JavaScript第二节课

// 每条代码后面可以加;

// 不写也不会报错，也不影响使用，因为js有自动加;的机制

一、引用数据类型

1. 数组 (Array)是一组按顺序排列的数据的集合，数组中的每个值都称为元素。

   数组中的每个元素都有一个唯一的索引，从0开始递增。

   数组中可以包含任意类型的数据（包括数字、字符串、布尔值、对象以及其他数组等）。

   在 JavaScript 中定义数组需要使用方括号`[ ]`，数组中的每个元素使用逗号进行分隔

   ```
   <script>
     // 创建一个空数组
     let array = [];
     // 创建一个包含初始值的数组
     let array = [1, 2, 3];
     // 直接创建
     let arr = [1, 2, 3, 4]
   </script>
   ```

   通过索引取值

   ```
     let arr = [1,2,3,4,5,6]
     console.log(arr[0]);
     console.log(arr[1]);
     console.log(arr[2]);
     console.log(arr[3]);
     console.log(arr[4]);
     console.log(arr[5]);
   ```

   以下是一些常用的数组操作方法：

   - `array.length`：获取数组的长度。
   - `array[index]`：通过索引获取数组中的元素。
   - `array.push(element)`：向数组末尾添加一个元素。
   - `array.pop()`：从数组末尾移除并返回一个元素。
   - `array.unshift(element)`：向数组开头添加一个元素。
   - `array.shift()`：从数组开头移除并返回一个元素。
   - `array.splice(start, count, element1, element2, ...)`：从指定位置(`start`)开始删除指定数量(`count`)的元素，并可选地插入新的元素(`element`)。
   - `array.concat(array1, array2, ...)`：合并一个或多个数组，返回一个新数组。
   - `array.indexOf(element)`：返回指定元素在数组中首次出现的索引，如果不存在则返回-1。

- `array.includes(element)`：判断数组是否包含指定元素，返回布尔值。

2. 对象 （Object）类型是一组由键、值组成的无序集合，定义对象类型需要使用花括号`{ }`

   其中每个键都是字符串（不加引号），而值可以是任何数据类型，包括数字、字符串、布尔值、数组、函数等。

   对象是JavaScript中非常重要的数据结构之一，它提供了一种灵活的方式来组织和处理数据。

```
<script>
  let arr = {
    name:'不渝',
    age:'18'
  }
</script>
```

   通过`对象名.键名字`取值，通过.名字修改值

```
<script>
  let arr = {
    name:'不渝',
    age: '18'
  }
  console.log(arr.name);
  arr.name = "小柒"
  console.log(arr.name);
  console.log(arr.age);
</script>
```

   还可以使用对象的方法来操作对象的属性和行为。例如，使用`Object.keys(obj)`可以获取对象的所有键，使用`Object.values(obj)`可以获取对象的所有值，使用`Object.entries(obj)`可以获取对象的所有键值对。

二、循环

循环就是重复做一件事，在编写代码的过程中，我们经常会遇到一些需要反复执行的操作，例如遍历一些数据、重复输出某个字符串等，如果一行行的写那就太麻烦了，对于这种重复的操作，我们应该选择使用循环来完成。

2.1 for循环

循环适合在已知循环次数时使用

`语法说明`

- 形参1: 初始化计数器变量
- 形参2: 设置循环的次数
- 形参3：每次循环结束后更新（递增或递减）计数器的值

```
for(形参1;形参2;形参3){
  //循环代码块;
}
```

```
    for(let a = 0; a < 10; a++){
      console.log(a);
    }
```

应用：

```
//遍历数组值
let arr = [1,2,3,4,5,6]

// 使用for循环遍历数组
for(let a = 0; a < arr.length; a++){
        console.log(arr[a]);
}
```

```
// 假设我们有一个包含对象的数组，每个对象都有姓名和年龄属性
let peopleArray = [
    { name: '张三', age: 25 },
    { name: '李四', age: 30 },
    { name: '王五', age: 35 },
    // ... 可以继续添加更多对象
];

// 使用for循环遍历数组
for (let i = 0; i < peopleArray.length; i++) {
    // 通过索引i访问数组中的每个对象
    let person = peopleArray[i];

    // 访问并打印对象的属性
    console.log('姓名: ' + person.name);
    console.log('年龄: ' + person.age);
    console.log('----------'); // 为了清晰，可以打印一条分隔线
}
```

2.2 while循环

适合根据条件循环(未知循环次数但满足条件的情况)

```
    while(条件){
        执行代码
    }
```

```
  <script>
    let a = 10
    while(a == 10){
      console.log('因为a等于10 满足条件 代码一直执行');
    }
  </script>
```

```
//遍历数组值
let list = ["A","B","C","D"] ;
i = 0 ;
while (i< list.length){
    console.log(`当前数组下标是${i},当前下标的值是${list[i]}`);
    i++;
}
```

课后思考题：

```
// 假设我们有一个包含对象的数组，每个对象都有姓名和年龄属性
let peopleArray = [
    { name: '张三', age: 25 },
    { name: '李四', age: 30 },
    { name: '王五', age: 35 },
    // ... 可以继续添加更多对象
];

// 要求： 使用while循环遍历数组，在循环过程中访问数组中的每个对象，打印对象的属性。
```

