JavaScript基础第04天笔记

1 - 数组

1.1 数组的概念

- 数组可以把一组相关的数据一起存放，并提供方便的访问(获取）方式。
- 数组是指**一组数据的集合**，其中的每个数据被称作**元素**，在数组中可以**存放任意类型的元素**。数组是一种将一组数据存储在单个变量名下的优雅方式。

1.2 创建数组

JS 中创建数组有两种方式：

- 利用 new 创建数组

  ```
  var 数组名 = new Array() ;
  var arr = new Array();   // 创建一个新的空数组
  ```

  注意 Array () ， A 要大写

- 利用数组字面量创建数组

  ```
  //1. 使用数组字面量方式创建空的数组
  var  数组名 = [];
  //2. 使用数组字面量方式创建带初始值的数组
  var  数组名 = ['小白','小黑','大黄','瑞奇'];
  ```

  - 数组的字面量是方括号 [ ]
  - 声明数组并赋值称为数组的初始化
  - 这种字面量方式也是我们以后最多使用的方式

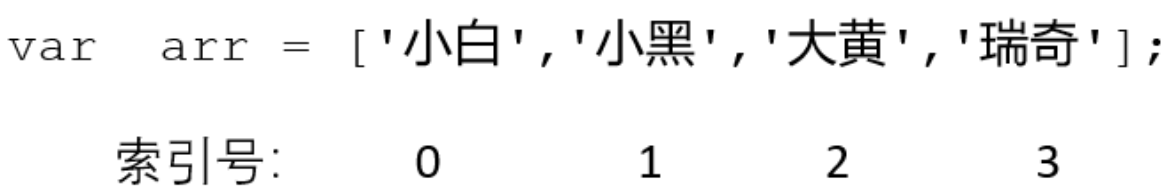
- 数组元素的类型

  数组中可以存放任意类型的数据，例如字符串，数字，布尔值等。

  ```
  var arrStus = ['小白',12,true,28.9];
  ```

1.3 获取数组中的元素

索引 (下标)： 用来访问数组元素的序号（数组下标从 0 开始）。



数组可以通过索引来访问、设置、修改对应的数组元素，可以通过“数组名[索引]”的形式来获取数组中的元素。

```
// 定义数组
var arrStus = [1,2,3];
// 获取数组中的第2个元素
alert(arrStus[1]);
```

注意：如果访问时数组没有和索引值对应的元素，则得到的值是undefined

1.4 遍历数组

- 数组遍历

  把数组中的每个元素从头到尾都访问一次（类似学生的点名），可以通过 for 循环索引遍历数组中的每一项

```
var arr = ['red','green', 'blue'];
for(var i = 0; i < arr.length; i++){
    console.log(arrStus[i]);
}
```

- 数组的长度

  数组的长度：默认情况下表示数组中元素的个数

  使用“数组名.length”可以访问数组元素的数量（数组长度）。

  ```
  var arrStus = [1,2,3];
  alert(arrStus.length);  // 3
  ```

  注意：

  - 此处数组的长度是数组元素的个数，不要和数组的索引号混淆。
- 当我们数组里面的元素个数发生了变化，这个 length 属性跟着一起变化
  - 数组的length属性可以被修改：
- 如果设置的length属性值大于数组的元素个数，则会在数组末尾出现空白元素；
  - 如果设置的length属性值小于数组的元素个数，则会把超过该值的数组元素删除

1.5 数组中新增元素

数组中可以通过以下方式在数组的末尾插入新元素：

```
数组[ 数组.length ] = 新数据;
```

2 - 函数

2.1 函数的概念

在 JS 里面，可能会定义非常多的相同代码或者功能相似的代码，这些代码可能需要大量重复使用。虽然 for循环语句也能实现一些简单的重复操作，但是比较具有局限性，此时我们就可以使用 JS 中的函数。

函数：就是**封装了一段可被重复调用执行的代码块**。通过此代码块可以**实现大量代码的重复使用**。

2.2 函数的使用

声明函数

```
// 声明函数
function 函数名() {
    //函数体代码
}
```

- function 是声明函数的关键字,必须小写
- 由于函数一般是为了实现某个功能才定义的， 所以通常我们将函数名命名为动词，比如 getSum

调用函数

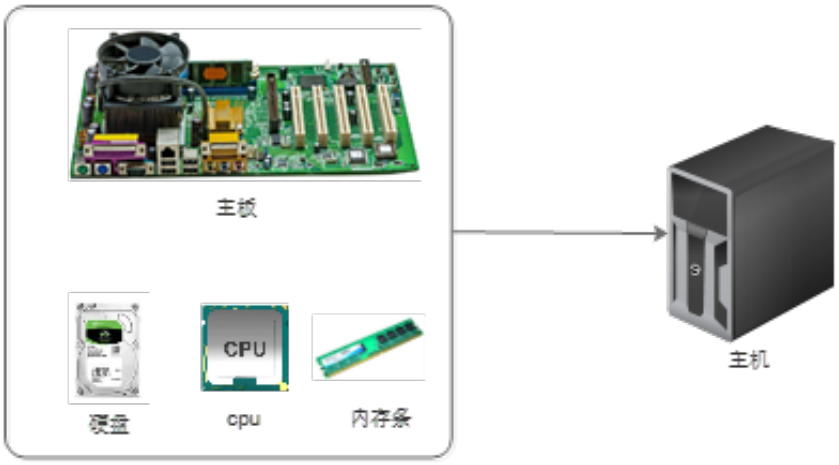
```
// 调用函数
函数名();  // 通过调用函数名来执行函数体代码
```

