第四章：JavaScript

第1节：js 介绍

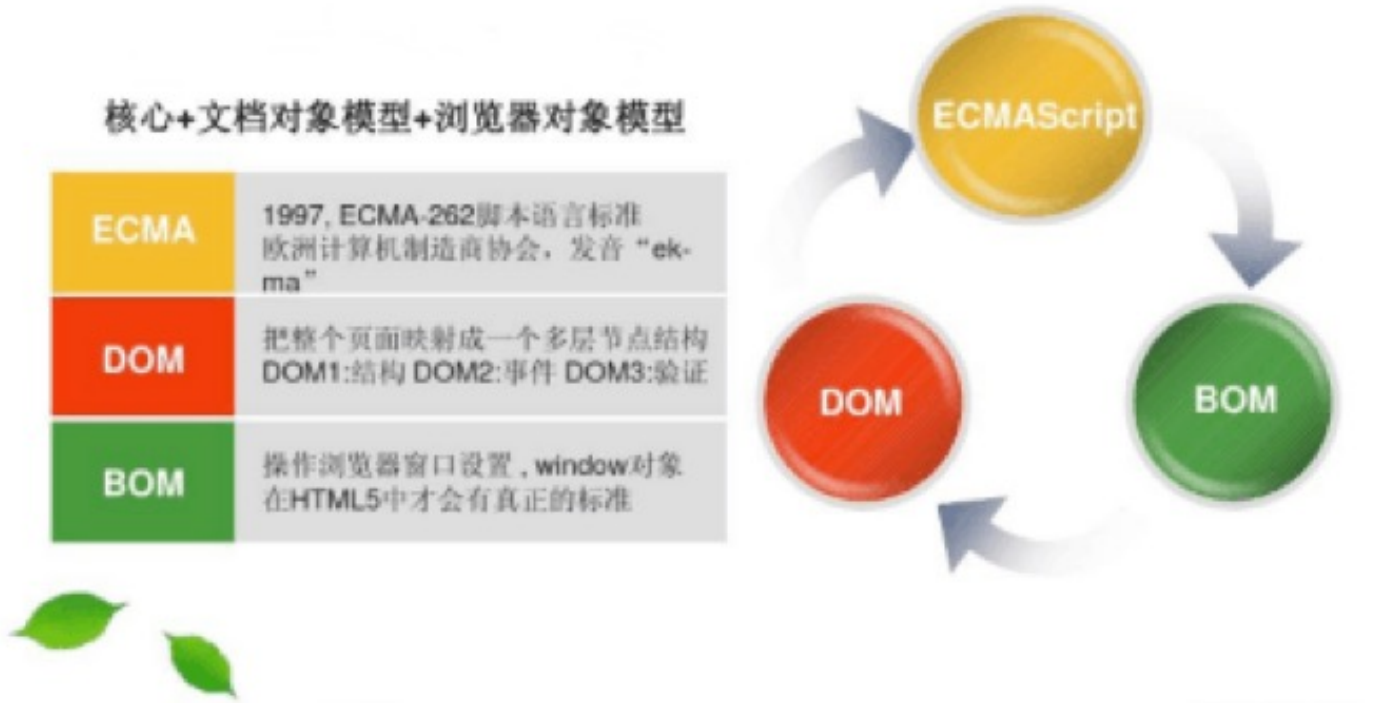
1.1 理解

1. JavaScript是一种即时编译型的编程语言，作为开发web页面的脚本语言，语法风格和Java相似，但是不是Java的产品。

1.2 作用

JavaScript可以实现页面标签的动态效果、可以校验html标签中的内容，直接和用户交互增强用户体验

1.3 组成



1. ECMAScript js的标准语法、事件等内容
2. DOM Document Object Model 文档对象模型　对标签的操作(增删改查)
3. BOM Broswer Object Model 浏览器对象模型 操作浏览器

1.4 使用

两种使用方式：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>使用</title>
    <!-- 方式2：使用script标签将外部的.js文件引入进来 -->
    <script src="js/xx.js" ></script>
</head>
<body>
    <!-- 方式1：使用script标签内部写js代码 script可以写在任何位置 -->
    <script>
        // alert("hello JS");
    </script>
</body>
</html>
```

第2节：ECMAScript

2.1 变量

格式：var 变量名 = 初始值;

