# 第一天

1、Javascript 概述

1、什么是JavaScript

简称 JS

一种运行于 JS解释器/引擎 解释型脚本语言

1、JS 解释器 / 引擎

2、解释型脚本语言

1、编译型

C/C++

先编译再执行

2、解释型

无需编译，直接执行，碰到错误则停止

JS解释器：

1、独立安装的JS解释器

NodeJS

2、嵌入在浏览器内核中的JS解释器

2、JS发展史

1、1992年 Nombas --> CMM(C--) -->ScriptEase

2、1995年 Netscape(网景) --> LiveScript --> Javascript

3、1996年 Microsoft --> JScript

4、1997年 Javascript1.1版本 作为草案 提交了 ECMA(欧洲计算机制造商协会) -->ECMA-262标准-->ECMAScript

完整的JS由三部分组成

1、JS核心-ECMAScript (ES)

2、文档对象模型(DOM : Document Object Model)

3、浏览器对象模型(BOM : Browser Object Model)

3、JS特点

1、任何文本编辑工具都可以编写

2、无需编译，由JS解释器负责运行

3、弱类型语言

由数据规范类型

强类型语言：由数据类型来规范数据

4、面向对象

4、JS用途

1、客户端JS(前端)

1、数据计算

2、表单验证

3、异步提交数据(Ajax)

4、事件的触发和处理

5、特效

2、服务器端JS(后端)

1、分布式运算

2、实时服务器

3、窗口应用

4、网络应用

2、JS语法

1、使用JS

1、浏览器内核

作用：负责渲染页面.由两部分组成

1、内容排版引擎：解析HTML与CSS

2、脚本执行引擎：解析JS

浏览器 内核 排版引擎 脚本引擎

IE Trident - Chakra

Firefox Gecko - Monkey~

Safari Webkit Webcore Nitro

Chrome Webkit Webcore V8

Opera Presto - Carakan

Opera(2013) Webkit Webcore V8

2、JS环境搭建-运行

1、独立安装的解释器 - NodeJS

1、开始菜单中 --> cmd

2、嵌入在浏览器中的引擎

1、使用浏览器控制台输入代码并执行

2、将JS脚本嵌入在HTML页面中(重要)

1、html元素事件中执行JS脚本

onclick事件：单击元素时所做的操作。

<button onclick="js脚本">测试</button>

2、html页面中 通过 <script></script>编写JS脚本

3、将JS脚本 编写在外部脚本文件中(js文件)

document.write("output...");

代码： console.log("待输出内容");

作用：在控制台上输出一句话

2、调试JS

1、代码错误

<script></script> 语句块1

<script></script> 语句块2

代码一旦碰到错误，会终止本语句块的执行，但是其他语句块的执行。

3、语法规范

1、基本规范

语句 ：组成JS的最基本的单位

console.log();

document.write();

var a = 10;

var r = a + b / c;

由 表达式、关键字、运算符

关键字：由JS定义，具备特殊功能的字符，称之为关键字

严格区分大小写

console.log("");

Console.log("");//错误

语句使用 ; 表示结尾

注释：

单行注释：//

多行注释：/\* \*/

3、变量和常量(重点)

1、什么是变量

内存：临时性保存，保存程序在运行过程中所需要用到的数据。

硬盘：永久性保存数据

内存空间：内存中，保存数据的一个空间。

变量：内存中的一个存储空间，一个可以变化的数据。

变量名：一个变量(内存空间)的别名

值：变量中所保存的数据

2、变量的声明

1、什么是变量的声明

如何到计算机内存中，申请一段空间用来保存数据

2、语法

var 变量名;

var 变量名1,变量名2,...;

eg:

1、声明一个空间，用于保存用户的名称

var userName;

2、声明几个空间，用于保存用户的年龄，性别，工资

var age,gender,salary;

3、赋值

语法：

变量名=值;

1、声明一个变量，用于保存年龄，并且赋值为 18

var age;

age=18;

声明并且赋值

var 变量名=值;

var age=18;

注意：

1、值为 数字

var age=18;

var score=98.5;

2、值为 非数字，为字符串时

必须使用"" 或 '' 包裹起来

var userName="张无忌";

4、打印输出 变量

console.log(变量名);

document.write(变量名);

var age = 18;

console.log(age); // 18

console.log("age");//age

5、变量名命名规范

1、不允许使用JS关键字以及保留关键字做变量名

2、可以包含字母、数字、下划线(\_)以及$

3、不能以数字开头

4、最好见名知意

5、命名方式

1、匈牙利命名法

txt : 文本框，密码框，...

rdo : 单选按钮

chk : 复选框

...

