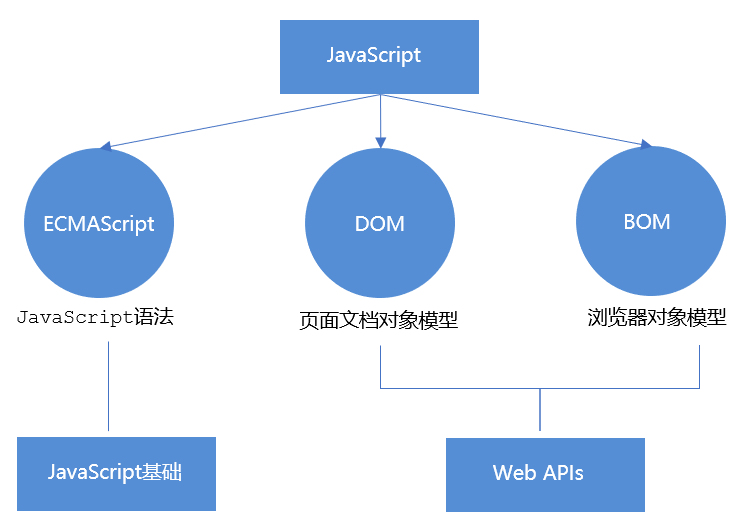
JavaScript核心Web APIs

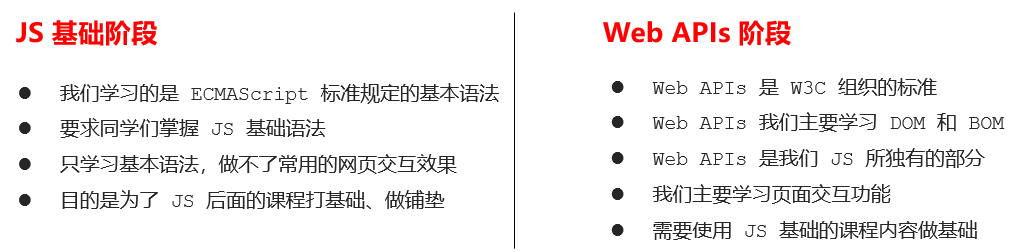
1.Web APIs简介

1.1Web APIs 和 JS 基础关联性

1、JS组成



2、JS 基础阶段以及 Web APIs 阶段



JS 基础学习 ECMAScript 基础语法为后面作铺垫， Web APIs 是 JS 的应用，大量使用 JS 基础语法做交互效果

1.2API 和 Web API

1、API

API（Application Programming Interface,应用程序编程接口）是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节

简单理解： API 是给程序员提供的一种工具，以便能更轻松的实现想要完成的功能

2、Web API

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

2.DOM

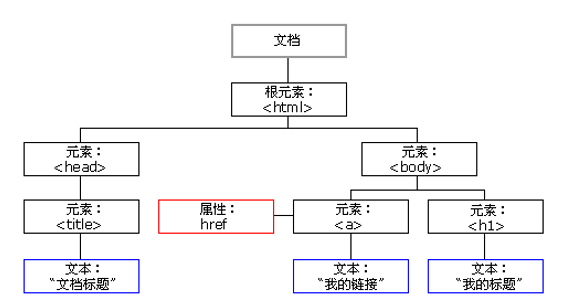
2.1DOM简介

2.1.1什么是DOM

文档对象模型（Document Object Model，简称 DOM），是 W3C 组织推荐的处理可扩展标记语言（HTML或者XML）的标准编程接口

W3C已经定义了一系列的DOM接口，通过这些DOM接口可以改变网页的内容、结构和样式

2.1.2DOM树



文档：一个页面就是一个文档，DOM中使用 document 表示

元素：页面中的所有标签都是元素，DOM中使用 element 表示

节点：网页中的所有内容都是节点（标签、属性、文本、注释等），DOM 中使用node表示

2.2获取元素

获取页面中的元素可以使用以下几种方式:

1）根据 ID 获取

2）根据标签名获取

3）通过 HTML5 新增的方法获取

4）特殊元素获取

2.2.1根据ID获取

使用 getElementById() 方法可以获取带有 ID 的元素对象



使用 console.dir() 可以打印我们获取的元素对象，更好的查看对象里面的属性和方法

2.2.2根据标签名获取

使用 getElementsByTagName() 方法可以返回带有指定标签名的对象的集合



注意：

1、因为得到的是一个对象的集合，所以我们想要操作里面的元素就需要遍历。

2、得到元素对象是动态的

3、如果获取不到元素,则返回为空的伪数组(因为获取不到对象)

还可以获取某个元素(父元素)内部所有指定标签名的子元素



注意：父元素必须是单个对象(必须指明是哪一个元素对象)获取的时候不包括父元素自己

2.2.3通过 HTML5 新增的方法获取







注意：

querySelector和querySelectorAll里面的选择器需要加符号

比如:document.querySelector('#nav');

