Web 前端知识体系精简

2017-06-20 前端大全

（点击上方公众号，可快速关注）

> 作者：一像素
> www.cnblogs.com/onepixel/p/7021506.html
> 如有好文章投稿，请点击 → 这里了解详情

Web前端技术由html、css和javascript三大部分构成，是一个庞大而复杂的技术体系，其复杂程度不低于任何一门后端语言。而我们在学习它的时候往往是先从某一个点切入，然后不断地接触和学习新的知识点，因此对于初学者很难理清楚整个体系的脉络结构。本文将对Web前端知识体系进行简单的梳理，对应的每个知识点点到为止，不作详细介绍。目的是帮助大家审查自己的知识结构是否完善，如有遗漏或不正确的地方，希望共勉。

JAVASCRIPT 篇

0、基础语法

Javascript基础语法包括：变量定义、数据类型、循环、选择、内置对象等。

数据类型有string,number,boolean,null,undefined,object等。其中，string,number和boolean是基础类型,null和undefined是JS中的两个特殊类型,object是引用类型。

Javascript可以通过typeof来判断基础数据类型，但不能够准确判断引用类型, 因此需要用到另外一个方法，那就是Object的toString,关于数据类型及其判断可以参考以下博客：*数据类型详解* 和 *判断JS数据*

类型的四种方法

JS常用的内置对象有Date、Array、JSON,RegExp等。 一般来讲，Date和Array用的最频繁，JSON可以对对象和数组进行序列化和反序列化，还有一个作用就是实现**对象的深拷贝。**

RegExp即正则表达式，是处理字符串的利器。 关于数据类型和正则表达式的介绍可以参考博客：*ES5对数组增强的9个API* 和 *JS正则表达式精简*

1、函数原型链

Javascript虽然没有继承概念，但Javascript在函数Function对象中建立了原型对象prototype，并以Function对象为主线，从上至下，在内部构建了一条原型链。

简单来说就是建立了变量查找机制，当访问一个对象的属性时，先查找对象本身是否存在，如果不存在就去该对象所在的原型连上去找，直到Object对象为止，如果都没有找到该属性才会返回undefined。

因此我们经常会利用函数的原型机制来实现JS继承。关于函数原型链可参考博客：*JS原型对象和原型链*

2、函数作用域

函数作用域就是变量在声明它们的函数体以及这个函数体嵌套的任意函数体内都是有定义的。**在JS中没有会块级作用域，只有函数作用域**，因此JS中还存在着另外一种怪异现象，那就是**变量提升。**关于作用域的介绍请参考博客：*函数的作用域和作用域链*

3、函数指针 this

this 存在于函数中，**它指向的是该函数在运行时被调用的那个对象。**在实际项目中，遇到this的坑比较多，因此需要对this作深入的理解。

Function对象还提供了call、apply和bind等方法来改变函数的this指向，其中call和apply主动执行函数，bind一般在事件回调中使用，而call和apply的区别只是参数的传递方式不同。关于call,apply和bind的用户请参考博客：*详解JS的call,apply和bind*

4、构造函数 new

JS中的函数即可以是构造函数又可以当作普通函数来调用，当使用new来创建对象时，对应的函数就是构造函数，通过对象来调用时就是普通函数。

普通函数的创建有：显式声明、匿名定义、new Function() 等三种方式。

当通过new来创建一个新对象时，JS底层将新对象的原型链指向了构造函数的原型对象，于是就在新对象和函数对象之间建立了一条原型链，通过新对象可以访问到函数对象原型prototype中的方法和属性。new的详细介绍请参考博客：*理解JS中的new运算符*

5、闭包

闭包其实是一个主动执行的代码块，这个代码块的特殊之处是可以永久保存局部变量，但又不污染全局变量，可以形成一个独立的执行过程，因此我们经常用闭包来定义组件。关于闭包的介绍请参考：*干货分享：让你分分钟学会JS闭包*

