HTML：

HTML 是用来描述网页的一种语言。

1、Tag: Content 装内容装数据

2、CSS: Display 渲染,显示

3、JavaScript: Behavior 交互,行为

HTML 指的是超文本标记语言: Hyper Text Markup Language

HTML 不是一种编程语言，而是一种标记语言

标记语言是一套标记标签 (markup tag)

HTML 使用标记标签来描述网页

HTML 文档包含了HTML 标签及文本 内容

HTML文档也叫做web 页面

CSS：

内部样式表：多用于首页。 加载快

外部样式表：用于其他页面。 所有页面可以共享同一个样式，便于风格统一；使内容和显示分离

内嵌（行内）样式表：不常用

CSS选择样式三原则：

1、就近原则：

2、具体性原则： #id选择器 > .class选择器 > 标签选择器 > 通配符选择器

3、重要性原则： !important

Box Model （盒模型） 由内到外：

内容 - padding - border - background-image - background-color - margin

CSS 选择器：

通配符选择器： \*

标签选择器：

类选择器： .

id选择器： #id

并列选择器： ,

后代选择器： 空格(所有后代)、>(儿子后代)

兄弟选择器： ~ (所有兄弟)

相邻兄弟选择器： + (必须是相邻兄弟)

边框和轮廓

border 设置对象边框的特性

border-collapse : collapse 设置表格的边框合并为一个单一的边框

border-color 设置或检索对象的边框颜色

outline 设置或检索对象外的线条轮廓。

字体

font-size 设置字体大小

font-family 规定文本的字体

font-style 规定文本的字体样式

font-weight 规定字体的粗细

文本

color 指定文本的颜色

line-height 设置行高

text-align 规定文本的水平对齐方式

text-decoration 规定添加到文本的装饰效果，：none取消下划线

内外边距

padding 简写属性在一个声明中设置所有填充属性。该属性可以有1到4个值

margin 在一个声明中设置所有外边距属性，该属性可以有1到4个值

背景

div是不支持background-color 的，background-color是table等的属性，你要用background：#（这里是颜色代码）；这样来写

background 复合属性，设置对象的背景特性

background-color 设置或检索对象的背景颜色 background-image 设置或检索对象的背景图像

display属性

display：none 此元素不会被显示。

display：block 此元素将显示为块级元素，此元素前后会带有换行符。

display：inline 此元素会被显示为内联元素，元素前后没有换行符。

hidden 元素是不可见的。

列表

list-style 在一个声明中设置所有的列表属性 list-style-position 设置列表项标记的放置位置

CSS 定位：

position：static 正常文档流

position：relative 相对定位（相对于元素原来的位置，没有脱离文档流，对周围元素没有影响）

position：absolute 绝对定位（相对于父元素来设定位置，脱离了文档流，对周围元素有影响）

position：fixed 固定定位（相对于浏览器窗口来设定位置，脱离了文档流）

CSS hack technology （CSS黑科技）

z-index索引 当脱离文档流时 元素冲突 设置z-index 值越大就渲染在越上方

在有浮动的情况下 ，为了让父级元素计算出它的高度的方法：

1、clear:both 清除浮动

2、在父元素的样式中加入 overflow：auto

CSS 设置字符集

@charset : utf-8

CSS 设置字体：

@font-face {

font-family： ‘字体名称’

src：url(字体资源路径)

}

JavaScript：

JavaScript : JS - Behavior

ECMAScript - 语法规范 - ES5

BOM - 浏览器对象模型 - window

DOM - 文档对象模型 - document

JavaScript 数据类型 （JavaScript 有隐式的类型转换）：

简单数据类型： number、string、boolean、null、undefined

复杂数据类型： object

可以使用 typeof() 查看数据类型

JavaScript 运算符：

算术运算符：+ 、-、 \*、 /、 % 、\*\*（ES7版本）

赋值运算符：= 、+=、 -=、 \*=、 /=、%=

比较（关系）运算符： >、 >=、 <、 <=、 ==、 != 、===（不带隐式类型转换的等于，全等于）、!==（严格不等于）

逻辑运算符： &&（短路与）、 ||（短路或）、 ！

自增自减运算符： ++ 、--

位运算、成员运算

编程范式（理念）：

面向对象编程

函数式编程

// 全局函数

isNaN() 是不是 不是一个数

parseInt() 取整，如果取不出，返回NaN

parseFloat() 取小数，如果取不出，返回NaN

isFinite() 是不是有限的数

eval() - evaluate - XSS（跨站脚本攻击）

encodeURIcomponent() 编成 百分号 编码

decodeURIcomponent() 从百分号编码 解码

获取引发事件的对象元素：

document.getElementById(#id)

this // this 当前对象元素

// evt 表示当前事件，evt.target 表示引发当前事件的事件源

// 需要在函数参数中 传入evt 对象

evt = evt || window.event; // 兼容低版本的IE

var target = evt.target || evt.srcElement;

Infinity 无穷大

-Infinity 负无穷大

NaN 不是一个数

在定义函数时，如果没有传入参数，那么函数中有一个隐藏的 arguments 对象，它代表了参数的列表，在调用时，可以传入任意多个参数。

function foo（）{

for （var i = 0；i < arguments.length；i++）{

alert（arguments[i]）；

}

}

自定义的兼容函数：

// 用cacheId 来缓存已经获取过的元素

var cacheId = {};

/\*\*

\* 根据id获取元素

\* @param {Object} id 元素的id

\*/

function $(id){

if (!cacheId[id]){

cacheId[id] = document.getElementById(id);

}

return cacheId[id];