2.2.4获取特殊元素

获取body元素



获取html元素



2.3事件基础

2.3.1事件概述

JavaScript 使我们有能力创建动态页面，而事件是可以被 JavaScript 侦测到的行为

简单理解： 触发--- 响应机制

网页中的每个元素都可以产生某些可以触发 JavaScript 的事件，例如，我们可以在用户点击某按钮时产生一个 事件，然后去执行某些操作

2.3.2事件三要素

1. 事件源 （谁）

2. 事件类型 （什么事件）

3. 事件处理程序 （做啥）

2.3.3执行事件的步骤

1. 获取事件源

2. 注册事件（绑定事件）

3. 添加事件处理程序（采取函数赋值形式）

2.3.4常见的鼠标事件



2.4操作元素

JavaScript 的 DOM 操作可以改变网页内容、结构和样式，我们可以利用 DOM 操作元素来改变元素里面的内容 、属性等。注意以下都是属性

2.4.1改变元素内容

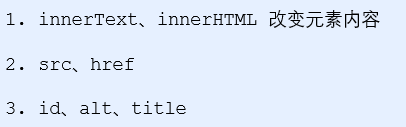


从起始位置到终止位置的内容, 但它去除 html 标签， 同时空格和换行也会去掉



起始位置到终止位置的全部内容，包括 html 标签，同时保留空格和换行

2.4.2常用元素的属性操作



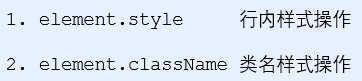
2.4.3表单元素的属性操作

利用 DOM 可以操作如下表单元素的属性：



2.4.4样式属性操作

我们可以通过 JS 修改元素的大小、颜色、位置等样式



注意：

1、JS 修改 style 样式操作，产生的是行内样式，CSS 权重比较高

2、如果样式修改较多，可以采取操作类名方式更改元素样式

3、class因为是个保留字，因此使用className来操作元素类名属性

4、className 会直接更改元素的类名，会覆盖原先的类名

2.4.5排他思想

如果有同一组元素，我们想要某一个元素实现某种样式， 需要用到循环的排他思想算法：

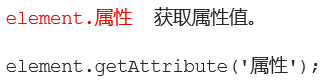
1. 所有元素全部清除样式（干掉其他人）

2. 给当前元素设置样式 （留下我自己）

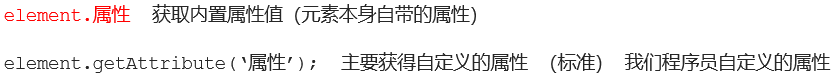
3. 注意顺序不能颠倒，首先干掉其他人，再设置自己

2.4.6自定义属性的操作

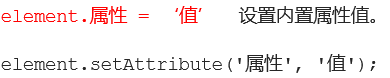
1. 获取属性值



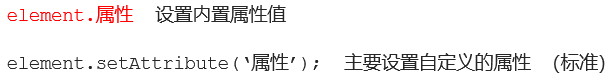
区别：



2、设置属性值



区别：



3、移除属性



2.4.7H5自定义属性

自定义属性目的：是为了保存并使用数据。有些数据可以保存到页面中而不用保存到数据库中

1、设置H5自定义属性

H5规定自定义属性data-开头做为属性名并且赋值

比如 <div data-index=“1”></div>

或者使用 JS 设置

element.setAttribute(‘data-index’, 2)

2、获取H5自定义属性

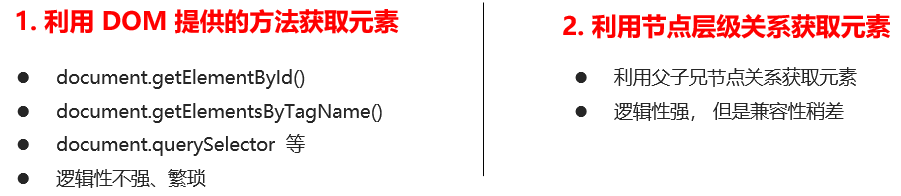
兼容性获取element.getAttribute(‘data-index’);

H5新增element.dataset.index或者element.dataset[‘index’] ie 11才开始支持

2.5节点操作

2.5.1为什么需要节点操作

获取元素通常使用两种方式：



2.5.2节点概述

网页中的所有内容都是节点（标签、属性、文本、注释等），在DOM 中，节点使用 node 来表示

HTML DOM 树中的所有节点均可通过 JavaScript 进行访问，所有 HTML 元素（节点）均可被修改，也可以创建或删除

一般地，节点至少拥有nodeType（节点类型）、nodeName（节点名称）和nodeValue（节点值）这三个基本属性

1）元素节点 nodeType 为 1

2）属性节点 nodeType 为 2

3）文本节点 nodeType 为 3 （文本节点包含文字、空格、换行等）

在实际开发中，节点操作主要操作的是元素节点

2.5.3节点层级

利用 DOM 树可以把节点划分为不同的层级关系，常见的是父子兄层级关系

1、父级节点

node.parentNode

注意：

1）parentNode 属性可返回某节点的父节点，注意是最近的一个父节点

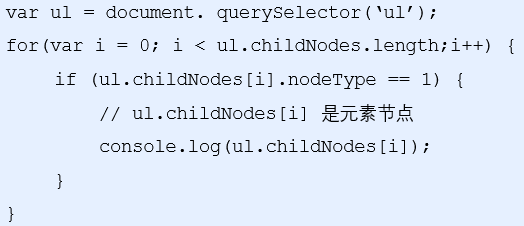
2）如果指定的节点没有父节点则返回 null

2、子节点

parentNode.childNodes（标准）

返回包含指定节点的子节点的集合，该集合为即时更新的集合

注意：返回值里面包含了所有的子节点，包括元素节点，文本节点等，如果只想要获得里面的元素节点，则需要专门处理。 所以一般不提倡使用childNodes



parentNode.children（非标准）

是一个只读属性，返回所有的子元素节点。它只返回子元素节点，其余节点不返回

parentNode.firstChild

返回第一个子节点，找不到则返回null。包含所有的节点

parentNode.lastChild

返回最后一个子节点，找不到则返回null。包含所有的节点

parentNode.firstElementChild

返回第一个子元素节点，找不到则返回null

parentNode.lastElementChild

返回最后一个子元素节点，找不到则返回null

注意：

1）如果想要第一个子元素节点，可以使用 parentNode.chilren[0]

2）如果想要最后一个子元素节点，可以使用 parentNode.chilren[parentNode.chilren.length - 1]

3、兄弟节点

node.nextSibling

返回当前元素的下一个兄弟元素节点，找不到则返回null。包含所有的节点

node.previousSibling

返回当前元素上一个兄弟元素节点，找不到则返回null。包含所有的节点

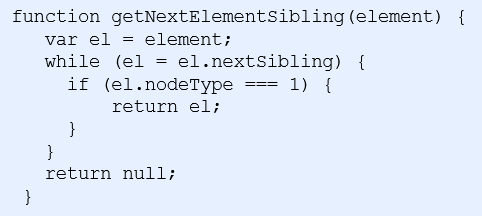
node.nextElementSibling

返回当前元素下一个兄弟元素节点，找不到则返回null

node.previousElementSibling

返回当前元素上一个兄弟节点，找不到则返回null

解决兼容性问题：



2.5.4创建、添加节点

document.createElement('tagName')

创建由 tagName 指定的 HTML 元素。因为这些元素原先不存在，是根据我们的需求动态生成的，所以我们也称为动态创建元素节点

node.appendChild(child)

将一个节点添加到指定父节点的子节点列表末尾。类似于 CSS 里面的 after 伪元素

node.insertBefore(child, 指定元素)

将一个节点添加到父节点的指定子节点前面。类似于 CSS 里面的 before 伪元素

2.5.5删除节点

node.removeChild(child)

方法从 DOM 中删除一个子节点，返回删除的节点

2.5.6复制节点

node.cloneNode()

返回调用该方法的节点的一个副本

注意：

1. 如果括号参数为空或者为 false ，则是浅拷贝，即只克隆复制节点本身，不克隆里面的子节点

2. 如果括号参数为 true ，则是深度拷贝，会复制节点本身以及里面所有的子节点

2.5.7三种动态创建元素区别

1）document.write()