```
<script>
    // 变量的定义  Java  数据类型 变量名 = 初始值
    /*
        格式：var 变量名 = 初始值 ;
    */
    var a = 10;
    var b = "der祥";
    var flag = true;
    var f = 3.14;
    var boy = {
        "name":"小鲁鲁",
        "age":18
    }
    console.log(a); // 浏览器的控制台输出
    console.log(b);
    console.log(flag);
    console.log(f);
    console.log(boy);

</script>
```

2.2 数据类型

常见类型：

1. number 所有的数值都属于该类型
2. string 使用单引号或者双引号 都属于该类型
3. boolean true或者false属于该类型
4. undefined 未定义类型 只声明不给初始值的变量
5. object 除以上之外都属于object

```
<script>

    var a = 1;
    var f = 3.14;
    var v = 'a';
    var str = "abcd";
    var b = false;
    var x;
    var n = null;
    var o = new Object();
    // typeOf() 获取变量的类型
    console.log(typeof(a)); // number类型
    console.log(typeof(f)); // number类型
    console.log(typeof(v)); // string类型
    console.log(typeof(str)); // string类型
    console.log(typeof(b)); // boolean类型

    console.log(typeof(x)); // undefined类型
    console.log(typeof(n)); // object
    console.log(typeof(o)); // object


</script>
```

2.3 运算符

1. 算术运算符：+ - * / % ++ --
2. 逻辑运算符：&& || !
3. 赋值运算符：= += -= *= /= %=
4. 比较运算符：== >= <= != > < ===(全等)
5. 位运算符：<< >> <<< >>> ^
6. 三目运算符：条件表达式?值1:值2

2.4 流程控制

1. 分支结构
   if(){}else{}   if(){}else if(){}else if(){}...
    switch(){}
2. 循环结构
   while(){} do{}while(); for()

2.5 函数

相当于Java中的方法

格式：

```
function 函数名(参数列表){
    函数体;
}
```

分类：

1. 无参无返回值
2. 有参无返回值
3. 无参有返回值
4. 有参有返回值
5. 匿名函数

```
// 定义函数
/*
格式:
    function 函数名(参数列表){
        函数体;
    }
*/
// 1. 无参无返回值
function func1(){
    console.log("无参无返回值函数执行");
}
// 调用
func1();

// 2. 有参无返回值
function func2(name, hobby){
    console.log("有参无返回值:"+name+","+hobby);
}
// 调用
func2("der祥", "小海海");

// 3. 无参 有返回值
function func3(){
    return "无参 有返回值";
}
// 调用
var result = func3();
console.log(result);
```

```
// 4. 有参 有返回值
function func4(name, age){
    return name+","+age+",有参 有返回值";
}
// 调用
var rel = func4("小杰杰", 18);
console.log(rel)

// 5. 匿名函数
var func5 = function (name){
    console.log("匿名函数: "+name)
}
// 调用
func5("嘿嘿");
```

2.6 事件

js对标签发生的某个动作 做某件事

```
<!-- 1. 点击事件 -->
<div id="d1" onclick="changeColor()" style="width: 100px; height: 100px; border: 1px solid g
</div>
<!-- 2. 双击事件 -->
<div ondblclick="changeColor1()" style="width: 100px; height: 100px; border: 1px solid green
</div>
<!-- 3. 内容改变事件 -->
<input onchange="func1()" type="text" />

<!-- 4. 鼠标悬停 和 5. 鼠标移出事件 -->
<div id="d2" onmouseover="func2()" onmouseout="func3()" style="width: 100px; height: 100px; bo
</div>

<!-- 6. 获取焦点 和 7. 失去焦点事件 -->
<input id="inp1" type="text" onfocus="func4()" onblur="func5()"/>
```

2.7 弹框

```
1. 弹警告框
    alert("信息");
2. 弹提示框
    confirm("提示信息");  返回boolean值
3. 弹输入框
    prompt("提示输入信息"); 返回输入框的值
```

