1. HTML基础

2. CSS

3. JavaScript

3.1. js的简介

1. js是什么

> js是可以嵌入到html中，是 基于对象 和 事件驱动 的 脚本语言

- 特点：
  - (1)交互性
  - (2)安全性：js不能访问本地磁盘
  - (3)跨平台：浏览器中都具备js解析器

2. js能做什么

- (1)js能动态的修改(增删)html和CSS的代码
- (2)能动态的校验数据

3. js历史及组成

> ECMAScript　BOM(浏览器对象模型) DOM(文档对象模型)

4. js被引入的方式

- (1)内嵌脚本

```
<input type="button" value="button" onclick="alert('xxx')" />
```

- (2)内部脚本

```
<script type="text/javascript">alert("xxx");</script>
```

- (3)外部脚本

1. 首先先创建一个js文件
2. 其次在html中引入

```
<script type="text/javascript" src="demo1.js"></script>
```

- js代码放在哪？

> 放在哪都行 但是在不影响html功能的前提下 越晚加载越好

3.2. js基本语法

1. 变量

- (1)

```
var x = 5;
x = 'javascript';
var y = "hello";
var b = true;
```

- (2)弱类型

```
x = 5;
```

2. 原始数据类型

- (1)number:数字类型
- (2)string：字符串类型
- (3)boolean:布尔类型
- (4)null:空类型 object
- (5)underfind:未定义

> **注意：number、boolean、string是伪对象**

- 类型转换：
  - number\boolean转成string
    - toString();
  - string\boolean转成number
    - parseInt()
    - parseFloat()
    - boolean不能转

> string可以将数字字符串转换成number 如果"123a3sd5" 转成123

- 强制转换
  - Boolean()(强转成布尔)
    - 数字强转成布尔 非零就是true 零就是false
    - 字符串强转成布尔 非""(空字符串)就是true 空字符串""就是false
  - Number()(强转成数字)
    - 布尔转数字 true转成1 false转成0
    - 字符串转数字 不能强转

3. 引用数据类型

- java: Object obj = new Object();
- js: var obj = new Object();
- var num = new Number();

4. 运算符

- (1)赋值运算符

```
var x = 5;
```

- (2)算数运算符
  - ■ ■ ■ / %
  - +: 遇到字符串变成连接
  - -：先把字符串转成数字然后进行运算
  - *: 先把字符串转成数字然后进行运算
  - /: 先把字符串转成数字然后进行运算
- (3)逻辑运算符
  - && ||

- (4)比较运算符
  - < > >= <= != ==
  - ===:全等：类型与值都要相等
- (5)三元运算符
  - 3<2?"大于":"小于"
- (6)void运算符

```
<a href="javascript:void(0);">xxxxxx</a>
```

- (7)类型运算符
  - typeof:判断数据类型 返回我的数据类型
  - instanceof：判断数据类型 是否是某种类型
  - var obj = new Object();
  - alert(typeof obj);//object
  - alert(obj instanceof Object);//true

5. 逻辑语句

- (1)if-else
  - //条件:
  - //数字非0 字符串非空====true

```
if(9){
  alert("true--");
  }else{
  alert("false--");
}
```

- (2)switch

```
var x = "java";
switch(x){
    case "CSS":
        alert("CSS");
        break;
    case "js":
        alert("js");
        break;
    case "java":
        alert("java");
        break;
    default:
        alert("def");
}
```

- (3)for

```
for (var i = 0; i < 5; i++) {
    alert(i);
}
```

- (4)for in

```
var arr = [1, 3, 5, 7, "js"];
for (index in arr) {//index代表索引
    //alert(index);
    alert(arr[index]);
}
```

3.3. js的内置对象

- (1)Number
    - 创建方式：

```
var myNum=new Number(value);
var myNum=Number(value);
```

> 属性和方法：

- toString():转成字符串
- valueOf()：返回一个 Number 对象的基本数字值
- (2)Boolean
    - 创建方式：

```
var bool = new Boolean(value);
var bool = Boolean(value);
```