> **结束、跳过循环（拓展）**
>
> ```
> /* break关键词使用 */
> for (let i = 0; i < 10; i++) {
>     if (i === 5) {
>         break; // 当i等于5时，结束循环
>     }
>     console.log(i);
> }
> // 运行结果-输出：0 1 2 3 4
> ```
>
> ```
> /* continue关键词使用 */
> for (let i = 0; i < 10; i++) {
>     if (i === 5) {
>         continue; // 当i等于5时，跳过当前循环的剩余部分
>     }
>     console.log(i);
> }
> // 运行结果-输出：0 1 2 3 4 6 7 8 9
> ```

三、反引法

在JavaScript中，你可以使用模板字面量（Template literals）和`${}`语法来插入变量和表达式。这提供了一种方便的方式来格式化输出字符串。

```
  <script>
    let a = 10
    console.log(`我今年${10}岁了`);
    //使用了反引号（``）来定义模板字面量，其中${}用于插入变量。
    //在${}内部，你可以放置任何有效的JavaScript表达式，包括变量、函数调用和运算符等。
    //模板字面量的好处是它提供了更直观、易读的方式来构建包含变量的字符串，而不需要使用字符串连接操作符（+）或复杂的字符串拼接方法。
  </script>
```

四、函数 function

函数是一组执行特定任务（具有特定功能）的，可以重复使用的代码块

5.1 定义函数

```
/*语法：*/
// 第一种方式：常规声明
function 函数名(){
      逻辑代码
}
// 第二种方式：箭头函数
let name = () =>{
    逻辑代码
}
```

5.2 调用函数

```
    let name = ()=>{

    }
    // 调用函数name()
    name()
```

5.3 形参和实参

```
    let name = (形参1,形参2)=>{

    }
    name(实参1,实参2)
```

```
    let name = (x,y)=>{
      console.log(x,y);
    }
    name(10,20)
```

5.4 return

返回函数内部的值

```
    let name = (x,y)=>{
      return x + y
    }
    a = name(10,20)
    console.log(a);
```

5.5函数形参默认值

y不传就默认为20 传了就以传过来的实参为准

```
    let name = (x,y=20)=>{
      return x + y
    }
    a = name(10)
    console.log(a);
```

5.6函数嵌套

```
  <script>
    let a = () => {
      console.log('我是函数a')
      b()
    }
    let b = () => {
      console.log('我是函数b')
    }
    a()
  </script>
```

```
/*课堂实例：*/
// 第一种方式：常规声明
function add(x,y){
console.log(`${x}+${y}=${x+y}`);
}
// 函数调用与python一样
add(1,1);
console.log(add(3,3));

// 第二种方式：箭头函数(可以给名字也可以不给名字)
let add2 = (x,y)=>{
return `${x}+${y}=${x+y}` ;
}
console.log(add2(2,2));

// 设置函数默认参数值
let add3 = (x,y=2)=>{
    return `${x}%${y}=${x%y}` ;
}
console.log(add3(10));

// 函数嵌套
let x = () => {
    console.log("这是x函数");
```

```
    y();
}
let y = () => {
    console.log("这是y函数");
    z();
}
let z = () => {
    console.log("这是z函数");
}

x();
```

六、定时器

JavaScript 定时器，有时也称为“计时器”，用来在经过指定的时间后执行某些任务，类似于我们生活中的闹钟。

JavaScript 中提供了两种方式来设置定时器，分别是 `setTimeout()` 和 `setInterval()`，它们之间的区别如下：

方法	说明
`setTimeout()`	在指定的时间后（单位为毫秒），执行某些代码，代码只会执行一次
`setInterval()`	按照指定的周期（单位为毫秒）来重复执行某些代码，定时器不会自动停止，除非调用 `clearInterval()` 函数来手动停止或者关闭浏览器窗口

6.1、`setTimeout()`

`setTimeout()` 函数用来在指定时间后执行某些代码，代码仅执行一次。

语法：setTimeout（函数，时间） 时间以毫秒为单位

```
  <script>
    setTimeout(()=>{
      console.log('3秒后此代码执行');
    },3000)
  </script>
```

6.2、`setInterval()`

`setInterval()` 函数可以定义一个能够重复执行的定时器，每次执行需要等待指定的时间间隔。

语法：setInterval（函数，时间）

```
    setInterval(()=>{
      console.log('每个3秒 当前代码执行一次');
    },3000)


    // 时间函数封装
let time = () => {
    // Date是js内置时间模块，可以获取当前时间对象
    let myDate = new Date();
```

```
    // 获取完整的年份(4位,1970-????)
    let n = myDate.getFullYear();
    let y = myDate.getMonth() + 1; // 获取当前月份(0-11,0代表1月)
    let r = myDate.getDate(); // 获取当前日(1-31)
    let s = myDate.getHours(); // 获取当前小时数(0-23)
    let f = myDate.getMinutes(); // 获取当前分钟数(0-59)
    let m = myDate.getSeconds(); // 获取当前秒数(0-59)
    let arr = n+"年"+y+"月" + r+"日"+ s+"时"+ f+"分"+ m +"秒";
    return arr;
 };
// 实现定时提醒当前时间的效果
setInterval(()=>{
    console.log(time());
},1000)
```

课后作业(交截图)：

1、用for 实现99乘法表，每行代码写注释。

2、将 `setTimeout()` 和 `setInterval()` 的案例自己练习一遍

3、拓展作业(可以写可以不写)：完成 拓展作业文档的题目。

4、预习 js三的内置 （注意：下节课之前，css的作业一定要完成）