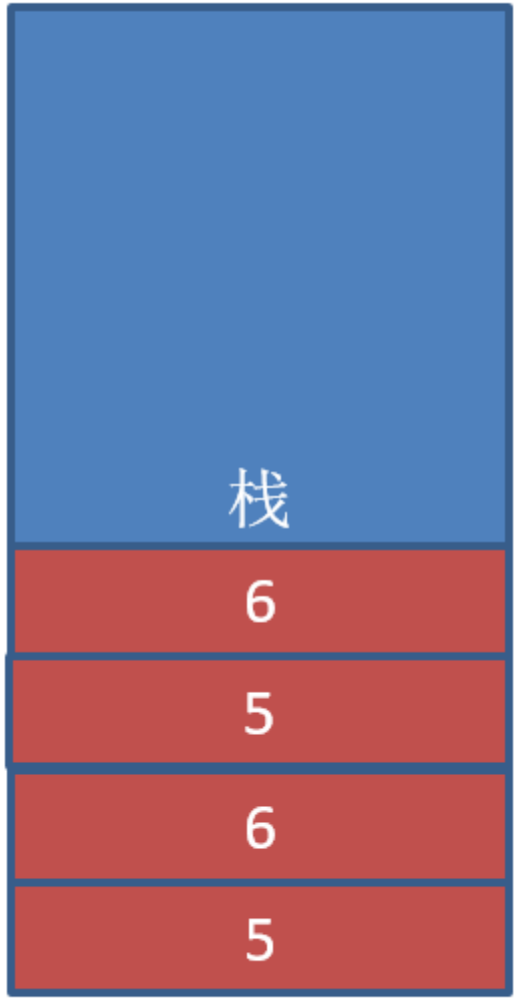
如果将复杂类型的数据赋值给一个变量，复杂类型的数据会在内存中创建一个原型，而变量中存储的是指向对象的一个地址，如果将变量赋值给另一个变量，相当于将地址复制一份给了新的变量，两个变量的地址相同，指向的是同一个原型，不论通过哪个地址更改了原型，都是在原型上发生的更改，两个变量下次访问时，都会发生变化。

```
var  p = new Person('zs',18,1000);
var p1 = p;
```



基本类型作为函数的参数

基本类型的数据作为函数的参数，符合基本类型的数据特点。




```
var x = 5;
var y = 6
f1(x, y);

function f1(a,b) {
    a = a + 1;
    b = b + 1;
    console.log("a = " + a);
    console.log("b = " + b);
}
```

内置对象

掌握内置对象的使用方法

复习

- JavaScript 包含：ECMAscript DOM BOM
- ECMAscript 包含：变量、数据、运算符、条件分支语句、循环语句、函数、数组、对象……
- JavaScript 的对象包含三种：自定义对象 内置对象 浏览器对象
- ECMAscript 的对象：自定义对象 内置对象
- 使用一个内置对象，只需要知道对象中有哪些成员，有什么功能，直接使用
- 需要参考一些说明手册 W3C/MDN

内置对象

掌握内置对象

- **MDN**
- Math 对象
- Array 对象
- String 对象

MDN

Mozilla 开发者网络（MDN）提供有关开放网络技术（Open Web）的信息，包括 HTML、CSS 和万维网及 HTML5 应用的 API。

- MDN：https://developer.mozilla.org/zh-CN/
- 比如：通过查询MDN学习Math对象的random()方法的使用

如何学习一个方法?

1. 方法的功能
2. 参数的意义和类型
3. 返回值意义和类型
4. demo 进行测试

内置对象

掌握内置对象

- MDN
- **Math 对象**
- Array 对象
- String 对象

Math对象

- Math 对象它具有数学常数和函数的属性和方法，我们可以直接进行使用
- 根据数学相关的运算来找 Math 中的成员（求绝对值，取整）

演示：

Math.PI	圆周率
Math.random()	生成随机数
Math.floor()/Math.ceil()	向下取整/向上取整
Math.round()	取整，四舍五入
Math.abs()	绝对值
Math.max()/Math.min()	求最大和最小值
Math.sin()/Math.cos()	正弦/余弦
Math.power()/Math.sqrt()	求指数次幂/求平方根

案例

求 10-20 之间的随机数

内置对象

掌握内置对象

- MDN
- Math 对象
- **Array 对象**
- String 对象

创建数组对象的两种方式

字面量方式

new Array() 构造函数方法

检测数组类型

- instanceof 检测某个实例是否是某个对象类型

toString()

- toString()　　把数组转换成字符串，逗号分隔每一项

数组常用方法

首尾数据操作：

push()	在数组末尾添加一个或多个元素，并返回数组操作后的长度
pop()	删除数组最后一项，返回删除项
shift()	删除数组第一项，返回删除项
unshift()	在数组开头添加一个或多个元素，并返回数组的新长度

数组常用方法

合并和拆分：

concat()

- 将两个数组合并成一个新的数组，原数组不受影响。参数位置可以是一个数组字面量、数组变量、零散的值。

slice(start,end)