在获取页面元素时使用

2、下划线命名法

var \_userName;

var userName;

3、驼峰命名法

变量名称由多个单词组成，第一个单词所有字母全小写，从第二个单词开始，每个单词的首字符变大写。

var userName;

3、变量的使用

1、变量未经初始化就使用

只声明，未赋值的变量 结果为 undefined

2、未经声明的变量

报错

3、声明变量不使用var关键字

gender="男";

如果声明变量时不使用var关键字而直接赋值的话，声明的是一个全局变量。容易引发"全局污染"的效果。

声明变量时，尽量使用 var 关键字

4、变量的存取操作

1、存 - 为变量赋值

变量 出现在 = 的左边，一定是赋值

var userName="张无忌";

userName="殷素素";

2、取 - 获取变量的值

1、放在函数中使用

document.write(变量);

console.log(变量);

2、变量出现在 = 的右边，一定是取值

var userName="殷素素";//赋值

var newName=userName;

userName:取值操作

newName:赋值操作

有 = 的时候，一定是先计算=右边的表达式或变量的值，再将其赋值给=左边的变量。而且=的左边，必须是变量

4、运算符

1、赋值运算符

=

将=右边的内容赋值给左边的变量

由赋值运算符组成的表达式称为"赋值表达式"

赋值表达式的值为 右侧表达式计算出来的结果

2、算术运算符

+,-,\*,/,%

1、%

取余数，俗称：模

5 % 2 : 结果为 1

4 % 2 : 结果为 0

2 % 5 : 结果为 2

场合：

1、判断某数字的奇偶性

55 % 2 结果: 1

2、判断某数字是不是n的倍数

判断 188 是不是 3 的倍数

判断 188 和 3 取余，结果是否为0，如果为0则为倍数。

3、获取某数字的最后几位

获取 1234 最后一位

1234 % 10 结果 4

获取 1234 最后两位

1234 % 100 结果 34

2、+

作用：

1、做 加法 操作

要求：+两边的操作数必须是数字

2、做 连接

如果 + 两边的操作数，有一个是字符串类型的，那么就是连接的效果

3、优先级

先算 \*/%

后算 +-

平级时：从左向右算

37+25/5\*6+12%3 ：67

可以通过 () 提升优先级

(37+25)/5\*((6+12)%3)

由算术运算符组成的表达式称之为 算术表达式。

5、常量

1 、什么是常量

一经声明，就不允许被修改的数据，称之为常量

2、目的

为了保证数据的准确性

3、语法

const 常量名=值;

注意：常量名，一般采用全大写方式表示

练习：声明一个变量，用于保存圆的半径r(数值自定)，计算该圆的周长和面积

周长：2\*PI\*半径

面积：PI\*半径\*半径

4、数据类型

问题：想在内存中保存一个数字，开辟多大空间比较合适。

1 、什么是数据类型

保存的数据的类型是什么

2、数据类型分类

1、原始类型(基本类型)

1、number类型

数字类型

可以表示32位(4字节)的整数，或64位(8字节)的浮点数(小数)

2、string 类型

字符串类型

注意：使用时，必须用""或''引起来

var num = 18;//number类型

var num1 = "18"; //string类型

转义字符

\n:换行

\r:回车

\t:制表符

\\:\

\':'

\":"

3、boolean 类型

布尔类型

作用：在程序中，保存真或假的结果，多数用于表示条件的结果

取值：

true ： 真，肯定，是

false ： 假，否定，否

注意：

boolean类型可以参与到整数的+-\*/%的运算

true:1

false:0

4、undefined

声明变量未赋值

访问的对象的属性不存在

5、null

空

2、引用类型

# 第二天

1、回顾

1、使用JS

1、独立的JS解释器

NodeJS

2、浏览器->脚本执行引擎

1、html元素事件

<button onclick="JS脚本"></button>

2、<script></script>元素中嵌入脚本

特点：所有的代码都会执行

3、外部JS文件中嵌入脚本

<script src=""></script>

2、调试

3、语法规范

1、JS代码组成

1、语句

console.log();

document.write();

var userName="";

var a = b + c;