```
<button id="btn1">点击按钮弹框(警告框)</button>
<button id="btn2">点击按钮弹框(提示框)</button>
<button id="btn3">点击按钮弹框(输入框)</button>
<script>
    // 1. 获取标签元素
   var btn1 = document.getElementById("btn1");
   var btn2 = document.getElementById("btn2");
   var btn3 = document.getElementById("btn3");
   // 2. 绑定点击事件
   btn1.onclick = function(){
       /* 1. 警告框 */
       // alert("点我干啥？");


   }

   btn2.onclick = function(){
        // 2. 提示框
       if(confirm("确定要删除吗？")){
           console.log("确定删除");
       }else{
           console.log("取消删除");
       }
   }

   btn3.onclick = function(){
     // 3. 弹输入框
     var av = prompt("请输入你喜欢的女神");
     console.log(av);
   }
```

2.8 常见对象

2.8.1 数组

数组：可以存储一组数据

js 数组的声明

```
var arr = new Array(); // 第一种
var arr1 = new Array(5); // 第二种
var arr2 = [1,3,5,7,9]; // 第三种
```

特点：

1. 数组长度可变
2. 数组中的数据可以存储不同的类型
3. js的数组相当于java中的集合 可以是单列集合 也可以是双列集合

数组的操作

1. 增 数组名[索引] = 值 或者 数组名.push(值)
2. 删 delete 数组名[索引]
3. 改 数组名[索引] = 新值
4. 查
   ① 普通for循环 要求索引为有序
   ② forin循环

注意：

数组名.length 获取数组中有序索引的最大值+1

```
// 1. 数组的定义
// java  数据类型[] 数组名 = new 数据类型[个数]    数据类型[] 数组名 = {值}
// js 数组的声明
var arr = new Array(); // 第一种
var arr1 = new Array(5); // 第二种
var arr2 = [1,3,5,7,9]; // 第三种

arr[1] = 10;
console.log(arr);

arr1[0] = 9;
arr1[1] = 3.14;
arr1[2] = true;
arr1[3] = null;
arr1[4] = "abc";
arr1[5] = "der祥";
arr1["abc"] = "甲一";
arr1[100] = 99;
// 注意: length 只获取有序索引的最大值+1
console.log(arr1.length);
console.log("----------------------")

// 2. 数组的操作(增删改查)
// 2.1 增  ① 数组名[索引] = 值  ② push函数
arr1.push("xxx");

// 2.2 删: delete 数组名[索引]
delete arr1[5];
```

```
// 2.3 改：数组名[索引] = 新值
arr1["abc"] = "一甲";

console.log(arr1);


// 2.4 查：单查 数组名[索引]
console.log(arr1[2]);
console.log(arr1["abc"]);

// 多查
// 方式1：普通for遍历数组(索引必须有序)
console.log("------------------------------")
for(var i=0; i<arr1.length; i++){
    console.log(arr1[i]);
}
console.log("------------------------------")
// 方式2：forin遍历数组（推荐使用）
for(var index in arr1){
    console.log(arr1[index]);
}
```

2.8.2 Math对象

```
console.log(Math.random()); // 随机数 [0,1)
console.log(Math.max(12,15)); // 取最大值
console.log(Math.min(12,15)); // 取最小值
console.log(Math.ceil(3.9)); // 向上取整
console.log(Math.floor(3.9)); // 向下取整
console.log(Math.round(5.5));  // 四舍五入
console.log(Math.PI); // 获取圆周率
console.log(Math.abs(-10)); // 取绝对值
```

2.8.3 String对象

```
1. 定义字符串
    var 变量名 = "";
    var 变量名 = new String("");
2. 常见函数
```

```
// 1. 定义字符串
var str1 = "abc";
var str2 = new String("abc");

// 2. 常见函数
console.log(str1.length);
console.log(str1.startsWith("b")); // 判断字符串以xx开头
console.log(str1.endsWith("c")); // 判断字符串以xx结尾

var str3 = "abcd#efghij";
var arr = str3.split("#"); // 拆分字符串
console.log(arr[0]+":::"+arr[1]);

// 1. 截取字符串 substring(起始索引,结束索引)
console.log(str3.substring(3));
console.log(str3.substring(3,6));