- 调用的时候千万不要忘记添加小括号
- 口诀：函数不调用，自己不执行

  注意：声明函数本身并不会执行代码，只有调用函数时才会执行函数体代码。

函数的封装

- 函数的封装是把一个或者多个功能通过函数的方式封装起来，对外只提供一个简单的函数接口
- 简单理解：封装类似于将电脑配件整合组装到机箱中 (类似快递打包)



例子：封装计算1-100累加和

```
/*
    计算1-100之间值的函数
*/
// 声明函数
function getSum(){
  var sumNum = 0;// 准备一个变量，保存数字和
  for (var i = 1; i <= 100; i++) {
    sumNum += i;// 把每个数值 都累加 到变量中
  }
  alert(sumNum);
}
// 调用函数
getSum();
```

2.3 函数的参数

函数参数语法

- 形参：函数定义时设置接收调用时传入
- 实参：函数调用时传入小括号内的真实数据

参数	说明
形参	形式上的参数 函数定义的时候 传递的参数 当前并不知道是什么
实参	实际上的参数 函数调用的时候传递的参数 实参是传递给形参的

参数的作用：在函数内部某些值不能固定，我们可以通过参数在调用函数时传递不同的值进去。

函数参数的运用：

```
// 带参数的函数声明
function 函数名(形参1, 形参2 , 形参3...) { // 可以定义任意多的参数，用逗号分隔
  // 函数体
}
// 带参数的函数调用
函数名(实参1, 实参2, 实参3...);
```

1. 调用的时候实参值是传递给形参的
2. 形参简单理解为：不用声明的变量
3. 实参和形参的多个参数之间用逗号（,）分隔

函数形参和实参数量不匹配时

参数个数	说明
实参个等于形参个数	输出正确结果
实参个数多于形参个数	只取到形参的个数
实参个数小于形参个数	多的形参定义为undefined，结果为NaN

> 注意：在JavaScript中，形参的默认值是undefined。

小结：

- 函数可以带参数也可以不带参数
- 声明函数的时候，函数名括号里面的是形参，形参的默认值为 undefined
- 调用函数的时候，函数名括号里面的是实参
- 多个参数中间用逗号分隔
- 形参的个数可以和实参个数不匹配，但是结果不可预计，我们尽量要匹配

2.4 函数的返回值

return 语句

> 返回值：函数调用整体代表的数据；函数执行完成后可以通过return语句将指定数据返回 。

```
// 声明函数
function 函数名 () {
    ...
    return   需要返回的值;
}
// 调用函数
函数名();     // 此时调用函数就可以得到函数体内return 后面的值
```

- 在使用 return 语句时，函数会停止执行，并返回指定的值
- 如果函数没有 return ，返回的值是 undefined

break ,continue ,return 的区别

- break： 结束当前的循环体（如 for、while）
- continue： 跳出本次循环，继续执行下次循环（如 for、while）
- return： 不仅可以退出循环，还能够返回 return 语句中的值，同时还可以结束当前的函数体内的代码

2.5 arguments的使用

当不确定有多少个参数传递的时候，可以用 arguments 来获取。JavaScript 中，arguments实际上它是当前函数的一个内置对象。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实参。arguments展示形式是一个伪数组，因此可以进行遍历。伪数组具有以下特点：

- 具有 length 属性
- 按索引方式储存数据
- 不具有数组的 push , pop 等方法

  注意：在函数内部使用该对象，用此对象获取函数调用时传的实参。

2.6 函数案例

> 函数内部可以调用另一个函数，在同一作用域代码中，函数名即代表封装的操作，使用函数名加括号即可以将封装的操作执行。

2.7 函数的两种声明方式

- 自定义函数方式(命名函数)

  利用函数关键字 function 自定义函数方式

  ```
  // 声明定义方式
  function fn() {...}
  // 调用
  fn();
  ```

  - 因为有名字，所以也被称为命名函数
  - 调用函数的代码既可以放到声明函数的前面，也可以放在声明函数的后面
- 函数表达式方式(匿名函数)

  利用函数表达式方式的写法如下：

  ```
  // 这是函数表达式写法，匿名函数后面跟分号结束
  var fn = function(){...};
  // 调用的方式，函数调用必须写到函数体下面
  fn();
  ```

  - 因为函数没有名字，所以也被称为匿名函数
  - 这个fn 里面存储的是一个函数
  - 函数表达式方式原理跟声明变量方式是一致的

- 函数调用的代码必须写到函数体后面