注意：

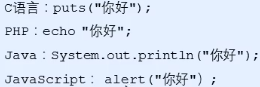
1. 红色为知识大部分。
2. 【】为知识分块。
3. ⚪黑点为【】内知识点。
4. 数字标号为⚪黑点分点。
5. 黄色为显眼重点。
6. 蓝色语法截图为重点语法。
7. 灰色表格截图为语法详解。
8. 黑色背景为编程示例。
9. 具体案例在相关html文件中。

***基础部分***

【基础】

* 编程语言：主动，有逻辑能力和行为能力，如if else，while，for等指令。

标记语言（html）：被动，被读取，不向计算机发送指令，常用于格式化和链接。

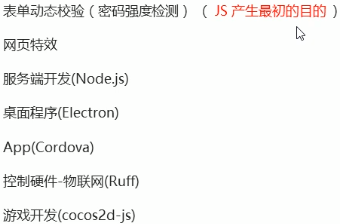


* JavaScript是运行在客户端的脚本语言。

脚本语言：不需编译，运行时由js解释器（js引擎）逐行解释和执行。

现在可以基于node.js技术进行服务器端编程。

* js作用：业务逻辑、页面控制



* 浏览器组成：

1. 渲染引擎：解析html和css，俗称内核，如chrome的blink、webkit。
2. js引擎（js解释器）：读取网页的JavaScript代码处理后运行（浏览器本身不执行JavaScript代码），如chrome的v8。

* js组成：

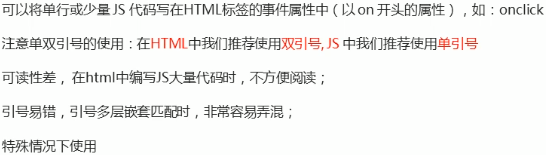
1. ECMAscript：JavaScript语法



1. dom（document object model）：页面文档对象模型。处理可扩展标记语言的标准编程接口，通过dom提供的接口可对页面上元素进行操作（大小、位置、颜色等）。
2. bom（browser object model）：浏览器对象模型。提供独立于内容、可与浏览器窗口进行互动的对象结构，通过bom可操作浏览器窗口（弹出框、控制浏览器跳转、获取分辨率等）。

* js书写位置

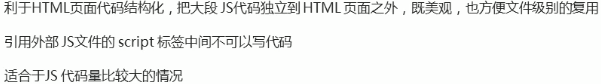
1. 行内



1. 内嵌



1. 外部



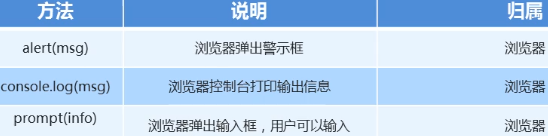
* js注释

单行：// 快捷键ctrl+/

多行：/\*\*/ 快捷键ctrl+alt+a

修改快捷键…

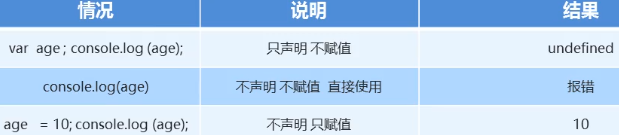
* js输入输出语句



* 变量声明var



无var声明时当全局变量。



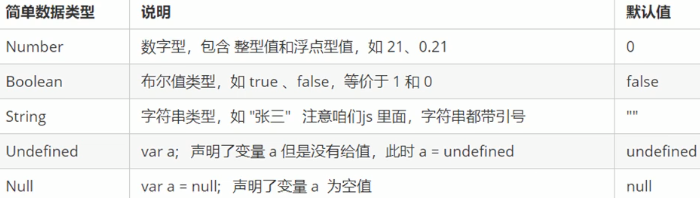
命名规范;

