本章内容

- js介绍及对象
- json

01_JavaScript的概述

- A.概念
  - 一门客户端脚本语言.
  - 脚本语言：不需要编译，直接就可以被浏览器解析执行.
- B.作用
  - 可以来增强用户和html页面的交互过程，可以来控制html元素，让页面有一些动态的效果，增强用户的体验。
- C.JavaScript发展史
  - a.1992年，Nombase公司，开发出第一门客户端脚本语言，专门用于表单的校验。命名为： C-- ，后来更名为：ScriptEase
  - b.1995年，Netscape(网景)公司，开发了一门客户端脚本语言：LiveScript。后来，请来SUN公司的专家，修改LiveScript，命名为JavaScript
  - c.1996年，微软抄袭JavaScript开发出JScript语言
  - d.1997年，ECMA(欧洲计算机制造商协会)，制定出客户端脚本语言的标准：ECMAScript，就是统一了所有客户端脚本语言的编码方式。
- D.总结
  - JavaScript = ECMAScript + JavaScript自己特有的东西(BOM+DOM)

02_ECMAScript之与html结合

- A.与html结合方式
  - a. 内部JS：
    - 定义<script>，标签体内容就是js代码
  - b. 外部JS：
    - 定义<script>，通过src属性引入外部的js文件
  - c.注意：
    a. <script>可以定义在html页面的任何地方。但是定义的位置会影响执行顺序。
    b. <script>可以定义多个。

03_ECMAScript之注释

- A.注释

- a.单行注释：

```
//注释内容
```

- b.多行注释：

```
/*注释内容*/
```

04_ECMAScript之数据类型：

- A.数据类型
  - a.原始数据类型(基本数据类型)：
    - number：数字。 整数/小数/NaN(not a number 一个不是数字的数字类型)
    - string：字符串。 字符串 "abc" "a" 'abc'
    - boolean: true和false
    - null：一个对象为空的占位符
    - undefined：未定义。如果一个变量没有给初始化值，则会被默认赋值为undefined
- B.引用数据类型：对象

05_ECMAScript之变量

- A.变量：一小块存储数据的内存空间
- Java语言是强类型语言，而JavaScript是弱类型语言。
  - 强类型：在开辟变量存储空间时，定义了空间将来存储的数据的数据类型。只能存储固定类型的数据
  - 弱类型：在开辟变量存储空间时，不定义空间将来的存储数据类型，可以存放任意类型的数据。
- 语法：
  - var 变量名 = 初始化值;

06_ECMAScript之typeof运方法

- 获取变量的类型。
  - 注：null运算后得到的是object

07_ECMAScript之运算符

- A.一元运算符：只有一个运算数的运算符
  - ++，--， +(正号)
- B. 算术运算符

```
+ - * / %
```

- C. 赋值运算符

```
= += -+
```

- D. 比较运算符

```
> < >= <= == ===(全等于)
```

  - 比较方式
  - 类型相同：直接比较
    - 字符串：按照字典顺序比较。按位逐一比较，直到得出大小为止。
  - 类型不同：先进行类型转换，再比较
    - ===：全等于,类型和值都比较。在比较之前，先判断类型，如果类型不一样，则直接返回false
- E. 逻辑运算符
  - && || !
- F. 三元运算符
  - 语法：
    - 表达式? 值1:值2;
    - 判断表达式的值，如果是true则取值1，如果是false则取值2；
- G.流程控制语句：
  - if...else...
  - switch:
    - 在java中，switch语句可以接受的数据类型： byte int shor char,枚举(1.5),String(1.7)
    - 在JS中,switch语句可以接受任意的原始数据类型
  - while
  - do...while
  - for
- H.注意事项
  - a. number：0或NaN为假，其他为真
  - b. string：除了空字符串("")，其他都是true
  - c. null&undefined:都是false
  - d. 对象：所有对象都为true