- 从当前数组中截取一个新的数组，不影响原来的数组，返回一个新的数组，包含从 start 到 end （不包括该元素）的元素。
- 参数区分正负，正值表示下标位置，负值表示从后面往前数第几个位置，参数可以只传递一个，表示从开始位置截取到字符串结尾。

数组常用方法

删除、插入、替换：

splice(index,howmany,element1,element2,……)

用于插入、删除或替换数组的元素

index：删除元素的开始位置

howmany：删除元素的个数，可以是0

element1,element2：要替换的新的数据。

数组常用方法

位置方法：

indexOf()　　查找数据在数组中最先出现的下标

lastIndexOf()　　查找数据在数组中最后一次出现的下标

注意：如果没找到返回-1

数组常用方法

倒序：reverse() 将数组完全颠倒，第一项变成最后一项，最后一项变成第一项。

数组常用方法

排序：sort();　　默认根据字符编码顺序，从小到大排序

如果想要根据数值大小进行排序，必须添加sort的比较函数参数。

该函数要比较两个值，然后返回一个用于说明这两个值的相对顺序的数字。比较函数应该具有两个参数 a 和 b，根据a和b的关系作为判断条件，返回值根据条件分为三个分支，正数、负数、0：

返回值是负数-1：a排在b前面。

返回值是正数1：a排在b后面。

返回值是0：a和b的顺序保持不变。

人为能控制的是判断条件。

数组常用方法

转字符串方法：将数组的所有元素连接到一个字符串中。

join()　　通过参数作为连字符将数组中的每一项用连字符连成一个完整的字符串

清空数组

方式1 推荐

```
arr = [];
```

方式2

```
arr.length = 0;
```

方式3

```
arr.splice(0, arr.length);
```

内置对象

掌握内置对象

- MDN
- Math 对象
- Array 对象
- **String 对象**

基本包装类型

为了方便操作简单数据类型，JavaScript 还提供了特殊的简单类型对象：String

基本类型是没有方法的。

当调用 str.substring() 等方法的时候，先把 str 包装成 String 类型的临时对象，再调用substring 方法，最后销毁临时对象。

可以使用 new String() 构造函数方法创建字符串对象。

字符串的特点

字符串是不可变的。

由于字符串的不可变，在大量拼接字符串的时候会有效率问题。

字符串对象的常用方法

字符串所有的方法，都不会修改字符串本身（字符串是不可变的），操作完成会返回一个新的字符串。

字符串属性

长度属性：str.length

字符串长度指的是一个字符串中所有的字符总数。

字符串方法

charAt() 方法可返回指定位置的字符。

- char：charator，字符
- at：在哪儿
- 参数是 index 字符串的下标。也是从 0 开始。
- 表示返回指定的下标位置的字符。

字符串方法

indexOf()　　方法可返回某个指定的字符串值在字符串中首次出现的位置。

- 找到指定的子字符串在原字符串中第一次出现的位置的下标。如果子字符串在原字符串中没有，返回值是 -1

字符串方法

concat() 方法用于连接两个或多个字符串。

- 参数比较灵活，可以是字符串、或者字符串变量、多个字符串。
- 生成的是一个新的字符串，原字符串不发生变化。

字符串方法

split()　　方法用于把一个字符串分割成字符串数组。

- 参数部分是分割符，利用分割符将字符串分割成多个部分，多个部分作为数组的每一项组成数组。
- 如果分割符是空字符串，相当于将每个字符拆分成数组中的每一项。

字符串方法

toLowerCase()	把字符串转换为小写。
toUpperCase()	把字符串转换为大写。

- 将所有的英文字符转为大写或者小写。
- 生成的是新的字符串，原字符串不发生变化。

字符串方法

slice()　　方法可提取字符串的某个部分，并以新的字符串返回被提取的部分。

- 语法：slice(start，end)
- 从开始位置截取到结束位置（不包括结束位置）的字符串。
- 参数区分正负，正值表示下标位置，负值表示从后面往前数第几个位置，参数可以只传递一个，表示从开始位置截取到字符串结尾。

字符串方法

substr() 方法可在字符串中抽取从 start 下标开始的指定数目的字符

- 语法：substr(start,howmany)
- 从开始位置截取到指定长度的字符串。
- start 参数区分正负。正值表示下标位置，负值表示从后往前数第几个位置。
- howmany 参数必须为正数，也可以不写，不写表示从 start 截取到最后。

字符串方法

substring()　　方法用于提取字符串中介于两个指定下标之间的字符。

- 语法：substring(start,end)
- 参数只能为正数。
- 两个参数都是指代下标，两个数字大小不限制，执行方法之前会比较一下两个参数的大小，会用小当做开始位置，大的当做结束位置，从开始位置截取到结束位置但是不包含结束位置。
- 如果不写第二个参数，从开始截取到字符串结尾.

拉勾教育

—互联网人实战大学—

下载「拉勾教育App」
获取更多内容