1. 变量名由字母、数字、下划线、$组成；
2. 不能以数字开头；
3. 驼峰命名法：首字母小写，后面单词首字母大写。

* js数据类型

JavaScript是一种弱类型/动态语言，不用提前声明变量类型，运行时由js引擎根据=右侧确定数据类型，同一变量可用于不同数据类型。

1. 简单数据类型



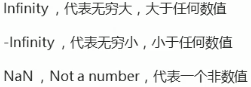
1. 数字型number：整数、小数（浮点数）

二进制、八进制、十进制、十六进制

js中数值的最大值和最小值：



数字型三个特殊值Infinity、-Infinity、NaN：



IsNaN(x)：x为数字返回false，x不是数字返回true。

1. 字符串型string

‘ ’或” ”，推荐js用单引号，html用双引号；

可单引号嵌套双引号，或双引号嵌套单引号。

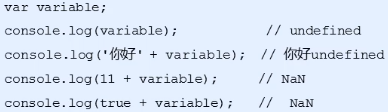
转义符：



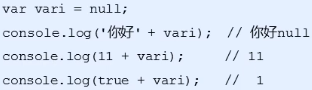
str.length：获取字符长度，空格为1字符。

字符串拼接：新字符串 = 字符串 + 任何类型（拼接前将其转为字符串）。字符串 + 变量名（不转为字符串）。

1. 布尔型Boolean
2. undefined：已声明未赋值的变量，或赋值为undefined。



1. null：空值



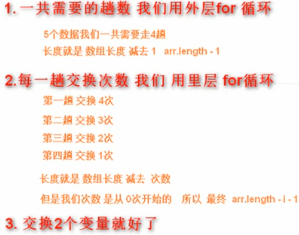
1. typeof：获得数据类型。typeof+空格+变量名。
2. 复杂数据类型

* 未看32-94，96-111
* 冒泡排序：

目的：将数据从小到大或从大到小排列；

原理：一次比较两个相邻数值，如果顺序不对则交换位置。

方法：



* 未看114-135
* 全局作用域：整个script标签，或一个单独js文件。

局部作用域：函数内部。

块级作用域：{ } 内，如if { }，for { }。es6才有。

作用域链：内部函数访问外部函数的变量以链式查找方式决定取值（就近原则）。

* 未看137
* js执行分两步：

1. 预解析：将所有var和function提升到当前作用域最前面。
2. 变量预解析（变量提升）：不提升赋值操作。
3. 函数预解析（函数提升）：不调用函数。
4. 代码执行：顺序执行。

* 对象是一组无序的相关属性（特征）和方法（行为）的集合。
* 创建对象

1. 用字面量创建对象：用 { } 包围。

// var obj = {};//创建空对象

        var obj = {

            uname: '张三',

            age: 18,

            say: function () {

                console.log('Hi~');

            }

        }

1. 对象里的属性和方法采用键值对形式。属性名：值。
2. 多个属性或方法用 , 隔开。
3. 方法冒号后跟一个匿名函数。

console.log(obj.uname);

console.log(obj['age']]);

1. 调用对象的属性：对象名 . 属性名、对象名 [ ‘属性名 ’ ]。



1. 调用对象的方法：对象名 . 方法名。
2. 变量和属性：

相同点：都用来存储数据；

不同点：变量：单独声明并赋值，使用时直接写变量名；

属性：属于对象里，不需要声明，使用时（对象 . 属性）。

1. 函数和方法：

相同点：都是实现某一功能；

不同点：函数：单独声明（function 函数名( ) { }）和调用（函数名()）；

方法：属于对象里，声明（函数名: function ( ) { }）使用时（对象 . 方法）。

1. 用new object创建对象

var obj = new Object();//创建空对象

        obj.uname = '张三';

        obj.age = 18;

        obj.say = function () {

            console.log('Hi~');

        }

1. 采用等号=赋值的形式添加属性和方法；
2. 每个属性和方法用 ; 分隔；
3. 调用：同1.。
4. 用构造函数（class）创建对象

构造函数：多个对象里相同的属性和方法抽象出来封装到函数里。

声明：

function 构造函数名( uname,age){

this.属性=值; 即this.name = uname;

this.age = age;

this.方法=function(){} 即this.say = function (sang) {

                 console.log(sang);

             }

}

调用：

var fn1 = new 构造函数名('张三', 18);

var fn2 = new 构造函数名('李四', 19);

返回的是对象。

1. 构造函数名首字母大写；
2. 构造函数不需要return即可返回结果；
3. 一次new调用创建一个对象。
4. 用构造函数创建对象的过程称为对象的实例化。