2）element.innerHTML

3）document.createElement()

区别：

1. document.write 是直接将内容写入页面的内容流，但是文档流执行完毕，则它会导致页面全部重绘

2. innerHTML 是将内容写入某个 DOM 节点，不会导致页面全部重绘

3. innerHTML 创建多个元素效率更高（不要拼接字符串，采取数组形式拼接），结构稍微复杂

4. createElement() 创建多个元素效率稍低一点点，但是结构更清晰

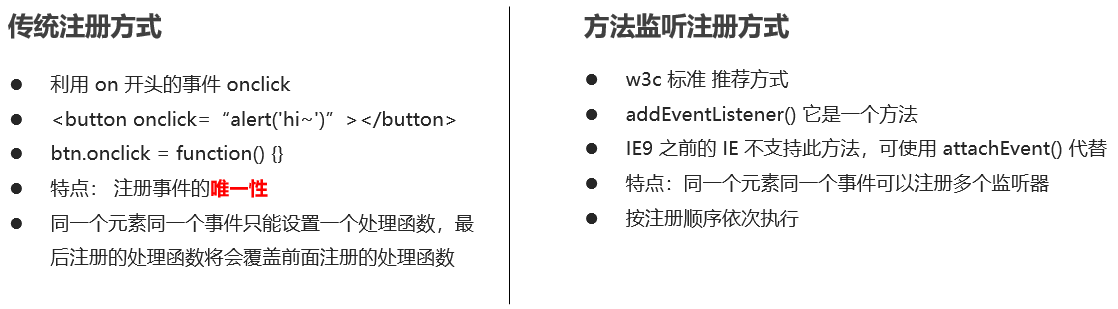
3.事件高级

3.1注册事件

3.1.1概述

给元素添加事件，称为注册事件或者绑定事件

注册事件有两种方式：传统方式和方法监听注册方式



3.1.2addEventListener 事件监听方式

eventTarget.addEventListener(type, listener, useCapture)

将指定的监听器注册到 eventTarget（目标对象）上，当该对象触发指定的事件时，就会执行事件处理函数

该方法接收三个参数：

1）type：事件类型字符串，比如 click 、mouseover ，注意这里不要带 on

2）listener：事件处理函数，事件发生时，会调用该监听函数

3）useCapture：可选参数，是一个布尔值，默认是 false

3.2删除事件

1、传统注册方式

eventTarget.onclick = null;

2、方法监听注册方式

eventTarget.removeEventListener(type, listener, useCapture);

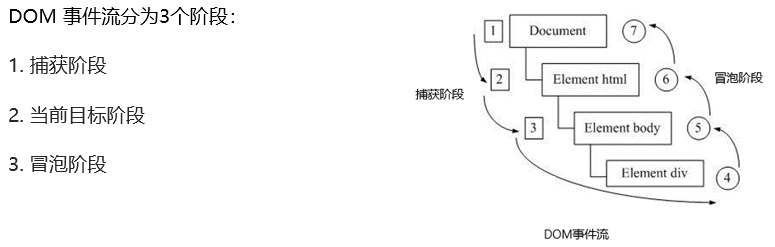
eventTarget.detachEvent(eventNameWithOn, callback);

3.3DOM事件流

事件流描述的是从页面中接收事件的顺序

事件发生时会在元素节点之间按照特定的顺序传播，这个传播过程即DOM事件流

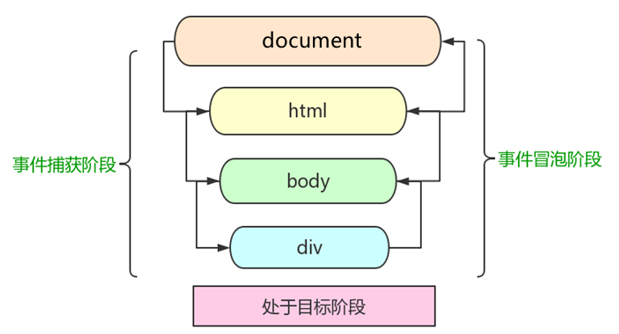
比如我们给一个div 注册了点击事件：



事件冒泡：IE最早提出，事件开始时由最具体的元素接收，然后逐级向上传播到到DOM最顶层节点的过程。

事件捕获：网景最早提出，由DOM最顶层节点开始，然后逐级向下传播到到最具体的元素接收的过程

我们向水里面扔一块石头，首先它会有一个下降的过程，这个过程就可以理解为从最顶层向事件发生的最具体元素（目标点）的捕获过程；之后会产生泡泡，会在最低点（ 最具体元素）之后漂浮到水面上，这个过程相当于事件冒泡



注意

1）JS 代码中只能执行捕获或者冒泡其中的一个阶段

2）onclick 和 attachEvent 只能得到冒泡阶段

3）addEventListener(type, listener, useCapture)第三个参数如果是 true，表示在事件捕获阶段调用事件处理程序；如果是 false（不写默认就是false），表示在事件冒泡阶段调用事件处理程序

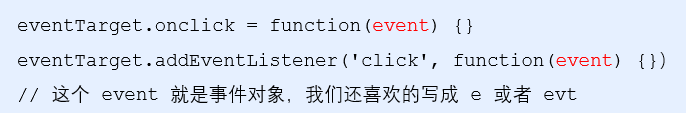
4）实际开发中我们很少使用事件捕获，我们更关注事件冒泡

5）有些事件是没有冒泡的，比如 onblur、onfocus、onmouseenter、onmouseleave

6）事件冒泡有时候会带来麻烦，有时候又会帮助很巧妙的做某些事件

3.4事件对象

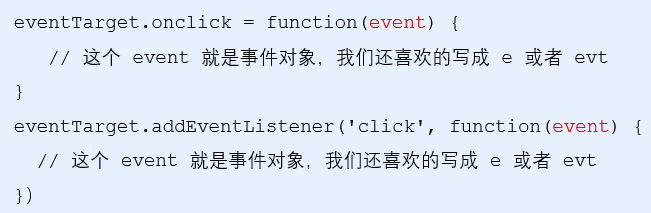
3.4.1什么是事件对象



官方解释：event 对象代表事件的状态，比如键盘按键的状态、鼠标的位置、鼠标按钮的状态

简单理解：事件发生后，跟事件相关的一系列信息数据的集合都放到这个对象里面，这个对象就是事件对象 event，它有很多属性和方法

3.4.2事件对象的使用语法



注意：

1）这个event是个形参，系统帮我们设定为事件对象，不需要传递实参过去

2）当我们注册事件时，event对象就会被系统自动创建，并依次传递给事件监听器（事件处理函数）

3.4.3事件对象的兼容性方案

事件对象本身的获取存在兼容问题：

1.标准浏览器中是浏览器给方法传递的参数，只需要定义形参e就可以获取到

2.在 IE6~8 中，浏览器不会给方法传递参数，如果需要的话，需要到window.event中获取查找

解决:

e = e || window.event;