> 属性和方法：

- toString():转成字符串
- valueOf()：返回一个 Boolean 对象的基本值(boolean)
- (3)String

> 当不用 new 运算符调用 String() 时，它只把 s 转换成原始的字符串，并返回转换后的值。

创建方式：

```
var str = new String(s);
var str = String(s);
```

属性和方法：

1. length:字符串的长度
2. charAt():返回索引字符
3. charCodeAt:返回索引字符unicode
4. indexOf():返回字符的索引
5. lastIndexOf();逆向返回字符的索引
6. split();将字符串按照特殊字符切割成数组
7. substr():从起始索引号提取字符串中指定数目的字符
8. substring():提取字符串中两个指定的索引号之间的字符
9. toUpperCase();转大写

- (4)Array

创建方式：

```
var arr = new Array();//空数组
var arr = new Array(size);//创建一个指定长度的数据
var arr = new Array(element0, element1, ..., elementn);//创建数组直接实例化元素
var arr = [];//空数组
var arr = [1,2,5,"java"];//创建数组直接实例化元素
```

属性和方法：

1. length:数组长度
2. join()：把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔一个
3. pop():删除并返回最后元素
4. push()：向数组的末尾添加一个或更多元素，并返回新的长度
5. reverse();反转数组
6. sort();排序

- (5)Date

创建方式：

1. var myDate = new Date();
2. var myDate = new Date(毫秒值);//代表从1970-1-1到现在的一个毫秒值
3.

属性和方法

1. getFullYear():年
2. getMonth():月 0-11
3. getDate():日 1-31
4. getDay()：星期 0-6
5. getTime():返回1970年1月1日午夜到指定日期（字符串）的毫秒数
6. toLocalString();获得本地时间格式的字符串

- (6)Math

> 创建方式：

- Math 对象并不像 Date 和 String 那样是对象的类，因此没有构造函数 Math()，像 Math.sin() 这样的函数只是函数，不是某个对象的方法。您无需创建它，通过把 Math 作为对象使用就可以调用其所有属性和方法。
    - 属性和方法
        - PI：圆周率
        - abs():绝对值
        - ceil():对数进行上舍入
        - floor():对数进行下舍入
        - pow(x,y)：返回 x 的 y 次幂
        - random():0-1之间的随机数
        - round():四舍五入
- (7)RegExp

> 创建方式：

```
var reg = new RegExp(pattern);
var reg = /^正则规则$/;
```

> 规则的写法：

1. [0-9]
2. [A-Z]
3. [a-z]
4. [A-z]
5. \d 代表数据
6. \D：非数字
7. \w：查找单词字符
8. \W：查找非单词字符
9. \s：查找空白字符
10. \S：查找非空白字符
11. n+：出现至少一次
12. n*：出现0次或多次
13. n?：出现0次或1次
14. {5}：出现5

15. {2,8}：2到8次

- 方法：
  - test(str):检索字符串中指定的值。返回 true 或 false

```
需求：校验邮箱：
var email = haohao_827@163.com
var reg = /^[A-z]+[A-z0-9_-]*\@[A-z0-9]+\.[A-z]+$/;
reg.test(email);
```

3.4. js的函数

3.4.1. js函数定义的方式

- (1)普通方式

  > 语法：function 函数名(参数列表){函数体}

```
示例：
function method(){
    alert("xxx");
}
method();
```

- (2)匿名函数

  > 语法：function(参数列表){函数体}

```
示例：
var method = function(){
    alert("yyy");
};
method();
```

- (3)对象函数

  > 语法：new Function(参数1,参数2,...,函数体);
  > 注意：参数名称必须使用字符串形式、最后一个默认是函数体且函数体需要字符串形式

```
示例：
var fn = new Function("a","b","alert(a+b)");
fn(2,5);
```

3.4.2. 函数的参数

- (1)形参没有var去修饰
- (2)形参和实参个数不一定相等
- (3)arguments对象 是个数组 会将传递的实参进行封装

```
function fn(a,b,c){
    //var sum = a+b+c;
    //alert(sum);
    //arguments是个数组 会将传递的实参进行封装
    for(var i=0;i<arguments.length;i++){
        alert(arguments[i]);
    }
}
fn(1,2,4,8);
```