* 构造函数new的工作流程：

1. 内存中创建一个空对象；
2. this指向这个空对象；
3. 执行构造函数，给这个空对象添加属性和方法；
4. 返回这个对象（所以不需要return）。

* for…in语句：用于对数组或对象的属性进行循环操作。

// for(变量 in 对象){}

for(var k in obj){

            console.log(k); //变量k，输出的是属性名

            console.log(obj[k]); //输出的是属性值

        }

* JavaScript对象分为：自定义对象、内置对象（Math、Date、Array、String）、浏览器对象，前两种属于js基础，是ECMAScript，第三个是js独有，由JS API讲解。
* 内置对象如何使用，可查询MDN、W3C



* Math内置对象：它不是一个构造器（所以不需要new调用），它的所有属性和方法都是静态的（直接使用即可）。
* 未看160-169
* 创建数组：

// 1.字面量

        var arr = [1, 2, 3];

        console.log(arr[0]);

        // 2.用new Array()

        var arr1 = new Array();   //创建一个空数组

        var arr1 = new Array(2);   //创建一个长度为2的空数组

        var arr1 = new Array(2, 3);   //创建一个元素为2，3的数组

* 未看171
* 添加/删除数组元素：



arr.push(1,'lalala');   //添加数组元素

console.log(arr.push(3));   //返回值是新数组长度

unshift用法类似push。

arr.pop();  //删除数组末尾一个元素，返回值是删除的那个元素

shift用法类似pop。

* 未看174-187
* 简单数据类型/基本数据类型/值类型：变量中存储的是值本身。如string、number、Boolean、undefined、null（返回值是空对象object）

复杂数据类型/引用类型：存储的是地址。用new创建（系统对象、自定义对象），如Object、Array、Date。

* 堆栈空间分配

栈（操作系统）：操作系统自动分配释放存放函数的参数值、局部变量的值等，存放简单数据类型（直接开辟空间存值）。

堆（操作系统）：存储复杂类型（对象）（栈里存储的16进制地址指向堆里存储的内容），由程序员分配释放，程序员若不释放则由垃圾回收机制回收。

值类型实参作为参数传递给形参时，其实是把变量在栈空间的值复制给形参，在方法内部修改形参不影响外部实参。

引用类型实参传递给形参，相当于复制了栈空间的地址，所以操作的是同一个对象。

* webAPIs：W3C标准，js独有，主要是DOM、BOM，交互效果。

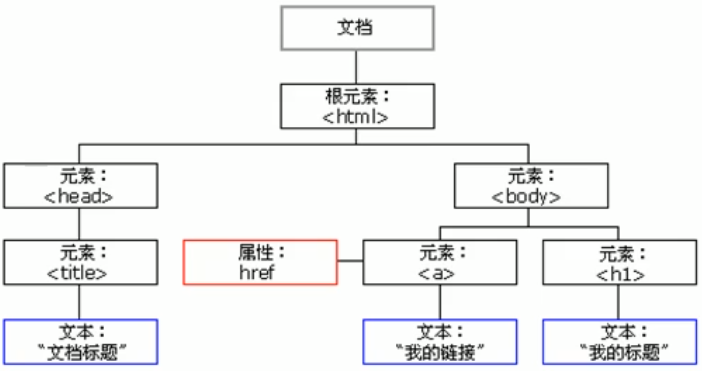
API（application programing interface应用程序编程接口）：是一些预先定义的函数，提供应用程序或开发人员基于某软件或硬件得以访问一组例程的能力，无需访问源码，无需了解内部工作机制。

webAPI：是浏览器提供的一套操作浏览器功能和页面元素的API（BOM、DOM）。



* DOM（document object model文档对象模型）：处理可扩展标记语言（html、xml）的标准编程接口。

1. DOM树



一个页面即一个文档document；

页面中标签即元素element；

网页中内容即节点node（标签、文本、属性、注释）；

以上内容均看作对象。

1. 获取元素
2. 根据ID获取

var x=document.getElementById('id');    //返回的是元素对象

console.dir(id);    //直接打印返回的元素对象，查看其属性和方法

1. 根据标签名获取

document.getElementsByTagName('div');   //返回元素对象的集合，以伪数组形式存储

element.getElementsByTagName('div');   //获得父元素内标签（父元素须为单个对象，获取时不包含父元素本身）

1. 根据类名获取

document.getElementsByClassName('class');

