今日内容 1. JavaScript基础

JavaScript：

* 概念：　一门客户端脚本语言
    * 运行在客户端浏览器中的。每一个浏览器都有JavaScript的解析引擎
    * 脚本语言：不需要编译，直接就可以被浏览器解析执行了

* 功能：
    * 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。

* JavaScript发展史：
    1. 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
    2. 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
    3. 1996年，微软抄袭JavaScript开发出JScript语言
    4. 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。

    * JavaScript = ECMAScript + JavaScript自己特有的东西(BOM+DOM)

* ECMAScript：客户端脚本语言的标准

    ECMAScript是一种由Ecma国际（前身为欧洲计算机制造商协会,英文名称是European Computer Manufacturers Association）通过ECMA-262标准化的脚本程序设计语言。这种语言在万维网上应用广泛，它往往被称为JavaScript或JScript，所以它可以理解为是JavaScript的一个标准,但实际上后两者是ECMA-262标准的实现和扩展。

    1. 基本语法：
        1. 与html结合方式
            1. 内部JS：
                * 定义<script>，标签体内容就是js代码
            2. 外部JS：
                * 定义<script>，通过src属性引入外部的js文件

            * 注意：
                1. <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。
                2. <script>可以定义多个。
        2. 注释
            1. 单行注释：//注释内容
            2. 多行注释：/*注释内容*/
        3. 数据类型：
            1. 原始数据类型(基本数据类型)：
                1. number：数字。 整数/小数/NaN(not a number 一个不是数字的数字类型)
                2. string：字符串。 字符串　"abc" "a" 'abc'
                3. boolean：true和false
                4. null：一个对象为空的占位符
                5. 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

            2. 引用数据类型：对象

        4. 变量
            * 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;

* typeof运算符：获取变量的类型。
    * 注：null运算后得到的是object

5. 运算符
    1. 一元运算符：只有一个运算数的运算符
        ++，-- ， +(正号)
        * ++ --: 自增(自减)
            * ++(--) 在前，先自增(自减)，再运算
            * ++(--) 在后，先运算，再自增(自减)
        * +(-)：正负号
        * 注意：在JS中，如果运算数不是运算符所要求的类型，那么js引擎会自动的将运算数进行类型转换
            * 其他类型转number：
                * string转number：按照字面值转换。如果字面值不是数字，则转为NaN（不是数字的数字）
                * boolean转number：true转为1，false转为0
                *

         * NaN
      Javascipt 裡面有一種型別在你轉成整數 parseInt或轉成浮點數 parseFloat時會出現：

            var p= parseInt("abc32564");
            alert(p);
            結果：NaN

            出現NaN這個型別代表輸出失敗，而NaN是什麼？

            NaN代表的就是「Not a Number」 ，不是一個數字。要檢查是不是NaN可用函數 isNaN來檢查：

            if(isNaN(p)){ alert("請輸入數值"); return false;}
            NaN比對範例
            基本上只要結果不是數值的就是NaN，但有一些「特別的」例外，例如空字串或真假(true/false)

            isNaN == false，是數值

            isNaN(123) //false
            isNaN(-1.23) //false
            isNaN(5-2) //false
            isNaN(0) //false
            isNaN(\0x32) //false 十六進位
            isNaN('123') //false
            isNaN('') //false
            isNaN(null) //false
            isNaN(true) //false
    2. 算数运算符
        + - * / % ...

    3. 赋值运算符
        = += -+....

    4. 比较运算符
        > < >= <= == ===(全等于)

        * 比较方式
          1. 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。
          2. 类型不同：先进行类型转换，再比较
              * ===：全等于。在比较之前，先判断类型，如果类型不一样，则直接返回false

    5. 逻辑运算符
        && || !
        * 其他类型转boolean：
            1. number：0或NaN为假，其他为真
            2. string：除了空字符串("")，其他都是true
            3. null&undefined:都是false
            4. 对象：所有对象都为true

    6. 三元运算符
        ? : 表达式
        ```
        var a = 3;
        var b = 4;

        var c = a > b ? 1:0;
        ```
        * 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；