3.4.3. 返回值

- (1)在定义函数的时候不必表明是否具有返回值
- (2)返回值仅仅通过return关键字就可以了 return后的代码不执行

```
function fn(a,b){
    return a+b;
    //alert("xxxx");
}
alert(fn(2,3));
```

3.4.4. js的全局函数

- (1)编码和解码
    - encodeURI()
    - decodeURI()
    - encodeURIComponet()
    - decodeURIComponent()
    - escape()
    - unescape()

    > 三者区别：进行编码的符号范围不同吧，实际开发中常使用第一种

- (2)强制转换
    - Number()
    - String()
    - Boolean()
- (3)转成数字
    - parseInt()

    - parseFloat()
- (4)eval()方法

> 将字符串当作脚本进行解析运行

```
//var str = "var a=2;var b=3;alert(a+b)";
//eval(str);
function print(str) {
    eval(str);
}
print("自定义逻辑");
```

3.5. js事件

> 事件:事件源　响应行为

3.5.1. js的常用事件

- onclick:点击事件
- onchange:域内容被改变的事件

```
<!-- 需求：实现二级联动 -->
<select id= "city">
    <option value="bj">北京</option>
    <option value="tj">天津</option>
    <option value="sh">上海</option>
</select >
<select id="area">
<option>海淀</option>
<option>朝阳</option>
<option>东城</option>
</select>
<script type="text/javascript">
    var select = document.getElementById("city");
    select.onchange = function () {
        var optionVal = select.value;
        switch (optionVal) {
            case 'bj':
                var area = document.getElementById("area");
                area.innerHTML = "<option>海淀</option><option>朝阳</option>
<option>东城</option>";
                break;
            case 'tj':
                var area = document.getElementById("area");
                area.innerHTML = "<option>南开</option><option>西青</option>
<option>河西</option>";
                break;
            case 'sh':
```

```
                var area = document.getElementById("area");
                area.innerHTML = "<option>浦东</option><option>杨浦</option>";
                break;
            default:
                alert("error");
        }
    };
</script>
```

- onfoucus:获得焦点的事件
- onblur:失去焦点的事件

> 需求：当输入框获得焦点的时候，提示输入的内容格式;当输入框失去焦点的时候，提示输入有误

```
<label for="txt">name</label>
    <input id="txt" type="text" />
    <span id="action"></span>
    <script type="text/javascript">
        var txt = document.getElementById("txt");
        txt.onfocus = function () {
            //友好提示
            var span = document.getElementById("action");
            span.innerHTML = "用户名格式最小8位";
            span.style.color = "green";
        };
        txt.onblur = function () {
            //错误提示
            var span = document.getElementById("action");
            span.innerHTML = "对不起 格式不正确";
            span.style.color = "red";
        };
    </script>
```

- onmouseover:鼠标悬浮的事件
- onmouseout:鼠标离开的事件

> 需求：div元素 鼠标移入变为绿色 移出恢复原色

```
#d1{background-color: red;width:200px;height: 200px;}
<div id="d1"></div>
<script type="text/javascript">
    var div = document.getElementById("d1");
    div.onmouseover = function(){
        this.style.backgroundColor = "green";
    };
    div.onmouseout = function(){
        this.style.backgroundColor = "red";
```

```
    };
</script>
```

- onload:加载完毕的事件,等到页面加载完毕再执行onload事件所指向的函数

```
<span id="span"></span>
<script type="text/javascript">
    window.onload = function(){
        var span = document.getElementById("span");
        alert(span);
        span.innerHTML = "hello js";
    };
</script>
```

3.5.2. 事件的绑定方式

- (1)将事件和响应行为都内嵌到html标签中

```
<input type="button" value="button"  onclick="alert('xxx')"/>
```

- (2)将事件内嵌到html中而响应行为用函数进行封装

```
<input type="button" value="button" onclick="fn()" />
                        <script type="text/javascript">
                                function fn(){
                                        alert("yyy");
                                }
                        </script>
```

- (3)将事件和响应行为 与html标签完全分离

```
<input id="btn" type="button" value="button"/>
<script type="text/javascript">
                                var btn = document.getElementById("btn");
                                btn.onclick = function(){
                                        alert("zzz");
                                };
                        </script>
```

