HTML

1. **HTML**

HTML（超文本标记语言）是一种用于创建网页的标记语言。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，HTML使用标记标签来描述网页。

1. **空元素、元数据**

空元素：没有内容的HTML元素。在开始标签中关闭，即<br/><img/>都是空元素

元数据：描述数据的数据。使信息的描述和分类可以实现格式化，为机器处理创造了可能。

* 元数据元素 - head, title, base, link, meta, style

<base>元素为页面上的所有连接规定默认地址或默认目标（target）

<mata>元素提供关于HTML文档的元数据，元数据不会显示在页面上，但是对于机器是可读的。Meta元素被用于规定页面的描述、关键词、文档的作者、最后修改时间以及其他元数据。<meta>标签始终位于head元素中，元数据可用于浏览器（如何显示内容或重新加载页面），搜索引擎（关键词），或其他web服务。

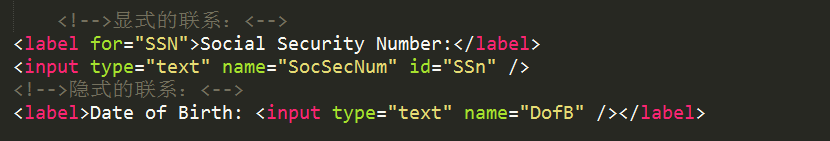
1. **Label for属性**

for 属性规定 label 与哪个表单元素绑定。

label 元素不会向用户呈现任何特殊效果。不过，它为鼠标用户改进了可用性。如果您在 label 元素内点击文本，就会触发此控件。就是说，当用户选择该标签时，浏览器就会自动将焦点转到和标签相关的表单控件上。

**隐式和显式的联系：**

标记通常以下面两种方式中的一种来和表单控件相联系：将表单控件作为标记标签的内容，这样的就是隐式形式，或者为 <label> 标签下的 for 属性命名一个目标表单 id，这样就是显式形式。



1. **Doctype作用？标准模式与兼容（怪异）模式各有什么区别?**

(Q1) <!DOCTYPE>告知浏览器的解析器用什么文档标准解析这个文档。DOCTYPE不存在或格式不正确会导致文档以兼容模式呈现。

(Q2) 标准模式的排版和JS运作模式都是以该浏览器支持的最高标准运行。在兼容模式中，页面以宽松的向后兼容的方式显示,模拟老式浏览器的行为以防止站点无法工作。

1. **HTML5 为什么只需要写 <!DOCTYPE HTML>？**

**HTML5不基于 SGML，因此不需要对DTD进行引用，**但是需要doctype来规范浏览器的行为（让浏览器按照它们应该的方式来运行）。

而HTML4.01基于SGML,所以需要对DTD进行引用，才能告知浏览器文档所使用的文档类型。

1. **em和rem的区别**

em相对**父级元素**设置的font-size来设置大小，如果父元素没有设置font-size ，则继续向上查找，直至有设置font-size元素

rem直接**参照html**标签字体大小，并且所有使用rem单位的都是参照html标签

1. **移动端你们一般采用什么布局？移动端设计稿是多大的尺寸？**

布局：采用viewport设置定宽布局

尺寸：640或750

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

viewport 是用户网页的可视区域。

* width：控制 viewport 的大小，可以指定的一个值，如 600，或者特殊的值，如 device-width 为设备的宽度（单位为缩放为 100% 时的 CSS 的像素）。
* height：和 width 相对应，指定高度。
* initial-scale：初始缩放比例，也即是当页面第一次 load 的时候缩放比例。
* maximum-scale：允许用户缩放到的最大比例。
* minimum-scale：允许用户缩放到的最小比例。
* user-scalable：用户是否可以手动缩放。

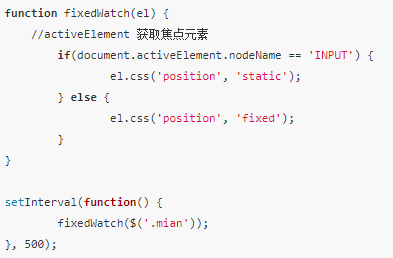
注意！！！

Initial-scale指的是将视口宽度缩放为理想宽度（CSS的100%宽度）的多少倍。

Width和initial-scale冲突时，选择值更大的。

如果设置的宽度》理想宽度，页面不能完整显示，需要水平拖动查看，若《理想宽度，即

1. **固定定位布局 键盘挡住输入框内容？**
2. 通过绑定窗口改变事件，监听键盘的弹出。然后去改变固定元素的位置。可以获取元素的offsetTop值，判断元素需要top=XXX，来移动元素
3. 通过定时器实时监听是否触发input。如果触发input框 就把固定定位，改变成静态定位。这样就会浏览器会总动把内容顶上去。