6. 流程控制语句：
    1. if...else...
    2. 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)
            * switch(变量):
                case 值:
        * 在JS中,switch语句可以接受任意的原始数据类型
    3. while
    4. do...while
    5. for
7. JS特殊语法：
    1. 语句以;结尾，如果一行只有一条语句则 ;可以省略 (不建议)
    2. 变量的定义使用var关键字，也可以不使用
        * 用： 定义的变量是局部变量
        * 不用：定义的变量是全局变量(不建议)

8. 练习：99乘法表
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>99乘法表</title>
    <style>
        td{
            border: 1px solid;
        }
```

```
        </style>

        <script>

            document.write("<table   align='center'>");


            //1.完成基本的for循环嵌套，展示乘法表
            for (var i = 1; i <= 9 ; i++) {
                document.write("<tr>");
                for (var j = 1; j <=i ; j++) {
                    document.write("<td>");

                    //输出  1 * 1 = 1
                    document.write(i + " * " + j + " = " + ( i*j) +"   ");

                    document.write("</td>");
                }
                /*//输出换行
                document.write("<br>");*/

                document.write("</tr>");
            }

            //2.完成表格嵌套
            document.write("</table>");

        </script>
    </head>
    <body>

    </body>
    </html>
```

2. 基本对象：根据vue.js，js应该可以创建自己的对象，类似于定义类
    1. Function：函数(方法)对象
        1. 创建：
            1. var fun = new Function(形式参数列表,方法体); //忘掉吧
            2.
                function 方法名称(形式参数列表){
                    方法体
                }

            3.
               var 方法名 = function(形式参数列表){
                    方法体
               }
        2. 方法：

        3. 属性：
            length:代表形参的个数
        4. 特点：

1. 方法定义是，形参的类型不用写,返回值类型也不写。
2. 方法是一个对象，如果定义名称相同的方法，会覆盖
3. 在JS中，方法的调用只与方法的名称有关，和参数列表无关
4. 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

5. 调用:

方法名称(实际参数列表);

2. Array:数组对象
    1. 创建:
        1. var arr = new Array(元素列表);
        2. var arr = new Array(默认长度);
        3. var arr = [元素列表];
    2. 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。
    3. 属性
        length:数组的长度
    4. 特点:
        1. JS中，数组元素的类型可变的。
        2. JS中，数组长度可变的。
3. Boolean
4. Date：日期对象
    1. 创建:

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

    2. 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
5. Math：数学对象
    1. 创建:
        * 特点：Math对象不用创建，直接使用。 Math.方法名();

    2. 方法:
        random():返回 0 ~ 1 之间的随机数。 含0不含1
        ceil(x)：对数进行上舍入。
        floor(x)：对数进行下舍入。
        round(x)：把数四舍五入为最接近的整数。
    3. 属性:
        PI
6. Number
7. String
8. RegExp：正则表达式对象
    1. 正则表达式：定义字符串的组成规则。
        1. 单个字符:[]
            如： [a] [ab] [a-zA-Z0-9_]
            * 特殊符号代表特殊含义的单个字符:
                \d:单个数字字符 [0-9]
                \w:单个单词字符[a-zA-Z0-9_]
        2. 量词符号:
            ?：表示出现0次或1次
            *：表示出现0次或多次
            +：出现1次或多次

{m,n}:表示 m<= 数量 <= n
* m如果缺省： {,n}:最多n次
* n如果缺省: {m,} 最少m次
3. 开始结束符号
* ^:开始
* $:结束
2. 正则对象:
1. 创建
1. var reg = new RegExp("正则表达式");
2. var reg = /正则表达式/;
2. 方法
1. test(参数):验证指定的字符串是否符合正则定义的规范
9. Global
1. 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。 方法名();
2. 方法:
encodeURI():url编码
decodeURI():url解码

encodeURIComponent():url编码,编码的字符更多
decodeURIComponent():url解码

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==比较全部问false

eval():讲 JavaScript 字符串，并把它作为脚本代码来执行。
3. URL编码
传智播客 = %E4%BC%A0%E6%99%BA%E6%92%AD%E5%AE%A2

* BOM

* DOM

今日内容:

```
1. JavaScript:
    1. ECMAScript:
    2. BOM:
    3. DOM:
        1. 事件
```

DOM简单学习：为了满足案例要求

```
* 功能：控制html文档的内容
* 获取页面标签(元素)对象：Element
    * document.getElementById("id值"):通过元素的id获取元素对象