表达式、关键字、运算符

区分大小写

都是以 ; 结尾

2、注释

// 单行注释

/\*\*/ 多行注释

4、变量

1、什么是变量

一个可以变化的数据

变量名：数据的名称，内存空间的别名

值：保存在变量中的数据，可以变的。

2、声明变量

var 变量名;

var 变量1,变量2,...;

3、为变量赋值

使用 = 为变量赋值

变量名=值;

var 变量名=值;

var 变量1=值,变量2=值;

4、变量名命名规范

1、不能使用JS关键字和保留关键字

2、由字母、数字、下划线 \_ 、$组成

3、不能以数字开头

4、最好见名知意

5、可以采用 “匈牙利”，“下划线”，“驼峰命名法”

5、变量的使用

1、使用未初始化的变量

声明好了，但是没有做第一次的赋值

var userName;

console.log(userName);

值为：undefined

var userName="";

console.log(userName);

值为：

var userAge;

console.log(userAge);

值为：undefined

var userAge = 0;

console.log(userAge);

值为：0

2、使用未声明过的变量

console.log(userName);

结果为：报错

3、声明变量不带 var

声明的是 ”全局变量“，容易产生 “全局污染”。

建议：声明变量时，尽量带着var

4、变量的存取操作

1、存 - 为变量赋值

特点：变量出现在赋值符号(=)的左边

2、取 - 使用变量的值

1、配合函数使用

console.log(userName);

document.write(userAge);

2、变量出现在赋值符号的右边

var a = 15;

var b = a;

5、常量

1、什么是常量

一旦声明之后，就不允许被修改的数据就是常量

2、目的

保护数据的正确性

3、声明常量

const 常量名=值;

const DAYOFWEEK=7;

DAYOFWEEK=14;

console.log(DAYOFWEEK);

结果为：7

6、数据类型

1、什么是数据类型

保存在内存中的数据的类型

2、JS数据类型

1、原始类型(基本类型)

1、number

数字类型

2、string 类型

字符串

注意：使用时，必须使用"" 或 ''引起来

var age = 18;//number

var myAge = "18";//string

转义字符：

\n : 换行

\r : 回车

\t : 制表符

\\ : \

\" : "

\' : '

3、boolean 类型

布尔类型

只负责描述 真 或 假 的结果

取值：

true 真

false 假

boolean 是允许参与到number类型的运算中。

true 当成 1

false 当成 0、

4、undefined

声明未赋值

访问对象的属性不存在

5、null

空

2、引用类型

7、运算符

1、赋值运算符

=

先算右边，再赋值给左边

2、算术运算符

+,-,\*,/,%

% : 取余

+ :

只要有操作数是 string 类型，一定是连接操作。

2、数据类型转换

1、弱类型

变量的 数据类型 取决于 值

2、typeof 函数(运算符)

作用：查看当前变量的数据类型

语法：

1、typeof(变量名)

2、typeof 变量名

3、隐式转换

1、问题

var userName="马莲梦露";

var userAge=18;

var result = userName+userAge;

2、什么是隐式转换

不同数据类型的数据在做运算的时候可以进行默认的数据类型的转换

3、隐式转换

NaN : Not a Number (不是一个数字)

isNaN(变量) :是不是一个数字

false : 是数字

true : 不是数字

1、string 类型转换

var a = "18";

var b = "18";

var r1 = a + b; //1818

var r2 = a - b; //?

var r3 = a \* b; //?

var r4 = a / b; //?

var r5 = a % b; //?

值为数字的字符串，在做-，\*，/，%的操作时，会自动转换成 number 类型进行运算,+仍然是 连接的操作

2、JS中，所有 用户手动输入的数据，默认都是 string 类型

接受用户输入数据：

1、window.prompt("提示的文本"[,默认值]);

2、prompt("提示的文本"[,默认值]);

4、显示转换

由转换函数 完成 显示转换

1、toString()

转换成 字符串， 将任意类型的数据转换成 string 类型

var num = 15;

num.toString();

2、parseInt(参数)

将 参数 转换成 整数

碰都第一个非数字的字符，则停止转换。

3、parseFloat(参数)