6、单线程和异步队列

setTimeout和setInterval是JS内置的两个定时器，使用很简单，但这两个方法背后的原理却不简单。

我们知道，JS是单线程语言，在浏览器中，当JS代码被加载时，浏览器会为其分配一个主线程来执行任务(函数)，主线程会形成一个全局执行环境，执行环境采用栈的方式将待执行任务按顺序依次来执行。

但在浏览器中有一些任务是非常耗时的，比如http请求、定时器、事件回调等，为了保证其他任务的执行效率不被影响，**JS在执行环境中维护了一个异步队列(也叫工作线程)，并将这些任务放入队列中进行等待，这些任务的执行时机并不确定，只有当主线程的任务执行完成以后，才会去检查异步队列中的任务是否需要开始执行。**这就是为什么setTimeout(fn,0) 始终要等到最后执行的原因。关于单线程和异步队列问题请参考：*setTimeout（0）*

7、异步通讯 Ajax技术

Ajax是浏览器专门用来和服务器进行交互的异步通讯技术，其核心对象是XMLHttpRequest,通过该对象可以创建一个Ajax请求。为了防止XSS攻击，浏览器对Ajax做了限制，不允许Ajax跨域请求服务器,就是只能访问当前域名下的url。

当然，如果确信你的站点不存在跨域的风险，可以在服务端主动开启跨域请求。 也可以直接通过CORS或JSONP来实现。

JSONP是利用脚本(script)跨域能力来模拟Ajax请求。

CORS是一个W3C标准，全称是”跨域资源共享”（Cross-origin resource sharing）。它允许浏览器向跨源服务器，发出XMLHttpRequest请求，从而克服了AJAX只能同源使用的限制。 关于CORS的介绍请参考：*跨域资源共享 CORS 详解*

8、DOM对象 document

document对象里保存着整个web页面dom结构，在页面上所有的元素最终都会映射为一个dom对象。document也提供了很多api来查找特定的dom对象，比如getElementById,querySelector等等。

9、事件系统 Event

事件是用户与页面交互的基础，到目前为止，DOM事件从PC端的 鼠标事件(mouse) 发展到移动端的 触摸事件(touch) 和 手势事件(guesture)

由于DOM结构可能会多层嵌套，因此也衍生出了两种事件流：事件捕获和事件冒泡，后者最常用。利用事件冒泡机制可以实现很多功能，比如页面点击统计。关于两种事件流的介绍请参考：*事件冒泡和捕获*

除此之外，在页面初始化、滚动、隐藏、返回等操作时分别内置了onload/onDOMContentLoaded、onscroll、onvisibility和onhashchange等事件，如果想要捕获这些事件，需要通过addEventLisener/attachEvent来进行绑定。

10、全局对象 window

在JS中，当一段JS代码在浏览器中被加载执行，JS引擎会在内存中构建一个全局执行环境，**执行环境的作用是保证所有的函数能按照正确的顺序被执行**，而window对象则是这个执行环境中的一个全局对象，window对象中内置了很多操作api和对象，document对象就是其中一个。关于JS执行环境的介绍请参考博客：*深入理解JS执行细节*

CSS 篇

css是用来对html进行修饰的一门语言。

1、选择器

css的选择器有很多种，常用的有类选择器、标签选择器、ID选择器、后代选择器、群组选择器、伪类选择器(before/after)、兄弟选择器(+~)、属性选择器等等。

2、定位

定位一般有相对定位(relative)、绝对定位(absolute)、固定定位(fixed)，relative和absolute在移动端用的最多，fixed 在移动端有兼容性问题，因此不推荐使用，在移动端替代fixed的方案是 absolute+内部滚动。

3、浮动

设置float为left或right，就能使该元素脱离文档流，向左或向右浮动。一般在做宫格模式布局时会用到，如果子元素全部设置为浮动，则父元素是塌陷的，这时就需要清除浮动，清除浮动的方法也很多，常用的方法是在元素末尾加空元素设置clear:both, 更高级一点的就给父容器设置before/after来模拟一个空元素，还可以直接设置overflow:auto/hidden。除过浮动可以实现宫格模式，行内盒子(inline-block)和table也可以。