1. **移动端兼容问题**

###### 1、移动端默认的input和按钮样式还有点击出现的样式不一样，需要默认去清除一些默认样式

**//去掉webkit的滚动条——display: none;**

//其他参数

::-webkit-scrollbar //滚动条整体部分

::-webkit-scrollbar-thumb //滚动条内的小方块

::-webkit-scrollbar-track //滚动条轨道

::-webkit-scrollbar-button //滚动条轨道两端按钮

::-webkit-scrollbar-track-piece //滚动条中间部分，内置轨道

::-webkit-scrollbar-corner //边角，两个滚动条交汇处

::-webkit-resizer //两个滚动条的交汇处上用于通过拖动调整元素大小的小控件

**// 禁止长按链接与图片弹出菜单**

a,img { -webkit-touch-callout: none }

**// 禁止ios和android用户选中文字**

html,body {-webkit-user-select:none; user-select: none; }

**// 改变输入框placeholder的颜色值**

::-webkit-input-placeholder { /\* WebKit browsers \*/

color: #999; }

:-moz-placeholder { /\* Mozilla Firefox 4 to 18 \*/

color: #999; }

::-moz-placeholder { /\* Mozilla Firefox 19+ \*/

color: #999; }

:-ms-input-placeholder { /\* Internet Explorer 10+ \*/

color: #999; }

input:focus::-webkit-input-placeholder{ color:#999; }

**// android上去掉语音输入按钮**

input::-webkit-input-speech-button {display: none}

**// 阻止windows Phone的默认触摸事件**

/\*说明：winphone下默认触摸事件事件使用e.preventDefault是无效的，可通过样式来禁用，如：\*/

html { -ms-touch-action:none; } //禁止winphone默认触摸事件

**//链接问题:**

//ios用户点击一个链接，会出现一个半透明灰色遮罩, 如果想要禁用，可设置-webkit-tap-highlight-color的alpha值为0去除灰色半透明遮罩；

//android用户点击一个链接，会出现一个边框或者半透明灰色遮罩, 不同生产商定义出来额效果不一样，可设置-webkit-tap-highlight-color的alpha值为0去除部分机器自带的效果；

//winphone系统,点击标签产生的灰色半透明背景，能通过设置<meta name="msapplication-tap-highlight" content="no">去掉；

//特殊说明：有些机型去除不了，如小米2。对于按钮类还有个办法，不使用a或者input标签，直接用div标签

a,button,input,textarea {

-webkit-tap-highlight-color: rgba(0,0,0,0);

-webkit-user-modify:read-write-plaintext-only; //-webkit-user-modify有个副作用，就是输入法不再能够输入多个字符

}

// 也可以

\* { -webkit-tap-highlight-color: rgba(0,0,0,0); }

//winphone下

<meta name="msapplication-tap-highlight" content="no">

###### 2、手机拍照和上传图片

/\*

\*IOS有拍照、录像、选取本地图片功能，

\*部分Android只有选择本地图片功能。

\*Winphone不支持

\*/

<input type="file" accept="images/\*" />

###### 3、消除transition闪屏

.css {

-webkit-transform-style: preserve-3d;

-webkit-backface-visibility: hidden;

-webkit-perspective: 1000;

}

======一下内容与上无关=====

//目前，像Chrome/Filefox/Safari/IE9+以及最新版本Opera都支持硬件加速，当检测到某个DOM元素应用了某些CSS规则时就会自动开启，从而解决页面闪白，保证动画流畅。

.css {

-webkit-transform: translate3d(0,0,0);

-moz-transform: translate3d(0,0,0);

-ms-transform: translate3d(0,0,0);

transform: translate3d(0,0,0);

}

###### 4:固定定位相关问题？

1. 固定定位会产生 键盘弹出遮住固定元素里面元素的input框
2. 固定定位的遮罩层弹出时 滚动到最底部 qq和uc浏览器会出现遮罩层下面的样式露出来

###### 5: 在ios上写有邮箱文本点击这个邮箱地址不会有反应，安卓手机上点击邮箱会自动打开本地邮箱软件。

1. **移动端click 300ms延迟原因**

移动端浏览器会有一些默认的行为，比如双击缩放、双击滚动。这些行为，尤其是双击缩放，主要是为桌面网站在移动端的浏览体验设计的。而在用户对页面进行操作的时候，移动端浏览器会优先判断用户是否要触发默认的行为。