3.4.4事件对象的常见属性和方法

e.target 和 this 的区别：

1）this 是事件绑定的元素，这个函数的调用者（绑定这个事件的元素）

2）e.target 是事件触发的元素



3.5阻止事件冒泡

3.5.1阻止事件冒泡的两种方式

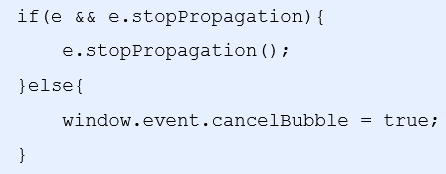
标准写法：利用事件对象里面的 stopPropagation()方法

e.stopPropagation()

非标准写法：IE 6-8 利用事件对象 cancelBubble 属性

e.cancelBubble = true;

3.5.2兼容性解决



3.6事件委托

事件委托也称为事件代理， 在jQuery里面称为事件委派

原理：不是每个子节点单独设置事件监听器，而是事件监听器设置在其父节点上，然后利用冒泡原理影响设置每个子节点

作用：提高程序的性能

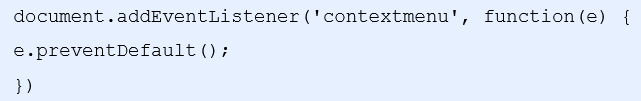
3.7常用的鼠标事件

3.7.1常用事件

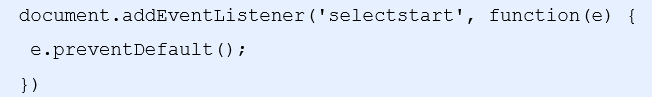


1、禁止鼠标右键菜单

contextmenu主要控制应该何时显示上下文菜单，主要用于程序员取消默认的上下文菜单



2.禁止鼠标选中（selectstart 开始选中）



3.7.2鼠标事件对象

event对象代表事件的状态，跟事件相关的一系列信息的集合。现阶段我们主要是用鼠标事件对象 MouseEvent 和键盘事件对象 KeyboardEvent



3.8常用的键盘事件

3.8.1常用事件



注意：

1）如果使用addEventListener不需要加on

2）onkeypress和前面2个的区别是，它不识别功能键，比如左右箭头，shift等

3）三个事件的执行顺序是： keydown--keypress—keyup

3.8.2键盘事件对象



注意：

1）onkeydown 和 onkeyup 不区分字母大小写，onkeypress 区分字母大小写。

2）在我们实际开发中，更多的使用keydown和keyup，它能识别所有的键（包括功能键）

3）keyCode属性能区分大小写，返回不同的ASCII值

4.BOM浏览器对象模型

4.1BOM描述

4.1.1什么是BOM

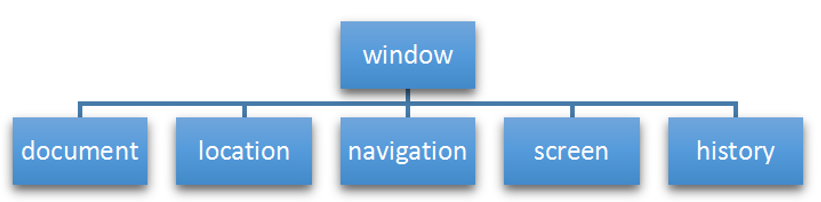
1）BOM（Browser Object Model）即浏览器对象模型，它提供了独立于内容而与浏览器窗口进行交互的对象，其核心对象是window

2）BOM 由一系列相关的对象构成，并且每个对象都提供了很多方法与属性

3）BOM 缺乏标准，JavaScript 语法的标准化组织是 ECMA，DOM 的标准化组织是 W3C，BOM 最初是Netscape 浏览器标准的一部分

4.1.2BOM的构成

BOM 比 DOM 更大，它包含 DOM



window对象是浏览器的顶级对象，它具有双重角色

1）它是JS访问浏览器窗口的一个接口

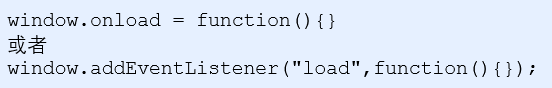
2）它是一个全局对象。定义在全局作用域中的变量、函数都会变成 window 对象的属性和方法

在调用的时候可以省略 window，前面学习的对话框都属于 window 对象方法，如 alert()、prompt() 等

注意：window下的一个特殊属性 window.name

4.2window对象的常见事件

4.2.1窗口加载事件



window.onload 是窗口 (页面）加载事件,当文档内容完全加载完成会触发该事件(包括图像、脚本文件、CSS 文件等), 就调用的处理函数

注意：

1）有了 window.onload 就可以把 JS 代码写到页面元素的上方，因为 onload 是等页面内容全部加载完毕，再去执行处理函数

2）window.onload 传统注册事件方式，只能写一次，如果有多个，会以最后一个 window.onload 为准

3）如果使用 addEventListener 则没有限制



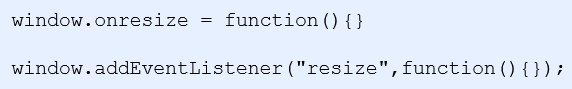
DOMContentLoaded 事件触发时，仅当DOM加载完成，不包括样式表，图片，flash等等

注意：

1）Ie9以上才支持

2）如果页面的图片很多的话, 从用户访问到onload触发可能需要较长的时间, 交互效果就不能实现，必然影响用户的体验，此时用 DOMContentLoaded 事件比较合适

4.2.2调整窗口大小事件



window.onresize是调整窗口大小加载事件, 当触发时就调用的处理函数

注意：

1）只要窗口大小发生像素变化，就会触发这个事件。

2）我们经常利用这个事件完成响应式布局，window.innerWidth 当前屏幕的宽度

4.3定时器

4.3.1两种定时器

window 对象给我们提供了2个非常好用的方法-定时器

1）setTimeout()

2）setInterval()