4、盒子模型

盒子模型是css最重要的一个概念，也是css布局的基石。 常见的盒子模型有块级盒子(block)和行内盒子(inline-block)，盒子最关键的几个属性包括margin、border、padding和content,这几个元素可以设置盒子和盒子之间的关系以及盒子和内容之间的关系。还有一个问题是计算盒子的大小，需要注意的是，box-sizing属性的设置会影响盒子的width和height。只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。

5、Flex布局

Flex布局的容器是一个伸缩容器，首先容器本身会更具容器中的元素动态设置自身大小；然后当Flex容器被应用一个大小时（width和height），将会自动调整容器中的元素适应新大小。Flex容器也可以设置伸缩比例和固定宽度，还可以设置容器中元素的排列方向（横向和纵向）和是否支持元素的自动换行。有了这个神器，做页面布局的可以方便很多了。注意，设为Flex布局以后，子元素的float、clear和vertical-align属性将失效。

6、transition(过渡) 和 transform(旋转)

应用transform可以对元素进行平移(translate)、旋转(rotate)、放大缩小(scale)、倾斜(skew)等处理，而transition使css属性值(包括transform)在一段时间内平滑的过渡。使用transition和transform就可以实现页面的滑动切换效果。

7、动画 Animation

Animation首先需要设置一个动画函数，然后以这个动画的方式来改变元素的css属性之的变化，动画可以被设置为永久循环演示。 和transition相比，animation设置动画效果更灵活更丰富，二者还有一个区别是：transition只能通过主动改变元素的css值才能触发动画效果，而animation一旦被应用，就开始执行动画。

8、Sprite图

对于大型站点，为了减少http请求的次数，一般会将常用的小图标排到一个大图中，页面加载时只需请求一次网络， 然后在css中通过设置background-position来控制显示所需要的小图标。

9、字体图标 iconfont

所谓字体图标就是将常用的图标转化为字体资源存在文件中，通过在CSS中引用该字体文件，然后可以直接通过控制字体的css属性来设置图标的样式。

HTML 篇

1、Web语义化 和 SEO

html 常规标签有html,head,body,div,span,table,ul,ol,dl,p,b,h1~h6,strong,form,input,img,em,i 等等，另外html5 还新增了很多语义化的标签，比如header,acticle,aside,section,footer,audio,radio 等等。

Web语义化是指使用语义恰当的标签，使页面有良好的结构，页面元素有含义，能够让人和搜索引擎都容易理解。

SEO是指在了解搜索引擎自然排名机制的基础之上，对网站进行内部及外部的调整优化，改进网站在搜索引擎中关键词的自然排名，获得更多的展现量，吸引更多目标客户点击访问网站，从而达到互联网营销及品牌建设的目标。

搜索引擎通过爬虫技术获取的页面就是由一堆html标签组成的代码，，人可以通过可视化的方式来判断页面上哪些内容是重点，而机器做不到。 但搜索引擎会根据标签的含义来判断内容的权重，因此，在合适的位置使用恰当的标签，使整个页面的语义明确，结构清晰，搜索引擎才能正确识别页面中的重要内容，并予以较高的权值。比如h1~h6这几个标签在SEO中的权值非常高，用它们作页面的标题就是一个简单的SEO优化。

2、页面渲染机制

页面渲染就是浏览器的渲染引擎将html代码根据CSS定义的规则显示在浏览器窗口中的过程。大致工作原理如下：

- 用户输入网址，浏览器向服务器发出请求，服务器返回html文件；
- 渲染引擎开始载入html代码，并将HTML中的标签转化为DOM节点，生成DOM树；
- 如果中引用了外部css文件，则发出css文件请求，服务器返回该文件；
- 如果中引用了外部js文件，则发出js文件请求，服务器返回该文件后开始运行；
- 渲染引擎继续载入html中的部分的代码，并开始解析前面返回的css文件，然后根据css选择器计算出节点的样式，创建渲染树；
- 从根节点递归调用，计算每一个元素的大小、位置等，给每个节点所应该出现在屏幕上的精确坐标；
- 如果body中的
  引用了图片资源，则立即向服务器发出请求，此时渲染引擎不会等待图片下载完毕，而是继续渲染后面的代码；
- 服务器返回图片文件，由于图片占用了一定面积，影响了后面段落的排版，因此引擎需要回过头来重新渲染这部分代码；
- 如果此时js脚本中运行了style.display=”none”，布局被改变，引擎也需要重新渲染这部分代码；
- 直到为止，页面渲染完毕。