}

/\*\*

\* 给element 元素绑定事件

\* @param {Object} element 元素

\* @param {Object} event 要绑定的事件

\* @param {Object} fn 绑定的函数

\*/

function bindEvent(element, event, fn){

if (element.addEventListener){

element.addEventListener(event, fn);

}else{

element.attachEvent('on' + event, fn);

}

}

/\*\*

\* 给element 元素取消绑定事件

\* @param {Object} element 元素

\* @param {Object} event 要取消的事件

\* @param {Object} fn 要取消的函数

\*/

function unbindEvent(element, event, fn){

if (element.removeEventListener){

element.removeEventListener(event, fn);

}else{

element.detachEvent('on' + event, fn);

}

}

/\*\*

\* 获取元素的只读的样式，读取元素的样式需要通过 getComputeStype() 来获取，IE9 以下的版本使用这个 currentStyle

\* @param {Object} element 元素

\*/

function getStyle(element){

return document.defaultView.getComputedStyle ? document.defaultView.getComputedStyle(element) : element.currentStyle;

}

/\*\*

\* 给事件绑定处理默认行为和冒泡行为的函数，兼容了IE11以下的版本

\* @param {Object} ev

\*/

function handleEvent(ev){

ev = ev || window.event;

ev.preventDefault = ev.preventDefault ||

function (){

this.returnValue = false;

}

ev.stopPropagation = ev.stopPropagation ||

function (){

this.cancelBubble = true;

}

return ev;

}

querySelector() ： 根据指定的选择器查找单个元素

querySelectorAll() ： 根据指定的选择器查找所有元素

window对象：

document 文档

location 地址栏 - href / reload / replace

history 历史记录 - go / forward / back

navigator 浏览器

screen 操作系统窗口 - availWidth/ availHeight / width / height

alert / confirm / prompt

open / close

setTimeout / setInterval / clearTimeout / clearInterval

document 获取元素的方式：

getElementById

getElementsByTagName

getElementsByClassName

querySelector

querySelectorAll

修改节点的内容和属性

textContent

innerHTML

nodeValue

访问属性：

访问成员运算符 .

setAttribute / getAttribute / removeAttribute

创建新节点

createElement

追加新节点

appendChild

insertBefore

移除子节点

removeChild

通过自己节点访问父节点

parentNode

通过自己节点访问所有子节点

children

firstChild

lastChild

通过自己节点访问兄弟节点

nextSibling 前面的兄弟

prevSibling 后面的兄弟

JQuery：

jQuery的$函数的作用:

1、$(function() {}) $() 函数传入的是一个函数，表示该函数是在页面加载完成之后要执行的回调函数

2、$(selector) $() 函数传入的是一个选择器，表示通过选择器获得对应的元素，并处理成 jQuery对象，jQuery对象的本质是一个数组，如果要将jQuery对象转成原生js对象，使用下标或者get() 运算

3、$(element) $() 函数传入的是一个原生js对象，表示将其转成jQuery对象

4、$(标签) $() 函数传入的是一个标签，表示创建对应的元素并处理成jQuery对象

注意：当jQuery库和其他的库中的 $函数 有冲突，可以使用以下方法解决：

jQuery.noConflict() 将jQuery库中的 $函数 不起作用，但是jQuery库可以使用jQuery来替代 $，其他库可以使用 $函数

查找元素

选择器

\* / element / #id / .class / selector1, selector2

ancestor descendant / parent>child / previous+next / previous~siblings

筛选器

基本筛选器：:not(selector) / :first / :last / :even / :odd / :eq(index) / :gt(index) / :lt(index) / :animated / :focus

内容筛选器：:contains('…') / :empty / :parent / :has(selector)

可见性筛选器：:hidden / :visible

子节点筛选器：:nth-child(expr) / :first-child / :last-child / :only-child

属性筛选器：[attribute] / [attribute='value'] / [attribute!='value'] / [attribute^='value'] / [attribute$='value'] / [attribute|='value'] / [attribute~='value']

表单： :input / :text / :password / :radio / :checkbox / :submit / :image / :reset / :button / :file / :selected / :enabled / :disabled / :checked

执行操作

内容操作

获取/修改内容：html() / text() / replaceWith() / remove()

获取/设置元素：before() / after() / prepend() / append() / remove() / clone() / unwrap() / detach() / empty() / add()

获取/修改属性：attr() / removeAttr() / addClass() / removeClass() / css()

获取/设置表单值：val()

查找操作

查找方法：find() / parent() / children() / siblings() / next() / nextAll() / prev() / prevAll()

筛选器：filter() / not() / has() / is() / contains()

索引编号：eq()

尺寸和位置

尺寸相关：height() / width() / innerHeight() / innerWidth() / outerWidth() / outerHeight()

位置相关：offset() / position() / scrollLeft() / scrollTop()

特效和动画

基本动画：show() / hide() / toggle()

消失出现：fadeIn() / fadeOut() / fadeTo() / fadeToggle()

滑动效果：slideDown() / slideUp() / slideToggle()

自定义：delay() / stop() / animate()

事件

文档加载：ready() / load()

用户交互：on() / off()

回调函数

evt参数代表的是事件对象 - 绑定了和事件相关的所有信息

如果事件回调函数中要用到和事件相关的属性和方法就最好指定evt参数

target / clientX / button / keyCode

preventDefault() / stopPropagation()

不管函数是否指定了evt参数当事件发生回调该函数时都会传入该参数

调构造函数 前面加new

数组里删元素要从后往前循环

AJAX - Asynchronous JavaScript and XML

让浏览器向服务器发出异步请求(非阻塞式请求)，等服务器返回了响应数据(JSON/XML)后就可以对页面进行局部刷新，这样就可以在不中断用户体验的前提下刷新页面数据

$.ajax({

url:

type:

data:

datatype:

success:

error:

});