注意：

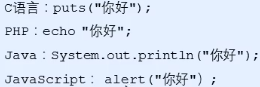
1. 红色为知识大部分。
2. 【】为知识分块。
3. ⚪黑点为【】内知识点。
4. 数字标号为⚪黑点分点。
5. 黄色为显眼重点。
6. 蓝色语法截图为重点语法。
7. 灰色表格截图为语法详解。
8. 黑色背景为编程示例。
9. 具体案例在相关html文件中。

***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。

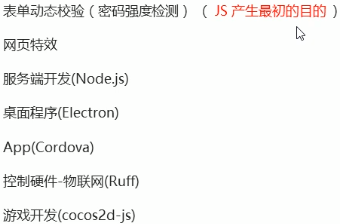


* JavaScript是运行在客户端的脚本语言。

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：

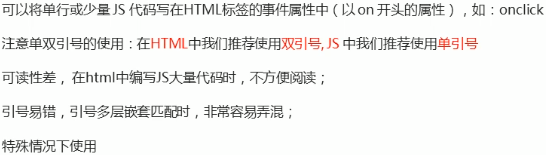
1. ECMAscript：JavaScript语法



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

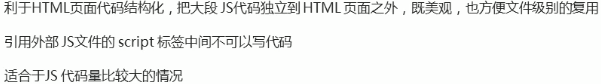
1. 行内



1. 内嵌



1. 外部



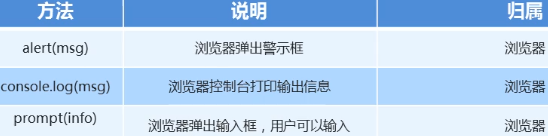
* js注释

单行：// 快捷键ctrl+/

多行：/\*\*/ 快捷键ctrl+alt+a

修改快捷键…

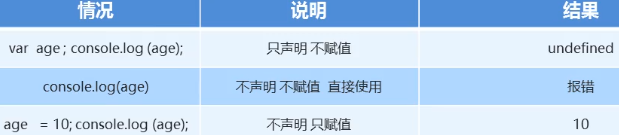
* js输入输出语句



* 变量声明var



无var声明时当全局变量。



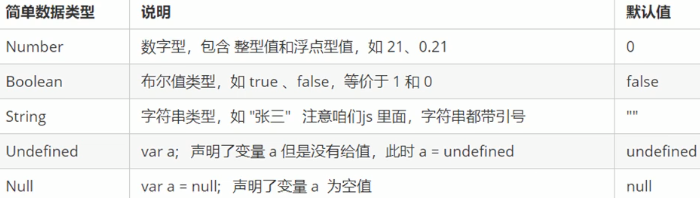
命名规范;

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



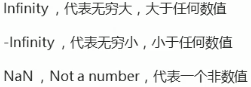
1. 数字型number：整数、小数（浮点数）

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。

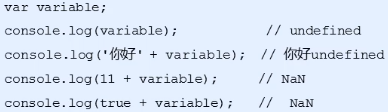
转义符：



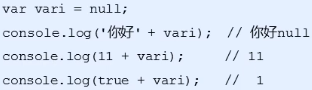
str.length：获取字符长度，空格为1字符。

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



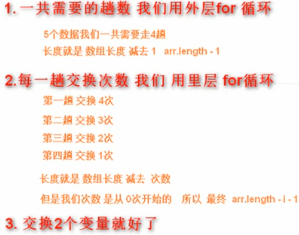
1. typeof：获得数据类型。typeof+空格+变量名。
2. 复杂数据类型

* 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. 创建一个空对象；
2. this指向这个空对象；

* 153 2.00