4.3.2setTimeout()定时器



setTimeout() 方法用于设置一个定时器，该定时器在定时器到期后执行调用函数

注意：

1）window 可以省略。

2）这个调用函数可以直接写函数，或者写函数名或者采取字符串’函数名()’三种形式。第三种不推荐

3）延迟的毫秒数省略默认是0，如果写，必须是毫秒。

4）因为定时器可能有很多，所以我们经常给定时器赋值一个标识符

setTimeout() 这个调用函数我们也称为回调函数callback

普通函数是按照代码顺序直接调用。

而这个函数，需要等待时间，时间到了才去调用这个函数，因此称为回调函数

element.onclick = function(){}或者element.addEventListener(“click”, fn);里面的函数也是回调函数

4.3.3停止setTimeout()定时器



clearTimeout()方法取消了先前通过调用setTimeout()建立的定时器

注意：

1）window 可以省略

2）里面的参数就是定时器的标识符

4.3.4setInterval()定时器



setInterval()方法重复调用一个函数，每隔这个时间，就去调用一次回调函数

注意：

1）window 可以省略

2）这个调用函数可以直接写函数，或者写函数名或者采取字符串 '函数名()' 三种形式

3）间隔的毫秒数省略默认是0，如果写，必须是毫秒，表示每隔多少毫秒就自动调用这个函数

4）因为定时器可能有很多，所以我们经常给定时器赋值一个标识符

5）第一次执行也是间隔毫秒数之后执行，之后每隔毫秒数就执行一次

4.3.5停止setInterval()定时器



clearInterval()方法取消了先前通过调用setInterval()建立的定时器

注意：

1）window 可以省略

2）里面的参数就是定时器的标识符

4.3.6this

this的指向在函数定义的时候是确定不了的，只有函数执行的时候才能确定this到底指向谁，一般情况下this的最终指向的是那个调用它的对象

现阶段，我们先了解一下几个this指向：

1）全局作用域或者普通函数中this指向全局对象window（注意定时器里面的this指向window）

2）方法调用中谁调用this指向谁

3）构造函数中this指向构造函数的实例

4.4JS执行队列

4.4.1JS是单线程

JavaScript 语言的一大特点就是单线程，也就是说，同一个时间只能做一件事。这是因为 Javascript 这门脚本语言诞生的使命所致——JavaScript 是为处理页面中用户的交互，以及操作 DOM 而诞生的。比如我们对某个 DOM 元素进行添加和删除操作，不能同时进行。应该先进行添加，之后再删除

单线程就意味着，所有任务需要排队，前一个任务结束，才会执行后一个任务。这样所导致的问题是：如果 JS 执行的时间过长，这样就会造成页面的渲染不连贯，导致页面渲染加载阻塞的感觉

4.4.2同步和异步

为了解决这个问题，利用多核CPU的计算能力，HTML5提出Web Worker标准，允许 JavaScript脚本创建多个线程。于是，JS 中出现了同步和异步

1、同步

前一个任务结束后再执行后一个任务，程序的执行顺序与任务的排列顺序是一致的、同步的。比如做饭的同步做法：我们要烧水煮饭，等水开了（10分钟之后），再去切菜，炒菜

2、异步

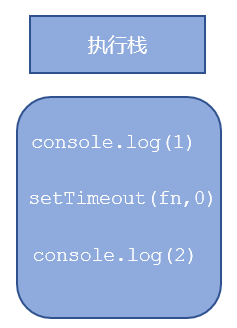
你在做一件事情时，因为这件事情会花费很长时间，在做这件事的同时，你还可以去处理其他事情。比如做饭的异步做法，我们在烧水的同时，利用这10分钟，去切菜，炒菜

本质区别： 这条流水线上各个流程的执行顺序不同

4.4.3同步任务和异步任务

1、同步任务

同步任务都在主线程上执行，形成一个执行栈



2、异步任务

JS 的异步是通过回调函数实现的

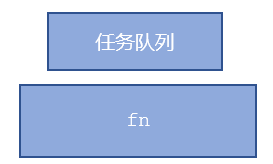
一般而言，异步任务有以下三种类型:

1）普通事件，如 click、resize 等

2）资源加载，如 load、error 等

3）定时器，包括 setInterval、setTimeout 等

异步任务相关回调函数添加到任务队列中（任务队列也称为消息队列）

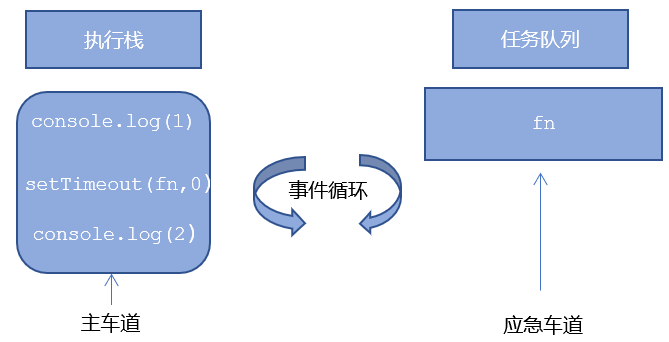


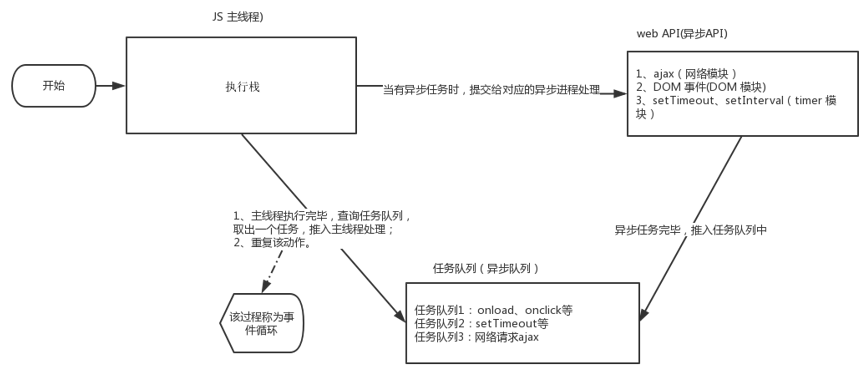
4.4.4JS执行机制

1.先执行执行栈中的同步任务

2.异步任务（回调函数）放入任务队列中

3.一旦执行栈中的所有同步任务执行完毕，系统就会按次序读取任务队列中的异步任务，于是被读取的异步任务结束等待状态，进入执行栈，开始执行





由于主线程不断的重复获得任务、执行任务、再获取任务、再执行，所以这种机制被称为事件循环（event loop）

4.5location对象

4.5.1什么是location对象

window对象给我们提供了一个location属性用于获取或设置窗体的URL，并且可以用于解析URL。因为这个属性返回的是一个对象，所以我们将这个属性也称为location对象