**github上有个fastclick插件，用来规避click事件的延时执行**

1. **本地存储**

本地存储分为两大类：

1. web storage：对HTML4中cookie机制改善。
   1. sessionStorage
   2. localStorage
2. 本地数据库：HTML5新增，可以在客户端本地建立一个数据库
   1. SQLLite：通过SQL语言来访问的文件型SQL数据库
   2. IndexedDB：NoSQL类型的数据库。完胜

indexedDB与localStorage存储对比

1. indexedDB存储IE10+支持，localStorage存储IE8+支持，后者兼容性好
2. localstorage适合存储简单数据结构，indexedDB存储比较适合键值对较多的数据
3. indexedDB存储可以在service workers中使用，localStorage貌似不可以。这就使得在进行PWA开发的时候，数据存储的技术选型落在了indexedDB存储上面。

总结下，如果是浏览器主窗体线程开发，同时存储数据结构简单，选择localstorage；如果数据结构复杂，同时对浏览器兼容性没有要求，使用indexedDB；如果是在Service Workers中开发应用，只能使用indexedDB。

1. **Web storage：sessionStorage和localStorage**

###### 概念

HTML5中的web storage包括两种存储方式：sessionStorage和localStorage。

**sessionStorage**：用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问，会话结束后数据也随之销毁（浏览器关闭销毁）。

**localStorage**：用于存储一个域名下的需要永久存在在本地的数据，这些数据可以被一直访问，直到这些数据被删除。

两者的区别：

1. 生命周期。
2. 同一浏览器的相同域名和端口的不同页面间可以共享相同的localStorage，（可用于标签页间通信）但是不同页面间不能共享sessionStorage。页面指顶级窗口，如果一个页面包含多个iframe且他们属于同源页面，那么他们之间是可以共享sessionStorage的。

方法：

#添加键值对： setItem(key,value);

#获取键值对： getItem(key);

#删除键值对： removeItem(key);

#清除所有键值对： clear();

#获取属性名称（键名称）： key(index);

#获取键值对的数量： length;

###### Web Storage 和 cookie 的异同点及优劣势

**相同点：**

* 1. 都用于存储用户数据
  2. 存储格式都是字符串
  3. 数据大小都有限制

**Cookie的限制：**

1. 大小限制：小于4kb
2. 浏览器限制站点在用户计算机上存储的cookie的数量，大多数浏览器只允许每个站点存储20个cookie；如果试图存储更多，则最旧的cookie会被丢弃
3. 有些浏览器对自身接收的来自所有站点的cookie的总数作限制，通常是300个
4. Cookie默认会随HTTP请求发送到后台，但是js、css、img等请求不需要cookie

**Web storage的优点：**

1. 相比cookie ，Web Storage 的优点主要表现在存储空间更大，可存储的内容更大。
2. cookie每次都随请求数据发送到服务器端，Web Storage不会和请求数据一同发送到服务器端，占用带宽更少。

**缺点：**

现在所有浏览器都支持 cookie 操作，而只有现在浏览器才支持 Web Storage 操作，如果需要兼容老旧浏览器，就不能使用 Web Storage。

###### Web Storage 事件

**storage事件**：当存储的数据发生变化时，会触发 storage 事件。不同于click类的事件会事件捕获和冒泡，storage 事件更像是一个通知，不可取消。**触发这个事件会调用同域下其他窗口的storage事件，不过触发storage的窗口（即当前窗口）不触发这个事件。sessionStorage的存储是无法通过storage事件获取的。**storage 的 event 对象的常用属性如下：

oldValue：更新前的值。如果该键为新增加，则这个属性为null。

newValue：更新后的值。如果该键被删除，则这个属性为null。

url：原始触发storage事件的那个网页的网址。

key：存储store的key名

###### Web Storage 使用场景

1. 存储一些不经常改动的、不敏感的数据，比如全国省市区县信息。
2. 存储一些不太重要的跟用户相关的数据，比如说用户的头像地址、主题颜色等，这些信息可以先存储在用户本地一份，便于快速呈现，等真正从服务器端读取成功后再更改头像地址，主题颜色。
3. 基于 storage 事件特点，Web Storage 还可以用于同域不同窗口间的通信。
4. **Cookie、localStorage、sessionStorage的区别**

| **特性** | **cookie** | **sessionStorage** | **localStorage** |
| --- | --- | --- | --- |
| 数据生命期 | 生成时就会被指定一个maxAge值，这就是cookie的生存周期，在这个周期内cookie有效，默认关闭浏览器失效 | 页面会话期间可用 | 除非数据被清除，否则一直存在 |
| 存放数据大小 | 4K左右（因为每次http请求都会携带cookie） | 一般5M或更大 | |
| 与服务器通信 | 由对服务器的请求来传递，每次都会携带在HTTP头中，如果使用cookie保存过多数据会带来性能问题 | 数据不是由每个服务器请求传递的，而是只有在请求时使用数据，不参与和服务器的通信 | |
| 易用性 | cookie需要自己封装setCookie，getCookie | 可以用源生接口，也可再次封装来对Object和Array有更好的支持 | |
| 共同点 | 都是保存在浏览器端，和服务器端的session机制不同 | | |