* 操作Element对象：
    1. 修改属性值：
        1. 明确获取的对象是哪一个？
        2. 查看API文档，找其中有哪些属性可以设置
    2. 修改标签体内容：
        * 属性：innerHTML
        1. 获取元素对象
        2. 使用innerHTML属性修改标签体内容
```

事件简单学习

```
* 功能： 某些组件被执行了某些操作后，触发某些代码的执行。
    * 造句： xxx被xxx,我就xxx
        * 我方水晶被摧毁后，我就责备对友。
        * 敌方水晶被摧毁后，我就夸奖自己。

* 如何绑定事件
    1. 直接在html标签上，指定事件的属性(操作)，属性值就是js代码
        1. 事件：onclick--- 单击事件
        <img id="light" src="img/off.gif"  onclick="fun();">

    2. 通过js获取元素对象，指定事件属性，设置一个函数

    * 代码：
        <body>
            <img id="light" src="img/off.gif"  onclick="fun();">
            <img id="light2" src="img/off.gif">

            <script>
                function fun(){
                    alert('我被点了');
                    alert('我又被点了');
                }
```

```
                function fun2(){
                    alert('咋老点我？');
                }

                //1.获取light2对象
                var light2 = document.getElementById("light2");
                //2.绑定事件
                light2.onclick = fun2;


            </script>
        </body>

* 案例1：电灯开关
    <!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>电灯开关</title>

    </head>
    <body>

    <img id="light" src="img/off.gif">

    <script>
        /*
            分析：
                1.获取图片对象
                2.绑定单击事件
                3.每次点击切换图片
                    * 规则：
                        * 如果灯是开的 on,切换图片为 off
                        * 如果灯是关的 off,切换图片为 on
                    * 使用标记flag来完成

         */

        //1.获取图片对象
        var light = document.getElementById("light");

        var flag = false;//代表灯是灭的。 off图片

        //2.绑定单击事件
        light.onclick = function(){
            if(flag){//判断如果灯是开的，则灭掉
                light.src = "img/off.gif";
                flag = false;

            }else{
                //如果灯是灭的，则打开
```

```
                light.src = "img/on.gif";
                flag = true;
            }


        }

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

BOM:

1. 概念：Browser Object Model 浏览器对象模型
    * 将浏览器的各个组成部分封装成对象。

2. 组成：
    * Window：窗口对象
    * Navigator：浏览器对象
    * Screen：显示器屏幕对象
    * History：历史记录对象
    * Location：地址栏对象

3. Window：窗口对象
    1. 创建
    2. 方法
        1. 与弹出框有关的方法：
            alert() 显示带有一段消息和一个确认按钮的警告框。
            confirm() 显示带有一段消息以及确认按钮和取消按钮的对话框。
                * 如果用户点击确定按钮，则方法返回true
                * 如果用户点击取消按钮，则方法返回false
            prompt() 显示可提示用户输入的对话框。
                * 返回值：获取用户输入的值
        2. 与打开关闭有关的方法：
            close() 关闭浏览器窗口。
                * 谁调用我 ，我关谁
            open() 打开一个新的浏览器窗口
                * 返回新的Window对象
        3. 与定时器有关的方式
            setTimeout() 在指定的毫秒数后调用函数或计算表达式。
                * 参数：
                    1. js代码或者方法对象
                    2. 毫秒值
                * 返回值：唯一标识，用于取消定时器
            clearTimeout() 取消由 setTimeout() 方法设置的 timeout。

            setInterval() 按照指定的周期（以毫秒计）来调用函数或计算表达式。
            clearInterval() 取消由 setInterval() 设置的 timeout。

    3. 属性:
        1. 获取其他BOM对象:
            history
            location
            Navigator
            Screen:
        2. 获取DOM对象
            document
    4. 特点
        * Window对象不需要创建可以直接使用 window使用。 window.方法名();
        * window引用可以省略。 方法名();

    5. 轮换图案例
            1.在页面上使用img标签展示图片
            2.定义一个方法，修改图片对象的src属性
            3.定义一个定时器，每隔3秒调用方法一次。