4.5.2URL

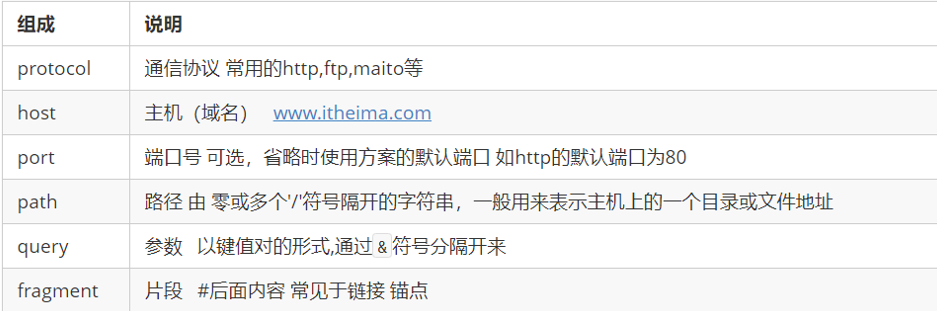
统一资源定位符 (Uniform Resource Locator, URL) 是互联网上标准资源的地址。互联网上的每个文件都有一个唯一的URL，它包含的信息指出文件的位置以及浏览器应该怎么处理它

URL 的一般语法格式为：



例如：



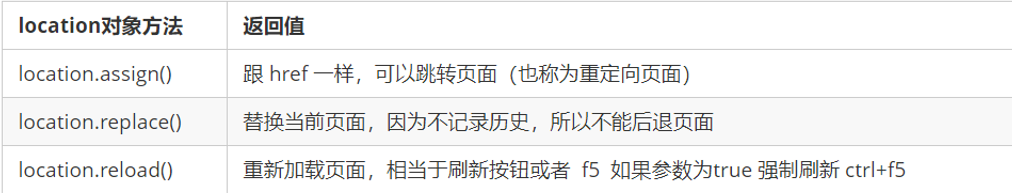


4.5.3location对象的属性



重点：href和search

4.5.4location对象的方法



4.6navigator对象

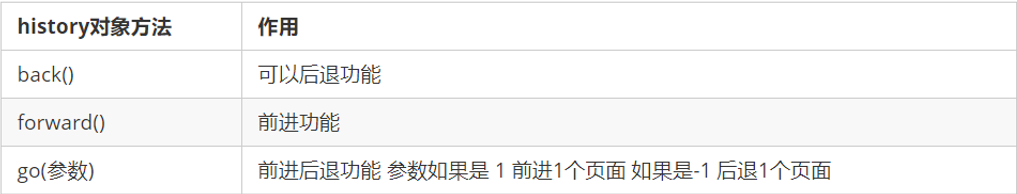
navigator对象包含有关浏览器的信息，它有很多属性，我们最常用的是userAgent，该属性可以返回由客户机发送服务器的user-agent头部的值

下面前端代码可以判断用户那个终端打开页面，实现跳转：



4.7history对象

window 对象给我们提供了一个 history 对象，与浏览器历史记录进行交互。该对象包含用户（在浏览器窗口中）访问过的 URL



history对象一般在实际开发中比较少用，但是会在一些OA办公系统中见到



5.PC端网页特效

5.1元素偏移量 offset 系列

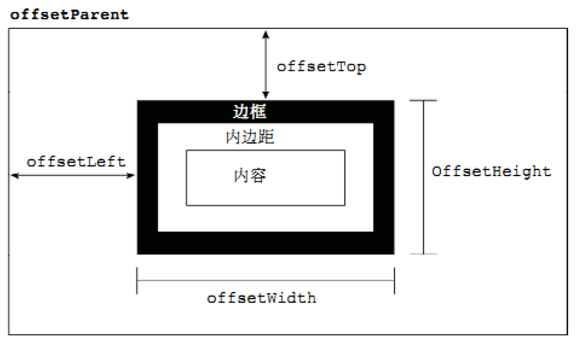
5.1.1offset概述

offset 翻译过来就是偏移量，我们使用offset系列相关属性可以动态的得到该元素的位置（偏移）、大小等

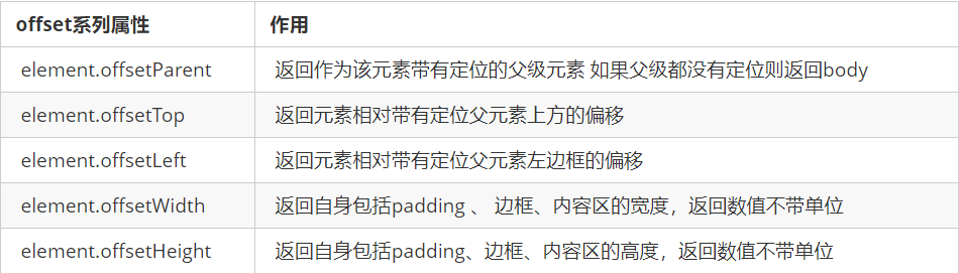
1）获得元素距离带有定位父元素的位置

2）获得元素自身的大小（宽度高度）

注意：返回的数值都不带单位



offset 系列常用属性：



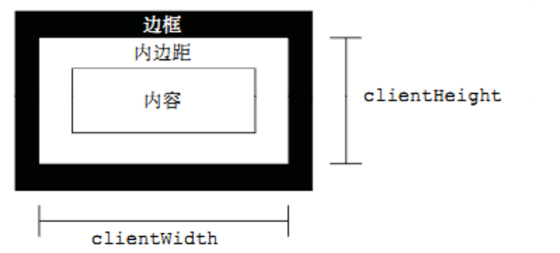
5.1.2offset与style区别



5.2元素可视区client系列

client 翻译过来就是客户端，我们使用client系列的相关属性来获取元素可视区的相关信息。通过 client 系列的相关属性可以动态的得到该元素的边框大小、元素大小等