将 参数 转换成 小数

3、函数

1、使用过的函数

console.log();

document.write();

window.prompt();

parseInt();

parseFloat();

2、为什么要使用函数

1、放在 <script>中的js脚本，在网页加载时就执行-执行时机没法选择。

2、代码冗余 - 将代码写在函数事件中。

3、可重用性不高

3、什么是函数

函数(Function):也被称之为方法(Method),或者过程(Procedure)

函数 是一段 被预定义好，并且可以被反复使用的代码块。

预定义：先编写完成，但不会被网页加载时就执行的代码。

反复使用：可以被无限次的使用

代码块：将多行代码放在一起去管理

函数其实就是一个完整的且功能独立的 功能体

4、函数的声明

使用 function 声明一个函数

语法：

function 函数名称(){

代码块;

可执行的操作语句;

}

练习：

1、编写一个函数，用于向页面输出 Hello World

2、编写一个函数，接收用户输入的数据，向控制台上打印输出

5、函数的调用

JS脚本中，通过 函数名() 进行函数的调用

练习：

声明一个函数，在函数体内，要求用户输入两个数字，计算两个数字的 +，-，\*，/，% 的结果，并且打印输出以上结果。

在页面上定义一个按钮，实现对以上函数的调用

6、定义参数

参数：函数在运行过程中，需要用到的数据进行辅助操作，数据就是参数。

榨汁机 -> 函数

榨汁机 -> 需要水果进行辅助操作

西瓜 --> 西瓜汁

葡萄 --> 葡萄汁

榴莲 --> 榴莲汁

水果 --> 参数

parseInt(25.5);

1、语法

function 函数名(参数列表){

//函数体

}

function 函数名(参数名1,参数名2){}

function zha(fruit){

console.log(fruit);

document.write(fruit);

}

定义函数的时候声明的参数-> 形参

调用带参函数：

函数名(参数值列表);

注意：调用带参数的函数时，参数的顺序、数量 要与声明时保持一致。

调用函数的时候传递的值 -> 实参

4、分支结构

1、流程控制

程序 = 数据结构 + 算法

任何程序都是由以下三种结构组成：

1、顺序结构

2、选择结构(分支结构)

根据“条件”，选择某一段代码去执行

3、循环结构

根据“条件”，选择某一段代码反复执行

2、分支结构

1、什么是分支结构

根据不同的条件，选择不同的代码去执行

if 结构：

当条件满足时，运行指定的语句，

当条件不满足时，则不运行这些语句

if...else 结构：

当条件满足时，运行指定的一组语句

当条件不满足时，运行另一组语句

2、if 结构

语法：

if(条件){

语句块;

}

条件：boolean类型的结果或表达式

比较运算符：

>,<,>=,<=,==,!=

var a=5;

var b=3;

a>b : true

a<b : false

a>=b: true

a<=b: false

a==b: false

a!=b: true

由比较运算符组成的表达式称之为 逻辑表达式。

注意：

1、if(条件){}

条件：必须是boolean类型

如果不是boolean类型的话，会产生隐式转换

以下情况会默认转换成 假:

if(0){}

if(0.0){}

if(""){}

if(null){}

if(undefined){}

if(NaN){}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if(1){} //真

if("我很帅吗"){}//真

2、if后的{}可以省略

如果省略{}的话，那么if只控制它下面的一条语句

3、if - else 结构

语法：

if(条件){

语句块1;

}else{

语句块2;

}

判断条件

如果条件为真，则执行语句块1的内容

如果条件为假，则执行语句块2的内容

4、多重if结构

1、问题

编写一个函数，让用户录入一个考试成绩，根据成绩值做以下判断：

成绩 == 100

买辆兰博基尼

成绩 >= 90

买辆法拉利

成绩 >= 80

买辆奥迪

成绩 >= 70

买辆奥拓

成绩 >= 60

买辆 永久牌 自行车

否则：

给俩大嘴巴子

2、语法

if(条件1){

语句块1;

}else if(条件2){

语句块2;

}else if(条件3){

语句块3;

}...else{

}

特点：

从上到下判断条件，满足哪个条件，则执行对应的语句块，执行完成后，就跳出该结构，即便满足其他条件，也不会判断。

练习：

计算 克莱托 指数

公式：体重(kg)/(身高(m)\*身高(m))

20-25 : 正常

20以下 : 偏瘦

25以上 : 偏胖

eg:

身高 1.7m

体重 60kg

结果为：20.761.....

