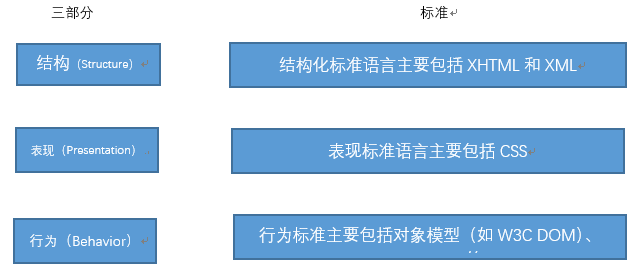
Web标准：

一系列标准的集合。网页主要由三部分组成：结构（Structure）、表现（Presentation）和行为（Behavior）。对应的标准也分三方面：结构化标准语言主要包括XHTML和XML，表现标准语言主要包括CSS，行为标准主要包括对象模型（如W3C DOM）、ECMAScript等）。



浏览器规范：

浏览器的主要功能是呈现你所选择的资源，通过向服务端请求并展示在浏览器窗口。

浏览器接受url开启一个网络请求线程

浏览器发出一个完整的http请求

服务器接收请求到后台接收请求

使用http请求请求页面

把请求回来的html代码解析成DOM树

CSS的可视化格式模型解析

根据CSS属性对元素进行渲染，得到内存中的位图

对位图的合成

绘制页面

浏览器使用流式布局模型 (Flow Based Layout)。

浏览器会把HTML解析成DOM，把CSS解析成CSSOM，DOM和CSSOM合并就产生了Render Tree。

有了RenderTree，我们就知道了所有节点的样式，然后计算他们在页面上的大小和位置，最后把节点绘制到页面上。

由于浏览器使用流式布局，对Render Tree的计算通常只需要遍历一次就可以完成，但table及其内部元素除外，他们可能需要多次计算，通常要花3倍于同等元素的时间，这也是为什么要避免使用table布局的原因之一。

移动端项目优化（H5）

目标：PC优化手段在Mobile端同样适用。除加载外渲染速度也是优化重点，合理处理代码减少渲染损耗，用户交互使用时也需注意性能

做法：减少HTTP请求（因为手机浏览器同时响应请求为4个请求，小于4个），使用缓存（减少向服务器的请求数），压缩HTML、CSS、Java（减少资源大小可以加快网页显示速度），减少Cookie，避免重定向以及异步加载第三方资源，按需加载，压缩图片

在 iPhone 6 上，屏幕宽度为 375px，共有 750 个物理像素，则 750rpx = 375px = 750 物理像素，1rpx = 0.5px = 1 物理像素。

**tcp如果没有第三次握手会怎么样**

假定不采用三次握手，那么只要B发出确认，新的连接就建立了。 由于现在A并没有发出建立连接的请求，因此不会理财B的确认，也不会向B发送数据。 但B却以为新的运输连接已经建立了，并一直等待A发来数据。 B的许多资源就这样浪费了

**有用过css的预处理语法吗**

Sass、Less和Stylus处理器

**webpack怎么把css文件单独分离出来**

单独打包css，在webpack需要使用一个插件，extract-text-webpack-plugin

**fectch和ajax区别，口述ajax代码，如何发送同步ajax？**

sync默认是true，即为异步方式，$.ajax执行后，会继续执行ajax后面的脚本，直到服务器端返回数据后，触发$.ajax里的success方法。若要将其设置为false，则所有的请求均为同步请求，在没有返回值之前，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行。

fetch和ajax 的主要区别

ajax代码

ajax的技术核心是 XMLHttpRequest 对象。

原理：ajax请求过程：创建XMLHttpRequest、连接服务器、发送请求、服务器做出响应、接收响应数据

var xmlHttp

function showHint(str) {

xmlHttp=GetXmlHttpObject()

url=url+"&sid="+Math.random();

xmlHttp.onreadystatechange=stateChanged;

xmlHttp.open("GET",url,true);

xmlHttp.send(null);

}

function stateChanged(){

if (xmlHttp.readyState==4) {

document.getElementById("txtHint").innerHTML=xmlHttp.responseText;

}

}

function GetXmlHttpObject(){

var xmlHttp=null;

try {

// Firefox, Opera 8.0+, Safari

xmlHttp=new XMLHttpRequest();

} catch (e)

{ // Internet Explorer

try{

xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");

}catch (e) {

xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");

}

}

return xmlHttp;}

1、fetch()返回的promise将不会拒绝http的错误状态，即使响应是一个HTTP 404或者500

2、在默认情况下 fetch不会接受或者发送cookies