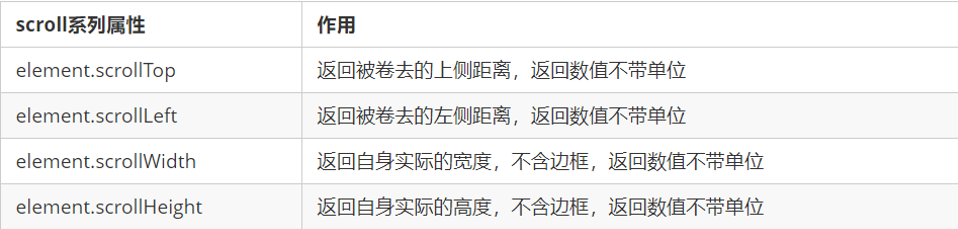


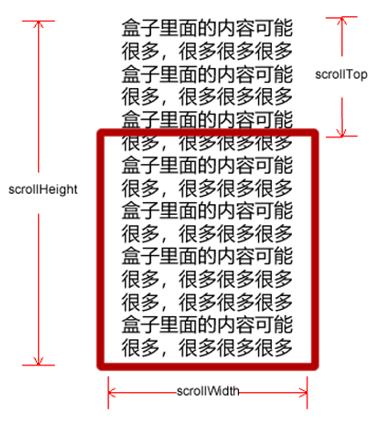


5.3元素滚动scroll系列

5.3.1元素 scroll 系列属性

scroll 翻译过来就是滚动的，我们使用 scroll 系列的相关属性可以动态的得到该元素的大小、滚动距离等





5.3.2页面被卷去的头部

如果浏览器的高（或宽）度不足以显示整个页面时，会自动出现滚动条。当滚动条向下滚动时，页面上面被隐藏掉的高度，我们就称为页面被卷去的头部。滚动条在滚动时会触发 onscroll事件

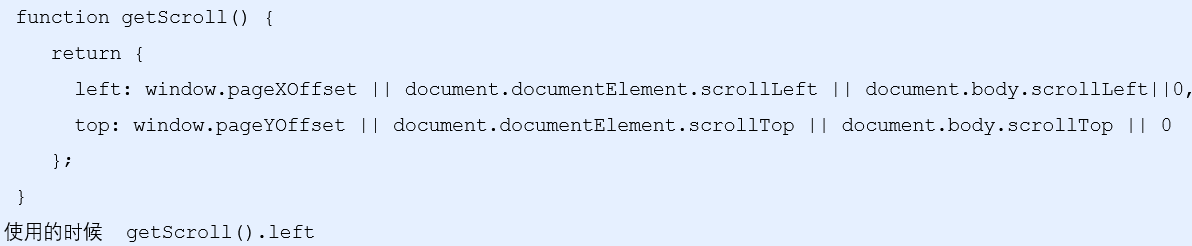
5.3.3页面被卷去的头部兼容性解决方案

需要注意的是，页面被卷去的头部，有兼容性问题，因此被卷去的头部通常有如下几种写法：

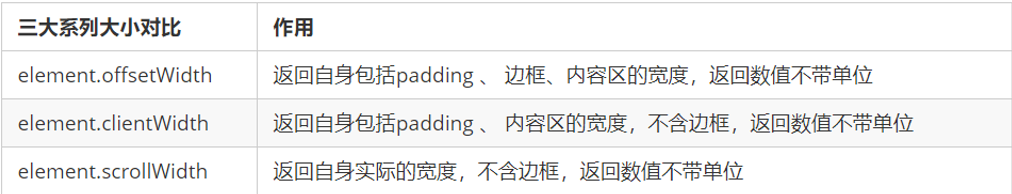
1.声明了 DTD，使用document.documentElement.scrollTop