// 2. 截取字符串 substr(起始索引,截取的个数)
console.log(str3.substr(3,4));
```

2.8.4 日期对象

```
var date =  new Date();
```

```
// Thu Aug 26 2021 10:25:53 GMT+0800 (中国标准时间)
var date = new Date();
console.log(date);

var year = date.getFullYear(); // 获取年份
var month = date.getMonth()+1; // 获取月份
var day = date.getDay();    // 获取当周的第几天
var day1 = date.getDate();  // 获取当月的第几天
var hours = date.getHours(); // 获取小时
var minutes = date.getMinutes(); // 获取分钟
var seconds = date.getSeconds(); // 获取秒数

console.log(year);
console.log(month);
console.log(day);
console.log(day1);
console.log(hours);
console.log(minutes);
console.log(seconds);


var xx = year+"年"+month+"月"+day1+"日 "+hours+"时"+minutes+"分"+seconds+"秒";

document.write(xx);
```

2.9 定时器

1. 一次性定时器：到达时间执行
   ① 创建定时器 参数1：到时做什么 参数2：计时单位ms
   `setTimeout(参数1,参数2);`
   ② 清除定时器
   `clearTimeout(定时器对象);`

```
// 1. 获取按钮
var btn = document.getElementById("btn");
var btn1 = document.getElementById("btn1");

var xx;
// 2. 绑定点击事件
btn.onclick = function(){
    // 5秒后弹框
    // 1. 创建一次性定时器 参数1：到时做什么 参数2：计时单位ms
    xx = setTimeout("tan()", 5000);
}

btn1.onclick = function(){
    // 取消一次性定时器
    clearTimeout(xx);
}
function tan(){
    alert("点我干啥！");
}
```

2. 循环定时器：每隔多长时间执行一次
   ① 创建循环定时器
   `setInterval(做什么事,间隔时间);`
   ② 清除循环定时器
   `clearInterval(定时器对象)`

```
<button id="btn">开始</button>
<button id="btn1">暂停</button>
<button id="btn2">开始</button>
<div id="clock">

</div>

<script>
    // 获取当前时间
    function getTime1(){
       // 1. 获取当前日期
       var date = new Date();
       var year = date.getFullYear(); // 获取年份
       var month = date.getMonth()+1; // 获取月份
       var day = date.getDate();  // 获取当月的第几天
       var hours = date.getHours(); // 获取小时
       var minutes = date.getMinutes(); // 获取分钟
       var seconds = date.getSeconds(); // 获取秒数
       var time = year+"-"+month+"-"+day+" "+hours+":"+minutes+":"+seconds;
       // 1. 获取div
      var div =  document.getElementById("clock");
      // 2. 在div中存放内容
      div.innerHTML = time;
    }
    // 调用
    getTime1();

var btn = document.getElementById("btn");
var btn1 = document.getElementById("btn1");

var xx ; // 接收定时器对象
btn.onclick = function(){
    // 1. 创建循环定时器
    xx = setInterval("getTime1()", 1000);

    //2. 改变btn中的内容
    // this.innerHTML = "暂停";
}

btn1.onclick = function(){
    // 暂停定时器
    clearInterval(xx);
}
```

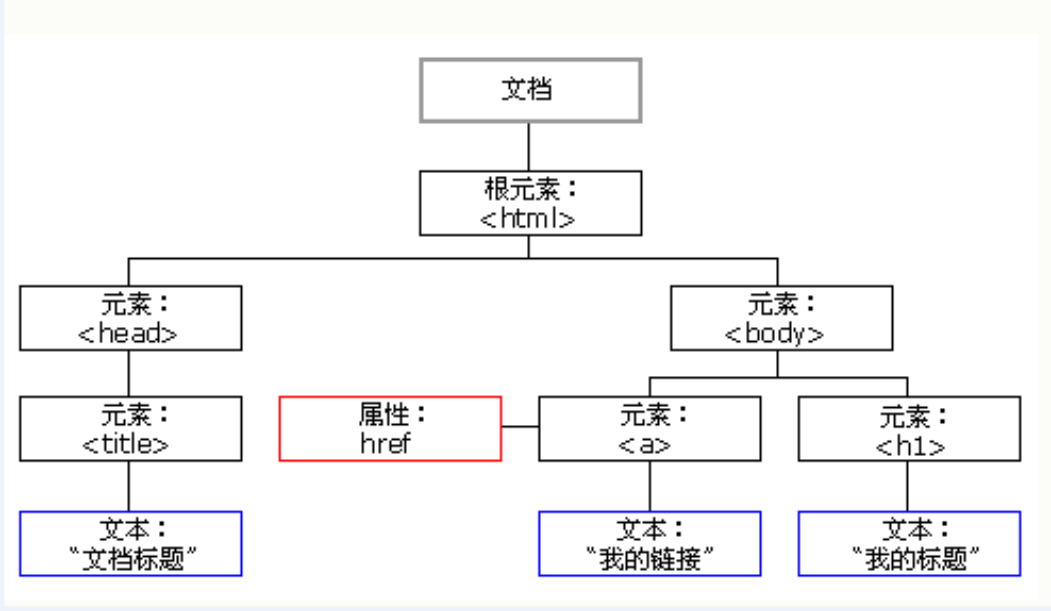
第3节：DOM

理解

Document Object Model 文档对象模型。js操作标签的时候 将所有的标签当成文档对象来操作。

作用:

1. 对页面标签的获取、创建、删除、修改
2. 对标签元素的属性、样式、内容的操作



3.1 获取标签元素

1. 查找元素的方法

① 根据标签的id值 获取标签元素对象 返回标签元素