document.querySelector('选择器');   //根据指定选择器返回第一个元素对象（选择器需加符号）

document.querySelectorAll('选择器');

        document.body;  //返回body

        document.documentElement;   //返回html

1. 事件三要素：事件源、事件类型、事件处理程序

对象. 事件=function



1. 改变元素内容

element.innerText：改变内容时不识别标签；获得内容时标签除外的内容，同时去除空格和换行。

element.innerHTML：改变内容时能识别标签；获得内容时包括标签，同时保留空格和换行。

* 未看206-216
* 排他思想：清除样式，再设置自己
* 未看218-226
* 自定义属性：保存并使用数据，保存到页面而不需数据库。

自定义属性获取：getAttribute(‘属性’);

H5规范：自定义属性名以data-开头。

设置自定义属性：element.setAttribute(‘data-a’,111);

获取自定义属性值：element.getAttribute(‘data-a’);

只能获取data-开头的：

element.dataset.index; //如果index为多个-链接词，index采取驼峰命名法

element.dataset[‘index’];

* 节点操作：用节点层级关系获取元素

节点至少有3个基本属性：节点类型（nodetype）、节点名称（nodename）、节点值（nodevalue）。

元素节点nodetype为1；

属性节点nodetype为2；

文本节点nodetype为3（包含文字、空格、换行等）。

* 未看229-266
* BOM（浏览器对象模型），提供独立于内容而与浏览器窗口交互的对象，核心对象是window。

window是浏览器的顶级对象，是一个全局对象，定义在全局作用域的变量、函数都是window对象的属性和方法，调用时可省略window.。

* 未看269-277
* this的指向：运行时确定。

1. 全局作用域或普通函数：this指向window（定时器里指向window）；
2. 方法调用：this指向调用对象；
3. 构造函数中：this指向构造函数的实例。

* js是单线程。

同步：顺序执行。

异步：一边……一边……

* js执行机制

同步任务在主线程上执行，形成一个执行栈，先执行；

异步任务通过回调函数实现，包括1.普通事件，如click、resize；2.资源加载，如load、error；3.定时器，如setinterval、settimeout。异步任务相关回调函数添加到任务队列（消息队列）中，等待被列入执行栈。

事件循环（event loop）。

* 未看282-358

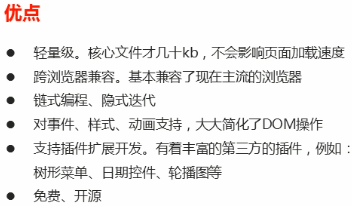
【jQuery】封装DOM

* 库library，是封装好的特定的集合（方法和函数）。



都是对原生js的封装，内部都是用js实现的。

jQuery优点：



下载：

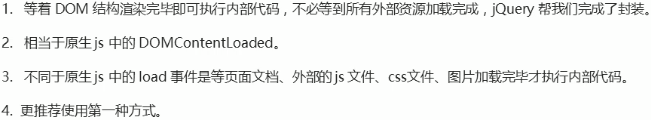


（production压缩版本，development未压缩版本）创建文件，复制内容到此：



* jQuery入口函数





* jQuery顶级对象$，相当于window，将元素用$包装成jQuery就可以调用jQuery方法。

$是jQuery的别称，代码中使用二者均可。

* 用原生js获取的对象是DOM对象。

jquery方法获取的对象是jQuery对象。本质是以$包装DOM对象产生的对象，以伪数组形式存储。

jQuery对象只能使用jQuery方法，DOM对象只能使用原生js方法。

1. DOM对象转为jQuery对象：$(‘DOM对象’); / $(DOM对象);（已用原生js获取DOM对象时）。
2. jQuery对象转为DOM对象：$(‘jQuery对象’)[index]; / $(‘jQuery对象’).get(index);

* 未看367-453