08_ECMAScript基本对象及Funciton对象

- A.概念
  - Function、Array、Boolean、Date、Number、String、RegExp、Global(全局函数对象)
- A. Function创建：
  - A. var fun = new Function(形式参数列表,方法体); //忘掉吧

```
<script>
    var fn1 = new Function("msg","console.log(msg);");
</script>
<button onclick="fn1('a')"></button>
```

- B.Function属性： length:代表形参的个数
- C.特点：
  - 方法定义是，形参的类型不用写,返回值类型也不写。
  - 方法是一个对象，如果定义名称相同的方法，会覆盖
  - 在JS中，方法的调用只与方法的名称有关，和参数列表无关
  - 在方法声明中有一个隐藏的内置对象（数组），arguments,封装所有的实际参数
- D.调用：
  - 方法名称(实际参数列表)

09_ECMAScript基本对象之Array对象

- A.创建：
  - var arr = new Array(元素列表);
  - var arr = new Array(默认长度);
  - var arr = [元素列表];
- B.方法
  - join(参数):将数组中的元素按照指定的分隔符拼接为字符串
  - push() 向数组的末尾添加一个或更多元素，并返回新的长度。
- C.属性
  - length:数组的长度
- D.特点：
  - JS中，数组元素的类型可变的。
  - JS中，数组长度可变的。

10_ECMAScript基本对象之Date对象

- A. 创建：
  - var date = new Date();
- B. 方法：
  - toLocaleString()：返回当前date对象对应的时间本地字符串格式
  - getTime():获取毫秒值。返回当前如期对象描述的时间到1970年1月1日零点的毫秒值差

11_ECMAScript基本对象之RegExp对象

- A. 创建
  - var reg = /^正则表达式$/;
- B. 方法
  - test(参数):验证指定的字符串是否符合正则定义的规范

12_ECMAScript基本对象之全局对象

- A. 特点
  - 全局对象，这个Global中封装的方法不需要对象就可以直接调用。
- B. 方法：
  - parseInt():将字符串转为数字
    - 逐一判断每一个字符是否是数字，直到不是数字为止，将前边数字部分转为number
  - isNaN():判断一个值是否是NaN
    - NaN六亲不认，连自己都不认。NaN参与的==比较全部为false
  - eval():计算JavaScript字符串，并把它作为脚本代码来执行

```
var jsonStr1 = '{"username":"root","password":"root"}';
//将json字符串转换成js对象
var obj = eval("("+jsonStr1+")");
//将js对象转换成json字符串
var jsonStr2 = JSON.stringify(obj);
console.log(jsonStr2)
```

12_Json介绍

- A.什么是Json?
  - JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于ECMAScript的一个子集。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯。这些特性使JSON成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成(网络传输速率)。
- B.Json语法
  - 一个数据:

```
{"键1":值1,"键2":值2}
```

  - 一组数据:

```
[{"键1":值1,"键2":值2},{"键1":值1,"键2":值2}]
```

  - a.数据在键值对里面
  - b.数据之间由逗号分隔
  - c.大括号保存对象
  - d.中括号保存数组
  - e.Json值
    - 数字（整数或浮点数）
    - 字符串（在双引号中）
    - 逻辑值（true 或 false）
    - 数组（在中括号中）
    - 对象（在大括号中）

        - null
- C.Json数据
    - Java类

```
class Province {
    int id;
    Stirng name;
}
```

    - Json数据
        - 单条数据，表示Java中的单一对象.
        - 一个省份

```
{"id":1,"name":"湖北省"}
```

        - 多条数据，表示Java中的集合或数组.
        - 多个省份

```
[{"id":1,"name":"湖北省"},{"id":2,"name":"湖南省"},{"id":3,"name":"四川省"}]
```

13_Gson的使用

- A.什么是Gson?
    - Gson是一个工具类:将对象,数组,List,Map集合转换成json字符串
- B.使用Gson
    - a.导入jar包
        - gson.jar
    - b.gson使用
        - 将对象转换成json字符串
        - 将List集合转换成json字符串
        - 将数组转换成json字符串
        - 将map集合转换成json字符串

14_Ajax概述

- A.概述
    - AJAX即“Asynchronous Javascript And XML”（异步JavaScript和XML），是指一种创建交互式网页应用的网页开发技术。
    - 通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。 传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页页面。
- B.作用
    - a.可以局部刷新页面
    - b.可以发起异步请求

- C.和同步请求的区别
  - 同步请求:当页面内容发生改变时,必须全部刷新,且刷新的时候不能发出其他请求
  - 异步请求:可以局部的改变网页上的内容,当正在发生改变时,其他的模块的内容也可以发出请求.