4. Location: 地址栏对象
    1. 创建(获取):
        1. window.location
        2. location

    2. 方法:
        * reload() 重新加载当前文档。刷新
    3. 属性
        * href 设置或返回完整的 URL。

5. History: 历史记录对象
    1. 创建(获取):
        1. window.history
        2. history

    2. 方法:
        * back() 加载 history 列表中的前一个 URL。
        * forward() 加载 history 列表中的下一个 URL。
        * go(参数) 加载 history 列表中的某个具体页面。
            * 参数:
                * 正数: 前进几个历史记录
                * 负数: 后退几个历史记录
    3. 属性:
        * length 返回当前窗口历史列表中的 URL 数量。

DOM:

* 概念: Document Object Model 文档对象模型
    * 将标记语言文档的各个组成部分，封装为对象。可以使用这些对象，对标记语言文档进行CRUD的动态操作

* W3C DOM 标准被分为 3 个不同的部分:

    * 核心 DOM - 针对任何结构化文档的标准模型
        * Document：文档对象
        * Element：元素对象
        * Attribute：属性对象
        * Text：文本对象
        * Comment:注释对象

        * Node：节点对象，其他5个的父对象
    * XML DOM - 针对 XML 文档的标准模型
    * HTML DOM - 针对 HTML 文档的标准模型

* 核心DOM模型：
    * Document：文档对象
        1. 创建(获取)：在html dom模型中可以使用window对象来获取
            1. window.document
            2. document
        2. 方法：
            1. 获取Element对象：
                1. getElementById() ： 根据id属性值获取元素对象。id属性值一般唯一
                2. getElementsByTagName()：根据元素名称获取元素对象们。返回值是一个数组
                3. getElementsByClassName():根据Class属性值获取元素对象们。返回值是一个数组
                4. getElementsByName(): 根据name属性值获取元素对象们。返回值是一个数组
            2. 创建其他DOM对象：
                createAttribute(name)
                createComment()
                createElement()
                createTextNode()

                ```
                var td_gender = document.createElement("td");
                var text_gender = document.createTextNode(gender);
                td_gender.appendChild(text_gender);
                ```
        3. 属性
    * Element：元素对象
        1. 获取/创建：通过document来获取和创建
        2. 方法：
            1. removeAttribute()：删除属性
            2. setAttribute()：设置属性
    * Node：节点对象，其他5个的父对象
        * 特点：所有dom对象都可以被认为是一个节点
        * 方法：
            * CRUD dom树：
                * appendChild()：向节点的子节点列表的结尾添加新的子节点。
                * removeChild() ：删除（并返回）当前节点的指定子节点。
                * replaceChild()：用新节点替换一个子节点。
        * 属性：
            * parentNode 返回节点的父节点。

* HTML DOM
    * 作用
    1. 标签体(内部)的设置和获取：innerHTML
    2. 使用html元素对象的属性
    3. 控制元素样式
        1. 使用元素的style属性来设置
            如：

```
//修改样式方式1
div1.style.border = "1px solid red";
div1.style.width = "200px";
//font-size--> fontSize
div1.style.fontSize = "20px";
```

        2. 提前定义好类选择器的样式，通过元素的className属性来设置其class属性值。

事件监听机制：

* 概念：某些组件被执行了某些操作后，触发某些代码的执行。
    * 事件：某些操作。如： 单击，双击，键盘按下了，鼠标移动了
    * 事件源：组件。如： 按钮 文本输入框...
    * 监听器：代码。
    * 注册监听：将事件，事件源，监听器结合在一起。 当事件源上发生了某个事件，则触发执行某个监听器代码。

* 常见的事件：
    1. 点击事件：
        1. onclick：单击事件
        2. ondblclick：双击事件
    2. 焦点事件
        1. onblur：失去焦点
        2. onfocus:元素获得焦点。

    3. 加载事件：
        1. onload：一张页面或一幅图像完成加载。

    4. 鼠标事件：
        1. onmousedown 鼠标按钮被按下。
        2. onmouseup 鼠标按键被松开。
        3. onmousemove 鼠标被移动。
        4. onmouseover 鼠标移到某元素之上。
        5. onmouseout 鼠标从某元素移开。

    5. 键盘事件：
        1. onkeydown 某个键盘按键被按下。
        2. onkeyup 某个键盘按键被松开。
        3. onkeypress 某个键盘按键被按下并松开。

    6. 选择和改变
        1. onchange 域的内容被改变。
        2. onselect 文本被选中。

7. 表单事件：
    1. onsubmit 确认按钮被点击。
    2. onreset 重置按钮被点击。