- **this关键字**

> this经过事件的函数进行传递的是html标签对象

```
<input id="btn" name="mybtn" type="button" value="button123"
onclick="fn(this)"/>
<script type="text/javascript">
    function fn(obj){
    alert(obj.name);
}
</script>
```

3.5.3. 阻止事件的默认行为

1. IE：window.event.returnValue = false;
2. W3c: 传递过来的事件对象.preventDefault();

```
//ie: window.event.returnValue = false;
//W3c: 传递过来的事件对象.preventDefault();
//W3c标准
if(e&&e.preventDefault){
    alert("w3c");
    e.preventDefault();
//IE标签
}else{
    alert("ie");
    window.event.returnValue = false;
}


//通过事件返回false也可以阻止事件的默认行为
<a href="demo11.html" onclick="return false">点击我吧</a>
```

3.5.4. 阻止事件的传播

- IE：window.event.cancelBubble = true;
- W3c: 传递过来的事件对象.stopPropagation();

```
if(e&&e.stopPropagation){
    alert("w3c");
    e.stopPropagation();
//IE标签
}else{
    alert("ie");
    window.event.cancelBubble = true;
}
```

```
    <div style="width:300px;height:300px;background-color:blue;padding:50px"
```

```
onclick="f1()">
        <div style="width:300px;height:300px;background-color:red;"
onclick="f2(event)"></div>
    </div>
    <script type="text/javascript">
    function f1(){
        log("div1")
    }
    function f2(e){
        if (e && e.stopPropagation) {
            log("w3c");
            e.stopPropagation();
            //IE标签
        }
        else {
            log("ie");
            var r = window.event.cancelBubble = true;
            log(r)
        }
    }

    </script>
```

3.6. js的bom

- (1)window对象
  - 弹框的方法：
    - 提示框：alert("提示信息");
    - 确认框：confirm("确认信息");
      - 有返回值：如果点击确认返回true 如果点击取消 返回false
    - 输入框：prompt("提示信息");
      - 有返回值：如果点击确认返回输入框的文本 点击取消返回null
  - open方法：
    - window.open("url地址");
    - open("../jsCore/demo10.html");

```
var res = confirm("您确认要删除吗？");
alert(res);

var res =  prompt("请输入密码？");
alert(res);
```

- 定时器：
  - setTimeout(函数,毫秒值);
  - clearTimeout(定时器的名称);清除定时器

```
setTimeout(
    function(){
        alert("xx");
    },
    3000
);
```

```
var i = 0;
var fn = function(){
    log(i++);
    setTimeout(fn, 500);
}
fn();
```

```
var timer;
var fn = function(){
    alert("x");
    timer = setTimeout(fn,2000);
};
var closer = function(){
    clearTimeout(timer);
};
fn();
setInterval(函数,毫秒值);
clearInterval(定时器的名称)
var timer = setInterval(
function(){
    alert("nihao");
},
2000
);
var closer = function(){
clearInterval(timer);
};
```

```
<!-- 需求：注册后5秒钟跳转首页 -->
<span id="second" style="color: red;">5</span>秒后跳转到首页，如果不跳转请
    <a href="../jsCore/demo10.html">点击这里</a>
    <script type="text/javascript">
            var time = 5;
            var timer;
            timer = setInterval(
                function () {
                    var second = document.getElementById("second");
```

```
                    if (time >= 1) {
                        second.innerHTML = time;
                        time--;
                    } else {
                        clearInterval(timer);
                        location.href = "../jsCore/demo10.html";
                    }
                },
                1000
            );
</script>
```

- (2)location
  - location.href="url地址";
- (3)history
  - back();
  - forward();
  - go();

```
<a href="demo7.html">后一页</a>
<input type="button" value="上一页" onclick="history.back()">
<input type="button" value="下一页" onclick="history.forward()">
<input type="button" value="上一页" onclick="history.go(-1)">
<input type="button" value="下一页" onclick="history.go(1)">
```

3.7. js的dom

- 理解一下文档对象模型
  - html文件加载到内存之后会形成一颗dom树，根据这些节点对象可以进行脚本代码的动态修改
  - 在dom树当中 一切皆为节点对象
- dom方法和属性
  - 笔记见代码