1：

行内元素：a,span,i,b,small,em,strong,img,br,button,input,label select,textarea任一几个就行

块级元素：div,h1,h2,h3,h4,h5,h6,table,ul,ol,dl,form

2：

src：指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求src资源时会将其指向的资源下载并应用到文档内

href：指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，用于超链接，当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕

3：

两者都是代表单位

px：值是固定的，绝对单位

em:值不是固定的，相对单位，跟父级单位设定或者浏览器默认字体设置存在关联

4：

共同点：用于浏览器端存储的缓存数据

不同点：

1：存储内容是否发送到服务器端：当设置了Cookie后，数据会发送到服务器端，造成一定的宽带浪费。 sessionStorage和localStorage会将数据保存到本地，不会造成宽带浪费

2：数据存储大小不同：Cookie数据不能超过4K,适用于会话标识sessionStorage和localStorage数据存储可以达到5M

3：数据存储的有效期限不同：cookie只在设置了Cookid过期时间之前一直有效，即使关闭窗口或者浏览器，sessionStorage,仅在关闭浏览器之前有效；localStorage,数据存储永久有效

4：作用域不同：cookie和localStorage是在同源同窗口中都是共享的，sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；

5：sessionStorage和localStorage有更多丰富易用的接口

5：

1：形成一个独立的空间，不影响外部也不受外部影响

2：函数中返回一个函数

6：

1：对 CommonJS 、 AMD 、ES6的语法做了兼容

2：对js、css、图片等资源文件都支持打包

3：串联式模块加载器以及插件机制，让其具有更好的灵活性和扩展性

4：有独立的配置文件webpack.config.js

5：可以将代码切割成不同的chunk，实现按需加载，降低了初始化时间

6：支持 SourceUrls 和 SourceMaps，易于调试

7：具有强大的Plugin接口，大多是内部插件，使用起来比较灵活

8：webpack 使用异步 IO 并具有多级缓存。这使得 webpack 很快且在增量编译上更加快

7：

opacity:元素本身依然占据它自己的位置并对网页的布局起作用。它也将响应用户交互

visibility:与 opacity 唯一不同的是它不会响应任何用户交互

display 设为 none 任何对该元素直接打用户交互操作都不可能生效。此外，读屏软件也不会读到元素的内容。这种方式产生的效果就像元素完全不存在

8:

1：link属于HTML标签，而@import是CSS提供的;

2：页面被加载的时，link会同时被加载，而@import被引用的CSS会等到引用它的CSS文件被加载完再加载;

3：import只在IE5以上才能识别，而link是HTML标签，无兼容问题;

4： link方式的样式的权重 高于@import的权重.

9:

Object、number、function、boolean、underfind

10：

1：表现方式一个在url中可以看到相关参数，一个是看不到的

2：发送数据大小方面get数据有大小限制

3：安全方面一个是一般请求，一个涉及到保密性，post方式不会直接暴露，相对更加安全一点

11：

在于参数的不同，call可以接受多个参数，apply只接受2个参数，apply的第二个参数只能为数组，call没限制

12：

利用事件冒泡的原理，让自己的所触发的事件，冒泡到他的父元素代替执行

13：

static

14:

split是分割字符串，join是将数组链接成字符串

15：

==：值相等就行，不严格判断

===：值相等且对象是同一个对象，严格相等

16：

1：ie6不支持png透明

2：ie6和ie7不支持position:fixed

3：图片默认间距问题

4：浏览器默认内外间距不一致问题

5：设置较小高度时出现高度超过所设高度问题

17：

1:递归调用没有设置退出条件

2：出现全局变量的情况

3：大消耗操作

4：频繁大量操作dom文档

5：没有及时回收相关变量

18：

19：

1：原型模式，原型继承机制

2：工厂模式，Object（）构造不同对象

3：混入模式，object设置不同属性

4：立即执行模式：隔离变量且初始化执行

20：

View 传送指令到 Controller

Controller 完成业务逻辑后，要求 Model 改变状态

Model 将新的数据发送到 View，用户得到反馈，所有通信都是单向的

Angular它采用双向绑定（data-binding）：View的变动，自动反映在 ViewModel，反之亦然。

组成部分Model、View、ViewModel

View：UI界面ViewModel：它是View的抽象，负责View与Model之间信息转换，将View的Command传送到Model；Model：数据访问层