3、重绘和回流

当渲染树中的一部分(或全部)因为元素的规模尺寸，布局，隐藏等改变而需要重新构建。这就称为回流。比如上面的img文件加载完成后就会引起回流，每个页面至少需要一次回流，就是在页面第一次加载的时候。

当渲染树中的一些元素需要更新属性，而这些属性只是影响元素的外观，风格，而不会影响布局的，比如background-color。则就叫称为重绘。

从上面可以看出，回流必将引起重绘，而重绘不一定会引起回流。

会引起重绘和回流的操作

- 添加、删除元素(回流+重绘)
- 隐藏元素，display:none(回流+重绘)，visibility:hidden(只重绘，不回流)
- 移动元素，比如改变top,left,transform的值，或者移动元素到另外一个父元素中。(重绘+回流)
- 对style的操作(对不同的属性操作，影响不一样)
- 还有一种是用户的操作，比如改变浏览器大小，改变浏览器的字体大小等(回流+重绘)

4、本地存储

本地存储最原始的方式就是 cookie,cookie 是存放在本地浏览器的一段文本，数据以键值对的形式保存，可以设置过期时间。 但是 cookie 不适合大量数据的存储，因为每请求一次页面，cookie 都会发送给服务器，这使得 cookie 速度很慢而且效率也不高。因此cookie的大小被限制为4k左右(不同浏览器可能不同,分HOST)，如下所示：

- Firefox和Safari允许cookie多达4097个字节，包括名（name）、值（value）和等号。
- Opera允许cookie多达4096个字节，包括：名（name）、值（value）和等号。
- Internet Explorer允许cookie多达4095个字节，包括：名（name）、值（value）和等号。

在所有浏览器中，任何cookie大小超过限制都被忽略，且永远不会被设置。

html5提供了两种在客户端存储数据的新方法：localStorage 和 sessionStorage, 它们都是以key/value的形式来存储数据，前者是永久存储，后者的存储期限仅限于浏览器会话(session),即当浏览器窗口关闭后，sessionStorage中的数据被清除。

localStorage的存储空间大约5M左右(不同浏览器可能不同，分 HOST)，这个相当于一个5M大小的前端页面的数据库，相比于cookie可以节约带宽，但localStorage在浏览器隐私模式下是不可读取的，当存储数据超过了localStorage的存储空间后会抛出异常。

此外，H5还提供了逆天的websql和indexedDB,允许前端以关系型数据库的方式来存储本地数据，相对来说，这个功能目前应用的场景比较少，此处不作介绍。

5、浏览器缓存机制

浏览器缓存机制是指通过 HTTP 协议头里的 Cache-Control（或 Expires）和 Last-Modified（或 Etag）等字段来控制文件缓存的机制。

Cache-Control 用于控制文件在本地缓存有效时长。最常见的，比如服务器回包：Cache-Control:max-age=600 表示文件在本地应该缓存，且有效时长是600秒（从发出请求算起）。在接下来600秒内，如果有请求这个资源，浏览器不会发出 HTTP 请求，而是直接使用本地缓存的文件。

Last-Modified 是标识文件在服务器上的最新更新时间。下次请求时，如果文件缓存过期，浏览器通过 If-Modified-Since 字段带上这个时间，发送给服务器，由服务器比较时间戳来判断文件是否有修改。如果没有修改，服务器返回304告诉浏览器继续使用缓存；如果有修改，则返回200，同时返回最新的文件。