```
document.getElementById("id值");
```

② 根据标签名 获取标签元素 返回的HTMLCollection

```
document.getElementsByTagName("标签名");
```

③ 根据标签的class值 获取标签元素对象 返回的HTMLCollection

```
document.getElementsByClassName("class值");
```

④ 根据标签的name值 获取标签元素对象 返回NodeList集合

```
document.getElementsByName("name值");
```

```
// 1. 根据标签的id值 获取标签元素对象 返回整个标签元素
var hh = document.getElementById("hh");
console.log(hh);

console.log("--------------");
//2. 根据标签名 获取标签元素 返回 HTMLCollection 集合
var sp = document.getElementsByTagName("span");
console.log(sp);

console.log("--------------");
//3. 根据标签的class值 获取标签元素 返回 HTMLCollection 集合
var sp1 = document.getElementsByClassName("sp1");
console.log(sp1);

console.log("--------------");
// 4. 根据标签的name值 获取标签元素对象 返回NodeList集合
var ho = document.getElementsByName("hobby");
console.log(ho);
```

3.2 创建元素

创建标签元素对象

```
document.createElement("标签名");
```

指定存放的位置

① `a.appendChild(b) // 将b标签元素 追加到a标签内部`

② `a.insertBefore(b,c)// 将b标签元素 插入到c标签元素的前面`

```
<body>
    <button id="btn">点击显示图片</button>

    <script>
        // 1. 获取标签元素
        var btn = document.getElementById("btn");

        // 2. 绑定点击事件
        btn.onclick = function(){
            // 3. 创建标签
            var img = document.createElement("img");

            img.src = "img/slj.jpg";

            // 4. 指定存放的位置
            // document.body.appendChild(img);
            document.body.insertBefore(img, btn); // 将img插入到btn的前面
        }

    </script>
</body>
```

3.3 删除元素

```
1. 标签元素对象.remove();  自己删除自己
2. a.removeChild(b); 根据父标签对象a删除子标签对象b
```

```
// 删除元素
var btn1 = document.getElementById("btn1");
btn1.onclick = function(){
    // 1. 删除标签元素
   var h2 = document.getElementsByTagName("h2")[0];

   // 2. 自己删除自己
   // h2.remove();
   // 3. 根据父标签删除子标签
   document.body.removeChild(h2);
}
```

3.4 修改元素

```
a.replaceChild(b,c) 将c标签替换成b标签
```

```
// 修改元素
function updateH2(){
    // 1. 获取要修改的元素标签
    var h2= document.getElementsByTagName("h2")[0];

    // 2. 创建修改后的标签
    var h3 = document.createElement("h3");
    h3.innerHTML = "大翔";
    // 3. 开始修改
    document.body.replaceChild(h3, h2); // 将h2替换成 h3

}
```

3.5 属性操作

```
1. 标签元素对象.属性名 = 属性值;  修改属性值
2. 标签元素.setAttribute("属性名","属性值"); 设置属性值
3. 标签元素.removeAttribute("属性名"); 删除属性值
4. 标签元素.getAttribute("属性名");  获取属性值
```