15_XMLHttpRequest对象详细介绍

- A.XMLHttpRequest概述
  - ajax异步请求对象
- B.属性
  - onreadystatechange
    - 用于指定XMLHttpRequest对象状态改变时的事件处理函数
  - readyState:XMLHttpRequest对象的处理状态
    - 0 :XMLHttpRequest对象还没有完成初始化
    - 1 :XMLHttpRequest对象开始发送请求
    - 2 :XMLHttpRequest对象的请求发送完成
    - 3 :XMLHttpRequest对象开始读取服务器的响应
    - 4 :XMLHttpRequest对象读取服务器响应完成
  - responseText:
    - 用于获取服务器的响应正文.
  - status
    - 服务器返回的响应状态码，只有服务器的响应已经完成时，才会有该状态码
- C.方法
  - open:打开链接.

```
open(请求方式,请求路径,flag);
```

    - flag为true则是异步请求,如果是false则是同步请求
  - send:发送数据

```
send(数据);
```

    - 请求方式为get请求时,不需要通过send方法来发送,直接将参数跟在请求路径后面
    - 请求方式为post请求时,就需要使用send方法
  - setRequestHeader:设置请求头

```
setRequestHeader("头","值");
```

16_Ajax入门案例之get方式

- A.步骤
  - a.创建异步的XMLHttpRequest对象.
  - b.设置监听:监听对象的状态的变化,触发一个函数.

    - c.打开链接:
    - d.发送数据:
- B.实现

```
function createXMLHttpRequest(){
   var xmlHttp;
   try{ // Firefox, Opera 8.0+, Safari
       xmlHttp=new XMLHttpRequest();
   } catch (e){
      try{// Internet Explorer
            xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
        }
       catch (e){
         try{
            xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
         }
         catch (e){}
         }
   }
   return xmlHttp;
}

function ajax_get(){
// * 1.创建异步的XMLHttpRequest对象.
var xhr = createXMLHttpRequest();
// * 2.设置监听:监听对象的状态的变化,触发一个函数.
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4){
        if(xhr.status == 200){
            var data = xhr.responseText;
            // 获得div:
            var div1 = document.getElementById("div1");
            div1.innerHTML = data;
        }
    }
}
// * 3.打开链接:
xhr.open("GET","${pageContext.request.contextPath}/ajaxServletDemo1",true);
// * 4.发送数据:
xhr.send();
}
```

17_Ajax入门案例之post方式

- A.步骤
    - a.创建异步的XMLHttpRequest对象.
    - b.设置监听:监听对象的状态的变化,触发一个函数.
    - c.打开链接:
    - d.设置请求头
    - e.发送数据:

- B.实现

```
function ajax_post() {
    // * 1.创建异步的XMLHttpRequest对象.
    var xhr = createXMLHttpRequest();
    // * 2.设置监听:监听对象的状态的变化,触发一个函数.
    xhr.onreadystatechange = function(){

        if(xhr.readyState == 4){
            if(xhr.status == 200){
                // 获得响应的数据:
                var data = xhr.responseText;
                // 获得div1:
                var div1 = document.getElementById("div1");
                div1.innerHTML = data;
            }
        }

    }
    // * 3.打开链接:
    xhr.open("POST","${ pageContext.request.contextPath }/ajaxServletDemo1",true);
    // 设置请求头:
    xhr.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    // * 4.发送数据:
    //POST请求传递参数:需要将参数写到send方法中.
    xhr.send("id=3&name=李四");

}
```