Cache-Control 通常与 Last-Modified 一起使用。一个用于控制缓存有效时间，一个在缓存失效后，向服务查询是否有更新。

Cache-Control 还有一个同功能的字段：Expires。Expires 的值一个绝对的时间点，如：Expires: Thu, 10 Nov 2015 08:45:11 GMT，表示在这个时间点之前，缓存都是有效的。

Expires 是 HTTP1.0 标准中的字段，Cache-Control 是 HTTP1.1 标准中新加的字段，功能一样，都是控制缓存的有效时间。当这两个字段同时出现时，Cache-Control 是高优化级的。

Etag 也是和 Last-Modified 一样，对文件进行标识的字段。不同的是，Etag 的取值是一个对文件进行标识的特征字串。在向服务器查询文件是否有更新时，浏览器通过 If-None-Match 字段把特征字串发送给服务器，由服务器和文件最新特征字串进行匹配，来判断文件是否有更新。没有更新回包304，有更新回包200。Etag 和 Last-Modified 可根据需求使用一个或两个同时使用。两个同时使用时，只要满足基中一个条件，就认为文件没有更新。

另外有两种特殊的情况：

- 手动刷新页面（F5），浏览器会直接认为缓存已经过期（可能缓存还没有过期），在请求中加上字段：Cache-Control:max-age=0，发包向服务器查询是否有文件是否有更新。
- 强制刷新页面（Ctrl+F5），浏览器会直接忽略本地的缓存（有缓存也会认为本地没有缓存），在请求中加上字段：Cache-Control:no-cache（或 Pragma:no-cache），发包向服务重新拉取文件。

6、History路由机制

用户访问网页的历史记录通常会被保存在一个类似于栈对象中，即history对象，点击返回就出栈，跳下一页就入栈。 它提供了一些方法来操作页面的前进和后退：

- window.history.back() 返回到上一个页面
- window.history.forward() 进入到下一个页面
- window.history.go([delta]) 跳转到指定页面

HTML5 对History Api 进行了增强，新增了两个Api和一个事件，分别是pushState、replaceState 和 onpopstate

pushState是往history对象里添加一个新的历史记录，即压栈。

replaceState 是替换history对象中的当前历史。

当点击浏览器后退按钮或js调用history.back都会触发onpopstate事件, 与其类似的还有一个事件: onhashchange 。

onhashchange是老API, 浏览器支持度高, 本来是用来监听hash变化的, 但可以被利用来做客户端前进和后退事件的监听，onpopstate是专门用来监听浏览器前进后退的, 不仅可以支持hash, 非hash的同源url也支持。

7、HTML5离线缓存

HTML5离线缓存又叫Application Cache，是从浏览器的缓存中分出来的一块缓存区，如果要在这个缓存中保存数据，可以使用一个描述文件（manifest file），列出要下载和缓存的资源。

manifest 文件是简单的文本文件，它告知浏览器被缓存的内容（以及不缓存的内容）。manifest 文件可分为三个部分：

– CACHE MANIFEST – 在此标题下列出的文件将在首次下载后进行缓存

– NETWORK – 在此标题下列出的文件需要与服务器的连接，且不会被缓存

– FALLBACK – 在此标题下列出的文件规定当页面无法访问时的回退页面（比如 404 页面）

离线缓存为应用带来三个优势：

- 离线浏览 – 用户可在应用离线时使用它们
- 速度 – 已缓存资源加载得更快
- 减少服务器负载 – 浏览器将只从服务器下载更新过或更改过的资源。

8、Canvas和SVG

Canvas 通过Javascript 来绘制 2D 图形。Canvas 是逐像素进行渲染的。在 Canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。

SVG 是一种使用 XML 描述 2D 图形的语言。SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。你可以为某个元素附加 JavaScript 事件处理器。在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。

Canvas和SVG相比，canvas更依赖于分辨率，不支持事件处理器，文本渲染能力弱，比较适合密集型游戏，其中的许多对象会被频繁绘制，而svg则比较适用于类似谷歌地图带有大型渲染区域的应用程序。