JavaScript 基础

第三课

> 目标
>
> 1. 数组及其应用
> 2. 函数及其应用

数组

为什么要学习数组

> 之前学习的数据类型，只能存储一个值(比如：Number/String。我们想存储班级中所有学生的姓名，此时该如何存储?
>
> 数组是一种数据类型，可以存储很多项，有一定的顺序，很多项的集合，就叫数组。 Array

数组的概念

> 所谓数组，就是将多个元素（通常是同一类型），按照一定的顺序放到一个集合中，那么这个集合我们就称之为数组

数组的定义

> 数组是一个有序列表，可以在数组中存放任意的数据，并且数组的长度可以动态的调整

通过数组字面量创建数组

```
// 创建一个空数组
var arr1 = [];
// 创建一个包含3个数值的数组，多个数组项以逗号隔开
var arr2 = [1, 3, 4];
// 创建一个包含2个字符串的数组
var arr3 = ['a', 'c'];

// 可以通过数组的length属性获取数组的长度
console.log(arr3.length);
// 可以设置length属性改变数组中元素的个数
arr3.length = 0;
```

获取数组元素

```
// 格式：数组名[下标]    下标又称索引
// 功能：获取数组对应下标的那个值，如果下标不存在，则返回undefined。
var arr = ['red',, 'green', 'blue'];
arr[0]; // red
arr[2]; // blue
arr[3]; // 这个数组的最大下标为2,因此返回undefined
```

数组遍历

```
for(var i = 0; i < arr.length; i++) {
    // 数组遍历的固定结构
}
```

在数组中新增元素

数组的赋值

```
// 格式：数组名[下标/索引] = 值;
// 如果下标有对应的值，会把原来的值覆盖，如果下标不存在，会给数组新增一个元素。
var arr = ["red", "green", "blue"];
// 把red替换成了yellow
arr[0] = "yellow";
// 给数组新增加了一个pink的值
arr[3] = "pink";
```

案例

> 1. 求一组数中的所有数的和和平均值
> 2. 求一组数中的最大值和最小值，以及所在位置
> 3. 将字符串数组用|或其他符号分割
> 4. 要求将数组中的0项去掉，将不为0的值存入一个新的数组，生成新的数组
> 5. 翻转数组
> 6. 冒泡排序

函数

为什么要有函数

> 如果要在多个地方求1-100之间所有数的和，应该怎么做?

什么是函数

> 把一段相对独立的具有特定功能的代码块封装起来，形成一个独立实体，就是函数，起个名字（函数名），在后续开发中可以反复调用
>
> 函数的作用就是封装一段代码，将来可以重复使用

函数的定义

- 函数声明

```
function 函数名(){
  // 函数体
}
```

- 特点
  - 函数声明的时候，函数体并不会执行，只要当函数被调用的时候才会执行。
  - 函数一般都用来干一件事情，需用使用动词+名词，表示做一件事情 `tellStory` `sayHello`等

函数的调用

- 调用函数的语法

```
函数名();
```

- 特点
  - 函数只有在调用的时候才会执行，调用需要()进行调用。
  - 函数可以调用多次(重复使用)

```
// 声明函数
function sayHi() {
  console.log("吃了没? ");
}
// 调用函数
sayHi();

// 求1-100之间所有数的和
function getSum() {
  var sum = 0;
  for (var  i = 0; i < 100; i++) {
    sum += i;
  }
  console.log(sum);
}
// 调用
getSum();
```

函数的参数

- 为什么要有参数

```
function getSum() {
  var sum = 0;
  for (var i = 1; i <= 100; i++) {
    sum += i;
  }
  console.log();
}

// 虽然上面代码可以重复调用，但是只能计算1-100之间的值
// 如果想要计算n-m之间所有数的和，应该怎么办呢?
```

- 语法

```
// 函数内部是一个封闭的环境，可以通过参数的方式，把外部的值传递给函数内部
// 带参数的函数声明
function 函数名(形参1, 形参2, 形参...){
  // 函数体
}

// 带参数的函数调用
函数名(实参1, 实参2, 实参3);
```

- 形参和实参

1. 形式参数：

> 在声明一个函数的时候，为了函数的功能更加灵活，有些值是固定不了的，对于这些固定不了的值。我们可以给函数设置参数。这个参数没有具体的值，仅仅起到一个占位置的作用，我们通常称之为形式参数，也叫形参。

2. 实际参数：

> 如果函数在声明时，设置了形参，那么在函数调用的时候就需要传入对应的参数，我们把传入的参数叫做实际参数，也叫实参。

```
var x = 5, y = 6;
fn(x,y);
function fn(a, b) {
  console.log(a + b);
}
//x,y实参，有具体的值。函数执行的时候会把x,y复制一份给函数内部的a和b，函数内部的值是复制的新值，无法修改外部的x,y
```

> 1. 求1-n之间所有数的和
> 2. 求n-m之间所有数额和
> 3. 圆的面积
> 4. 求2个数中的最大值
> 5. 求3个数中的最大值
> 6. 判断一个数是否是素数

函数的返回值

> 当函数执行完的时候，并不是所有时候都要把结果打印。我们期望函数给我一些反馈（比如计算的结果返回进行后续的运算），这个时候可以让函数返回一些东西。也就是返回值。函数通过return返回一个返回值
>
> 每个函数都有返回值，如果没有写返回值，默认返回undefined

```
//声明一个带返回值的函数
function 函数名(形参1, 形参2, 形参...){
  //函数体
  return 返回值;
}

//可以通过变量来接收这个返回值
var 变量 = 函数名(实参1, 实参2, 实参3);
```

函数的调用结果就是返回值，因此我们可以直接对函数调用结果进行操作。

> 返回值：
>
> - 如果函数没有显示的使用 return语句 ，那么函数有默认的返回值：undefined
> - 如果函数使用 return语句，那么跟再return后面的值，就成了函数的返回值
> - 如果函数使用 return语句，但是return后面没有任何值，那么函数的返回值也是：undefined
>
> 函数使用return语句后，这个函数会在执行完 return 语句之后停止并立即退出，也就是说return后面的所有其他代码都不会再执行。
>
> 要么让函数始终都返回一个值，要么永远都不要返回值。

- 案例
    1. 求一组数中的最大值
    2. 求一组数中的最小值
    3. 求阶乘
    4. 求1!+2!+3!+....+n!

arguments的作用

> JavaScript中，arguments对象是比较特别的一个对象，实际上是当前函数的一个内置属性。也就是说所有函数都内置了一个arguments对象，arguments对象中存储了传递的所有的实参。arguments是一个伪数组，因此及可以进行遍历

- 案例
    1. 求任意个数的最大值
    2. 求任意个数的和

代码规范

```
1.命名规范
    函数 和变量的命名 必须要有意义，变量的名称 一般用名词    函数 一般用动词
    遵循驼峰命名法
2.变量规范
    var name = 'zs';
3.注释规范
    // 这里是注释
4.空格规范
5.换行规范
    var arr = [1, 2, 3, 4];
    if (a > b) {

    }
    for(var i = 0; i < 10; i++) {

    }
    function fn() {

    }
```

作业

1. 求斐波那契数列Fibonacci中的第n个数是多少？ 1 1 2 3 5 8 13 21...
2. 翻转数组，返回一个新数组
3. 对数组排序，从小到大
4. 输入一个年份，判断是否是闰年[闰年：能被4整数并且不能被100整数，或者能被400整数]
5. 输入某年某月某日，判断这一天是这一年的第几天?