正常

# 第三天

1、回顾

1、数据类型转换

1、什么是数据类型转换

不同类型的数据之间在做运算的时候，会发生的类型的转换。

2、数据类型转换分类

1、隐式转换

自动完成

string 类型参与运算时，其他数据类型都会转换成string类型

注意：如果string类型中的数据是数字类型，那么所做的-，\*，/，%全部都是数据运算。

2、显示转换

转换函数：

1、toString()

将任意类型的数据转换成string类型

2、parseInt()

将指定的数据转换成整数。如果碰见第一个非数字字符，则停止转换。

parseInt(53.5)->53

parseInt("75")->75

parseInt("75Hello")->75

parseInt("H75")->NaN

3、parseFloat()

将指定数据转换成小数

2、函数

1、什么是函数

函数：Function,又称为方法(Method),过程(Procedure)

是一段 预定义好，可以被反复使用的代码块。

window.prompt()

document.write()

2、定义函数

语法：

function 函数名(){}

3、调用函数

函数名();

<button onclick="函数名()">

<script>

函数名();

function 函数2(){

}

function 函数1(){

函数2();

}

function 函数3(){

函数3();

}

</script>

4、带参函数

1、声明

语法：

function 函数名(参数列表){

}

参数列表：用 , 隔开的若干参数名称

2、调用

函数名(参数值列表);

function print(n1,n2){

console.log(n1);

console.log(n2);

}

print(5,10);

结果：

5

10

print(10,5)

结果：

10

5

3、分支(选择)结构

1、程序的流程

1、顺序结构

自顶向下，逐条语句执行

2、分支结构

根据指定的条件，选择某段代码去执行

3、循环结构

根据指定的条件，反复的执行某段代码

2、分支结构

1、if 语句

语法：

if(条件){

语句块;

}

流程：

1、判断条件

2、条件为真，则执行语句块

3、条件为假，则不执行语句块

条件：boolean类型的值或表达式

注意：

1、如果条件为非boolean的值的话

以下情况当做false来处理

if(0)

if(0.0)

if(null)

if(undefined)

if("")

if(NaN)

2、if-else 语句

语义：如果...否则...

语法：

if(条件){

语句块1;

}else{

语句块2;

}

流程：

1、判断条件是否为真

2、如果为真，则执行语句块1

3、否则(条件为假)，则执行语句块2

3、多重if结构

if(条件1){

语句块1;

}else if(条件2){

语句块2;

}...else{

语句块n;

}

练习：

编写一个 “个人所得税计算器” 函数

该函数 接受一个参数值，来表示税前工资,在函数内，将工资所对应的 税额 计算出来并打印输出

个税计算梯度如下：

3500以下：免征

3500~5000部分：缴纳3%个税

5000~9000部分：缴纳10%个税

工资：7000元

3500 : 不交税

1500(超出3500，低于5000部分) : 3% -> 45

2000(超出5000，低于9000部分) :

10% -> 200

9000以上部分：缴纳20%个税

税前工资：35000

3500 : 免征

1500 : 45

4000 : 400

26000 :5200

35000 -> 5645税

2、循环结构

1、问题

1、在控制台上打印输出一遍 Hello

console.log("Hello");

2、在控制台上打印输出十遍 Hello

console.log("Hello");

console.log("Hello");

console.log("Hello");

console.log("Hello");

console.log("Hello");

console.log("Hello");

console.log("Hello");

console.log("Hello");

console.log("Hello");

console.log("Hello");

条件：

10遍

操作：

console.log("Hello");

3、在控制台上打印输出100遍 Hello

条件：

100遍

操作：

console.log("Hello");

4、将 Hello 全部改成 你好

条件：

100遍

操作：

将 Hell 改成 你好

5、在输出 你好 的基础上 增加

第1遍 你好

第2遍 你好

...

第100遍 你好

条件：

同上

操作：

增加 遍数 的打印

2、循环

1、什么是循环

循环：一遍又一遍，重复的执行相同或者相似的事情(代码)

2、生活中的循环

1、上学

从 5月30 日开始

到 9月30 日结束

2、运动员的比赛

从第一场开始

到 淘汰 结束

3、活着

从 出生开始

到 ... 结束

4、地球的公转自转

...