```
<body>
    <button onclick="changeImg()">点击修改图片</button><br >
    <img src="img/slj.jpg" />
    <script>
        // 修改属性
        function changeImg(){
            // 1. 获取元素标签
            var img = document.getElementsByTagName("img")[0];

            // 2. 修改属性 src
            // img.src = "img/001.jpg";
            img.setAttribute("src","img/001.jpg");
            img.setAttribute("title","凤姐");

            // 3. 获取属性值
            console.log(img.src);


            // 4. 删除属性
            img.removeAttribute("title");

            var v = img.getAttribute("src");
            console.log(v);
        }
    </script>
</body>
```

3.6 样式操作

1. 元素标签.style.样式名 = 样式值; 设置样式

```
<body>
    <button onclick="changeStyle()">点击修改div的样式</button>
    <div id="d1" style="width: 100px; height: 100px; border: 1px solid blue;">

    </div>
    <script>
        // 操作样式
        function changeStyle(){
            // 1. 获取需要操作的元素
            var d1 = document.getElementById("d1");

            // 2. 操作样式
            // 2.1 获取样式值
            // console.log(d1.style.width);

            // 2.2 修改样式
            d1.style.width = "200px";
        }
    </script>
</body>
```

2. 通过设置属性 class id style方式设置样式

```
    <style>
        .dd{
            width: 200px;
            height: 200px;
            background-color: red;
        }

        #d1{
            width: 100px; height: 100px; border: 1px solid blue;
        }
    </style>
</head>
<body>
    <button onclick="changeStyle()">点击修改div的样式</button>
    <div id="d1">

    </div>
    <script>
        // 操作样式
        function changeStyle(){
            // 1. 获取需要操作的元素
            var d1 = document.getElementById("d1");

            // 2. 操作样式
            // 2.1 获取样式值
            // console.log(d1.style.width);

            // 2.2 修改样式
            d1.style.width = "200px";


              d1.removeAttribute("id");

              d1.setAttribute("class","dd");

            }
    </script>
</body>
</html>
```

3.7 内容操作

1. 操作双边标签中的内容
   ① 标签元素.innerText;
   ② 标签元素.innerHTML
2. 操作input框内容
   标签元素.value

```
<body>
    <h2><i>你喜欢凤姐吗？</i></h2>
    <input type="text" name="username" value="哈哈哈" >
    <script>
        // 1. 获取元素对象
        var h =  document.getElementsByTagName("h2")[0];
        var inp = document.getElementsByName("username")[0];

        // 2. 获取h2中的文本内容
        console.log(h.innerText) // 获取标签的文本内容
        // h.innerText = "喜欢";  // 修改标签的文本内容

        // 3. 获取h2中的内容
        console.log(h.innerHTML); // 获取标签的内容

        h.innerHTML = "<i><a>喜欢</a></i>"; // 修改标签中的内容


        // 4. 获取input中的内容
        console.log(inp.value); // 获取input框内容
        inp.value = "嘿嘿嘿";  // 设置input框内容

    </script>
</body>
```

第4节：BOM

1. 理解

    BOM Browser Object Model浏览器对象模型，在浏览器初始化页面时 在内存中创建一个全局对象，用来描述窗体的属性和状态的。这个全局对象称为浏览器对象模型

2. BOM有一个核心对象window ,包含5个核心模块对象
    ① Document 文档对象
    ② History 页面的浏览器记录
    ③ Location 页面的地址
    ④ Screen 显示屏幕信息
    ⑤ Navigator 浏览器的相关信息
3. 常见函数
    ① 弹框
        【window.】alert()
        【window.】confirm()
        【window.】prompt()
    ② 定时器
        【window.】setTimeout()
        【window.】setInterval();

```html
<body>
    <button onclick="shua()">刷新</button>
    <button onclick="tui()">后退</button>
    <button onclick="jin()">前进</button>
    <button onclick="tiao()">点击跳转</button>
    <script>
        function tiao(){
            // js 页面跳转
            window.location.href = "11-dom之内容操作.html";
        }

        function jin(){
            // 前进
            window.history.forward();
        }

        function tui(){
            // 后退
            window.history.back();
        }

        function shua(){
            // 刷新
            window.location.reload();
        }
    </script>
</body>
```