1. **语义化的理解与应用**

语义化是指：根据内容的结构，选择合适的标签进行代码的编写，便于开发者阅读和写出更优雅的代码，同时让浏览器的爬虫和机器很好地解析。

语义化有以下几方面作用：

1）在没有CSS的情况下，页面也能呈现出很好地内容结构、代码结构

2）有利于SEO，搜索引擎的爬虫是读不懂无语义的span和div的，和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息。例如：爬虫依赖于标签来确定上下文和各个关键字的权重；

3）方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；

4）便于团队开发和维护，语义化更具可读性，遵循这个标准，可以减少差异化。

语义化常体现在以下几方面：

1）不使用纯样式标签，如：b、font、u等，改用css设置；若是需要强调的文本，则可以包含在strong、em、mark等标签中

2）利用header、footer、nav、aside、article、section等进行页面布局，而不是仅通过div+css进行布局

3）使用audio、video、canvas、time定义声音、视频、图形、日期

4）ul应用于无序列表，ol应用于有序列表

5）每个input标签对应的说明文本使用label标签，并且通过为input设置id属性，在lable标签中设置for=inputld来让说明文本和相对应的input关联起来

1. **结构、表现、行为分离原则的理解与实践应用**

HTML结构层是网页最重要的基础，HTML标签给予内容含义；CSS表现层则是定义HTML的样式，该如何显示；JavaScript行为层为页面增加交互。

**分离的好处**：

（1）一个网页HTML是必需的，CSS和JavaScript都是可选项。分离有利于在不支持或禁用css、js的浏览器中还能正常显示基本的内容，实现平稳退化。

CSS表现层和JavaScript表现层分别归属于独自的.css和.js文件。

（2）结构、表现、行为分离使得关注点分离，各司其职更容易维护。例如样式改变仅需修改HTML中外部的修改csc文件即可

**实践应用：**：

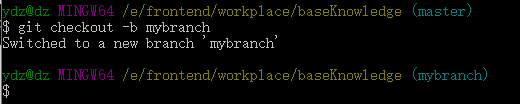
1. 分离表现与结构：
   1. html中不使用内联样式或内部样式表，使用外部样式表
   2. 不使用b（设置字体为粗体）、font（设置字体样式）等纯表现性的标签
2. 分离行为和结构：不把任何JavaScript代码写入HTML页面中
   1. 将JavaScript函数定义在一个分离的.js文件中，让所需的HTML页面连接到它；
   2. 删除所有的事件处理句柄（注：即行内的那些诸如onmouseover、onsubmit、onclick等）并归入同一.js文件中去，通过js获取元素添加绑定事件。

（3）分离表现与行为：

通过js设置、修改、删除元素的class来实现样式的更改，而不是直接通过js的style来修改样式，在样式表文件中编写不同class的样式代码

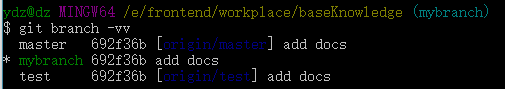
1. **Git常用指令**
2. 创建+切换本地分支

git checkout -b 分支名



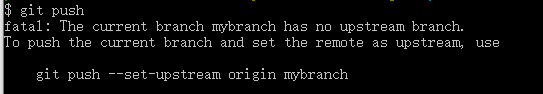
1. 查看当前的本地分支与远程分支的关联关系

git branch -vv



可以看出本地分支mybranch没有和远程分支相关联

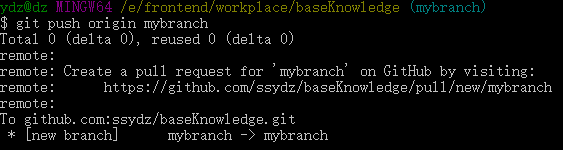
直接git push会报错



git push –set-upstream origin mybranch会创建一个远程分支origin/mybranch，并将本地的mybranch与其关联。

1. 将本地新建分支push到自己的本地远程origin上，因为只在本地创建了一个新分支，远程origin上还没有该分支，

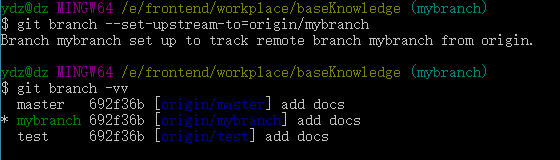
git push origin 远程分支名



此时创建了远程分支mybranch，但是本地分支和远程分支尚未关联

1. 将本地分支与远程origin的分支关联处理（通过 --set-upstream-to 命令）

git branch --set-upstream-to=origin/远程分支名



本地分支mybranch与远程分支orgin/mybranch已经关联，再使用git pull或git push时，无需再手动指定远程分支

git push origin master（手动指定）

git pull origin master（手动指定）