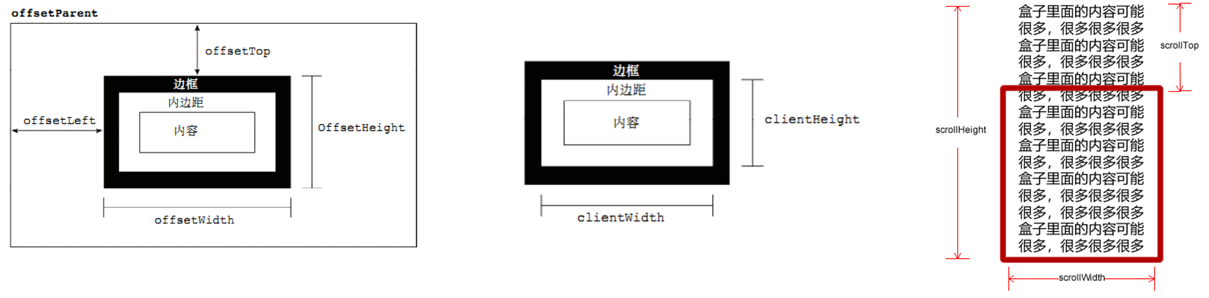
2.未声明 DTD，使用document.body.scrollTop

3.新方法 window.pageYOffset 和 window.pageXOffset，IE9开始支持



5.4三大系列总结





他们主要用法：

1）offset系列经常用于获得元素位置offsetLeft offsetTop

2）client经常用于获取元素大小clientWidth clientHeight

3）scroll经常用于获取滚动距离scrollTop scrollLeft

4）注意页面滚动的距离通过window.pageXOffset获得

5.5mouseenter 鼠标事件

1）当鼠标移动到元素上时就会触发mouseenter事件

2）类似 mouseover，它们两者之间的差别是

3）mouseover鼠标经过自身盒子会触发，经过子盒子还会触发。mouseenter只会经过自身盒子触发

4）之所以这样，就是因为mouseenter不会冒泡

5）跟mouseenter搭配 鼠标离开 mouseleave同样不会冒泡

5.6动画函数封装

5.6.1动画实现原理

核心原理：通过定时器setInterval()不断移动盒子位置

实现步骤：

1. 获得盒子当前位置

2. 让盒子在当前位置加上1个移动距离

3. 利用定时器不断重复这个操作

4. 加一个结束定时器的条件

5. 注意此元素需要添加定位，才能使用element.style.left

5.6.2动画函数简单封装

注意函数需要传递2个参数，动画对象和移动到的距离

5.6.3动画函数给不同元素记录不同定时器

如果多个元素都使用这个动画函数，每次都要var 声明定时器。我们可以给不同的元素使用不同的定时器（自己专门用自己的定时器）

核心原理：利用 JS 是一门动态语言，可以很方便的给当前对象添加属性

5.6.4缓动效果原理

缓动动画就是让元素运动速度有所变化，最常见的是让速度慢慢停下来

思路：

1）让盒子每次移动的距离慢慢变小，速度就会慢慢落下来

2）核心算法： (目标值 - 现在的位置)/10 做为每次移动的距离步长

3）停止的条件是： 让当前盒子位置等于目标位置就停止定时器

4）注意步长值需要取整

5.6.5动画函数多个目标值之间移动

可以让动画函数从 800 移动到 500

当我们点击按钮时候，判断步长是正值还是负值：

1）如果是正值，则步长往大了取整

2）如果是负值，则步长向小了取整

5.6.6动画函数添加回调函数

回调函数原理：函数可以作为一个参数。将这个函数作为参数传到另一个函数里面，当那个函数执行完之后，再执行传进去的这个函数，这个过程就叫做回调

回调函数写的位置：定时器结束的位置

5.6.7动画函数封装到单独JS文件里面

因为以后经常使用这个动画函数，可以单独封装到一个JS文件里面，使用的时候引用这个JS文件即可

1）单独新建一个JS文件

2）HTML文件引入JS文件

5.6.8节流阀

防止轮播图按钮连续点击造成播放过快

节流阀目的：当上一个函数动画内容执行完毕，再去执行下一个函数动画，让事件无法连续触发

核心实现思路：

1）利用回调函数，添加一个变量来控制，锁住函数和解锁函数

2）开始设置一个变量var flag = true;

3）If(flag) {flag = false; do something} 关闭水龙头

4）利用回调函数，动画执行完毕， flag = true 打开水龙头

6.移动端网页特效

6.1触屏事件

6.1.1概述

移动端浏览器兼容性较好，我们不需要考虑以前 JS 的兼容性问题，可以放心的使用原生 JS 书写效果，但是移动端也有自己独特的地方。比如触屏事件 touch（也称触摸事件），Android 和 IOS 都有

touch 对象代表一个触摸点。触摸点可能是一根手指，也可能是一根触摸笔。触屏事件可响应用户手指（或触控笔）对屏幕或者触控板操作

常见的触屏事件如下：



6.1.2触摸事件对象（TouchEvent）

TouchEvent 是一类描述手指在触摸平面（触摸屏、触摸板等）的状态变化的事件。这类事件用于描述一个或多个触点，使开发者可以检测触点的移动，触点的增加和减少，等等

touchstart、touchmove、touchend 三个事件都会各自有事件对象

触摸事件对象重点我们看三个常见对象列表：



因为平时我们都是给元素注册触摸事件，所以重点记住 targetTocuhes

6.1.3移动端拖动元素

1.touchstart、touchmove、touchend 可以实现拖动元素

2.但是拖动元素需要当前手指的坐标值 我们可以使用targetTouches[0]里面的pageX和pageY

3.移动端拖动的原理：手指移动中，计算出手指移动的距离，然后用盒子原来的位置+手指移动的距离

4.手指移动的距离：手指滑动中的位置减去手指刚开始触摸的位置

拖动元素三步曲：

1）触摸元素 touchstart：获取手指初始坐标，同时获得盒子原来的位置

2）移动手指 touchmove：计算手指的滑动距离，并且移动盒子

3）离开手指 touchend

注意：手指移动也会触发滚动屏幕所以这里要阻止默认的屏幕滚动e.preventDefault();

6.2移动端常见特效

6.2.1classList属性

classList属性是HTML5新增的一个属性，返回元素的类名。但是ie10以上版本支持。

该属性用于在元素中添加，移除及切换 CSS 类。有以下方法：

1、添加类

element.classList.add(’类名’);

2、移除类

element.classList.remove(’类名’);

3、切换类

element.classList.toggle(’类名’);

注意：以上方法里面，所有类名都不带点

6.2.2click延时解决方案

移动端click事件会有300ms的延时，原因是移动端屏幕双击会缩放页面

解决方案：

1. 禁用缩放。浏览器禁用默认的双击缩放行为并且去掉 300ms 的点击延迟



2. 利用touch事件自己封装这个事件解决 300ms 延迟

原理就是：

1）当我们手指触摸屏幕，记录当前触摸时间

2）当我们手指离开屏幕， 用离开的时间减去触摸的时间

3）如果时间小于150ms，并且没有滑动过屏幕， 那么我们就定义为点击

3. 使用插件。fastclick 插件解决300ms延迟

6.3移动端常用开发插件

6.3.1什么是插件

JS 插件是 js 文件，它遵循一定规范编写，方便程序展示效果，拥有特定功能且方便调用。如轮播图和瀑布流插件

特点：它一般是为了解决某个问题而专门存在，其功能单一，并且比较小

6.3.2插件的使用

1）确认插件实现的功能

2）去官网查看使用说明

3）下载插件

4）打开demo实例文件，查看需要引入的相关文件，并且引入

5）复制demo实例文件中的结构html，样式css以及js代码

6.4移动端常用开发框架

6.4.1框架概述

框架，顾名思义就是一套架构，它会基于自身的特点向用户提供一套较为完整的解决方案。框架的控制权在框架本身，使用者要按照框架所规定的某种规范进行开发

插件一般是为了解决某个问题而专门存在，其功能单一，并且比较小

前端常用的框架有 Bootstrap、Vue、Angular、React 等。既能开发PC端，也能开发移动端

前端常用的移动端插件有 swiper、superslide、iscroll等

框架： 大而全，一整套解决方案

插件： 小而专一，某个功能的解决方案

6.4.2Bootstrap

Bootstrap 是一个简洁、直观、强悍的前端开发框架，它让 web 开发更迅速、简单。

它能开发PC端，也能开发移动端

Bootstrap JS插件使用步骤：

1）引入相关js 文件

2）复制HTML 结构

3）修改对应样式

4）修改相应JS 参数

7.本地存储

7.1概述

随着互联网的快速发展，基于网页的应用越来越普遍，同时也变的越来越复杂，为了满足各种各样的需求，会经常性在本地存储大量的数据，HTML5规范提出了相关解决方案

7.2本地存储特性

1、数据存储在用户浏览器中

2、设置、读取方便、甚至页面刷新不丢失数据

3、容量较大，sessionStorage约5M、localStorage约20M

4、只能存储字符串，可以将对象JSON.stringify() 编码后存储

7.3window.sessionStorage

1、生命周期为关闭浏览器窗口

2、在同一个窗口(页面)下数据可以共享

3、以键值对的形式存储使用

存储数据：



获取数据：



删除数据：



删除所有数据：



7.4window.localStorage

1、声明周期永久生效，除非手动删除 否则关闭页面也会存在

2、可以多窗口（页面）共享（同一浏览器可以共享）

3、以键值对的形式存储使用

存储数据：



获取数据：



删除数据：



删除所有数据：