3、循环特点：

1、循环条件

制约执行循环的次数的东西

从什么时候开始

到什么时候结束

2、循环操作

又称为循环体

要执行的相同的或者相似的事情是什么

3、while 循环

1、语法

while(条件){

循环操作

}

流程：

1、判断 条件

2、如果条件为真，则执行循环操作

2.1 、执行完循环操作之后，再一次判断条件

2.2 、如果条件为真，再一次执行循环操作

...

3、如果条件为假，则不执行循环体

2、注意

一定要在循环体内，更新循环条件，否则，就是死循环

i=i+1;

i+=n;--> i=i+n;

i++; --> i=i+1;

i--; --> i=i-1;

练习：

1、打印 1-100之间所有的数字

2、打印 1-100之间所有的奇数

3、break

1、作用

用于循环中，终止循环的执行

2、练习

要求用户 循环的输入信息并打印 ，直到输入 0 则停止

条件：

输入0为止，否则，无限循环

操作：

1、输入

2、打印

4、for 循环

1、while 循环

打印 1-100之间所有的数字

var i=101; //1、声明循环变量

while(i<=100){//2、循环条件

//3、循环体

console.log(i);

//4、更新循环变量

i++;

}

2、for 语法

for(声明循环变量;循环条件;更新循环变量){

循环体

}

过程:

1、先计算表达式1的值

即声明循环变量

2、计算表达式2的值

即判断循环条件

如果结果为true，则继续向下，否则，退出循环

3、执行循环体

4、计算表达式3的值

即更新循环变量

5、计算表达式2的值

即判断循环条件

如果结果为true，则继续向下，否则，退出循环

3、使用场合

1、for

已经 明确 了循环次数的时候，优先使用 for 循环。

2、while

不确定循环次数时，相比之下，可以优先使用 while 循环

4、for 三个表达式的特殊用法

1、第一个表达式为空时

var i=1;

for(;i<100;i++){

console.log(i);

}

注意：可以省略第一个表达式，但是一定要在循环之外，将第一个表达式内容补齐。否则就是语法错误。

2、第二个表达式为空时

for(var i=1;;i++){

console.log(i);

}

省略第二个表达式时，默认为真-死循环

如果省略第二个表达式的话，一定要在 循环体内将条件补足。否则就是死循环

3、第三个表达式为空时

for(var i=1;i<100;){

console.log(i);

}

省略第三个表达式时，结果为死循环。如果要省略的话，那么就要在循环体内补足第三个表达式的值或给定退出条件

4、省略所有表达式

for(;;){

}

省略所有表达式-死循环。补足所有的内容即可。

5、表达式1以及表达式3的位置处，允许接收多个表达式。中间用,隔开即可。

3、数组

1、问题

保存班级中，所有同学的名字

var name1="张三";

var name2="李四";

var name3="王二麻子"

var name4="王五";

2、什么是数组

多个数据所组成的集合。

将多个数据保存在一个变量名中

在内存中，是按照线性顺序排列的(有序排列)。

下标：数组中，每个元素的序号。下标从0开始，到数组的元素个数-1结束

3、定义和初始化数组

语法：

var 数组名=[];//声明一个空数组

var 数组名=[值1,值2,值3];//声明一个包含3个元素的数组

var 数组名=new Array();//声明一个空数组

var 数组名=new Array(值1,值2,值3);//声明一个包含3个元素的数组

4、访问数组

1、访问数组元素的方式

index:下标，索引 ...

数组名称[下标];

2、存 - 设置数组元素的值

数组名称[下标]=值;

注意：如果下标所对应的元素已经存在的话，那么就会实现覆盖

3、取值

数组名称[下标];

注意：访问数组中不存在的元素，结果为 undefined

4、获取数组的长度

数组长度：即数组中，元素的个数

语法：

数组名称.length;

可以通过 length 属性，循环遍历数组中的每一个元素

# 第四天

1、DOM概述

1、DHTML介绍

D : Dynamic 动态的

功能：当网页加载到客户端之后依然有能力改变它上面的内容

DHTML=HTML+CSS+Javascript

包含：

1、动态改变页面元素

2、与用户进行交互

3、DHTML对象模型中主要包含

BOM：浏览器对象模型

DOM：文档对象模型

2、DHTML-BOM与DOM

1、BOM

作用：专门用来访问和操作浏览器窗口。

BOM 没有标准和规范，单支持性较好。

2、DOM

作用：专门用来访问和操作网页文档内容

3、DOM概述

标准被分为3个部分:

1、核心 DOM

针对任意一个结构化文档所提出的标准模型(属性和方法)

结构化文档：

1、html ：用于显示数据

2、xml : 用于记录数据

2、XML DOM

针对XML文档所提供的属性和方法

3、HTML DOM

针对HTML文档所提供的属性和方法

1、DOM发展

1、DOM1：定义了DOM的最底层的内容

2、DOM2：增加 交互模块

DOM2 Core：DOM1底层基础上增加了新内容

DOM2 Style:增加了操作HTML样式的属性和方法

DOM2 Tranversal And Range：增加了遍历页面元素的属性和方法

DOM2 Event 标准化事件

3、DOM3

增加新方法和属性

4、DOM树

又称为 节点树

将HTML页面 解析称为一个 倒挂的树状结构

文档中的元素，属性，文本，注释都是树状结构中的一个节点

整个文档 是一个 文档节点(document)

每个HTML元素 是一个 元素节点(Element Node)

包含在HTMl元素中的文本 是一个 文本节点(Text Node)

包含在HTML元素中的属性 是一个 属性节点(Attribute Node)

页面中的注释属于 注释节点(Comment Node)

5、document 对象

该对象由JS解释器负责创建，不需要声明，直接就能使用

2、通过 HTML 选取元素

1、document.getElementById()

返回值：指定ID的元素对象

2、node.getElementsByTagName()

node : 元素节点对象

作用：根据"标签名"称获取一组对象

返回值：动态数组

3、node.getElementsByClassName()

node : 元素节点对象

作用 ：根据 class 属性值 获取一组对象

返回值：动态数组

4、通过节点关系获取页面元素

1、节点关系

1、父节点(Parent Node)

2、子节点(Child Node)

3、兄弟节点(Sibling)

2、属性

1、

1、parentNode

获取当前元素的父节点对象

2、childNodes

获取当前元素的所有子节点对象

返回：数组

3、children

获取当前元素的所有子元素节点对象

4、firstChild

获取当前元素的第一个子节点对象

5、lastChild

...

6、previousSibling

作用：获取上一个兄弟节点

7、nextSibling

作用：获取下一个兄弟节点

2、获取 元素节点对象

1、firstElementChild

第一个元素子节点

2、lastElementChild

最后一个元素子节点

3、previousElementSibling

获取上一个 兄弟 元素节点

4、nextElementSibling

获取下一个 兄弟 元素节点

3、读取和修改 节点内容

1、元素的内容

1、HTML内容

属性：innerHTML

作用：获取 或 设置元素节点中的HTML内容

2、文本内容

属性：

innerText(火狐不支持)

textContent

作用：获取 或 设置元素节点中的 文本内容

2、元素的属性

1、属性集合

元素节点属性:attributes

返回一个属性集合,类型为 NamedNodeMap 是一个 类数组对象

2、获取 属性值

1、elem.attributes[下标].value

2、elem.attributes["属性名"].value;

3、element.getAttributeNode("属性名").value;

4、element.getAttribute("属性名");

3、修改 属性值

1、元素对象.属性名=value;

2、element.setAttribute(name,value);(IE8不支持该方法)

3、element.setAttributeNode(attrNode);

//前提 ：先创建 属性节点对象

3、元素的样式

1、CSS样式三种使用方式

1、内联

<div style="">

2、内部样式表

<style></style>

3、外部样式表

<link>

2、常用访问样式的方式

常用：访问元素的 内联样式

属性：style 属性

style.样式属性名称

样式属性名称组成：

CSS:background-color

JS :backgroundColor

CSS:border-left-color

JS :borderLeftColor

3、获取 计算的(Computed)样式

变相获取 内部和外部样式表中的值

DOM：

document.defaultView.getComputedStyle(元素对象).属性名 ;

IE：

元素对象.currentStyle.属性名

注意：该方法只能获取样式属性值，不能设置。设置样式全部采用style属性进行设置

4、修改 样式表中的样式值

1、先获取样式表对象

2、获取 样式表对象中所有样式的集合

3、在集合中，找到要修改的样式规则

4、获取或设置指定的样式值