# 1. <!DOCTYPE>声明是什么？有什么作用

* 声明文档类型
* 告诉浏览以兼容/标准模式解析文档
  + 标准：使用最新标准解析
  + 兼容：模拟老式浏览器

# 2. 为什么只写<!DOCTYPE HTML>

* H5以后不再基于SGML(标准通用标记语言)，故不再需要声明DTD(文档类型定义)

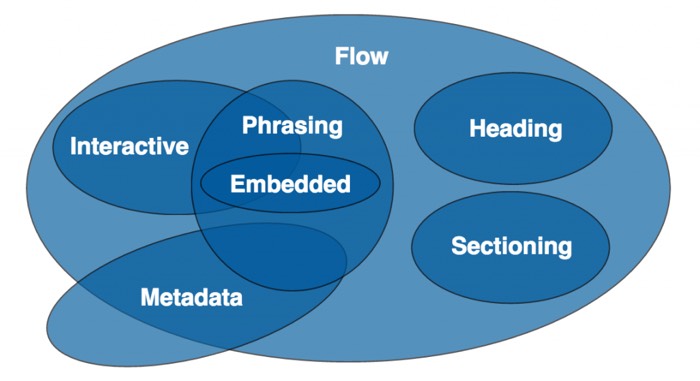
# 3. SGML/HTML/XML/XHTML

|  |  |
| --- | --- |
| 标记语言 | 区别/作用 |
| SGML(标准通用标记语言) | 起源 |
| HTML(超文本标记语言) | 规定如何显示网页，标签固定有限 |
| XML(可扩展标记语言) | 未来标记语言发展方向，标签可自定义无限 |
| XHTML(可扩展超文本标记) | 作为XML的应用重新定义HTML，更纯粹更严格的HTML(标签都要闭合，标签名都要小写等) |

# 4. 行内元素和块级元素

|  |  |  |
| --- | --- | --- |
| 两大元素 | 举例 | 区别 |
| 行内 | a span img strong button input label select textarea | 不换行，(通常情况下)只能包含文本和行内元素，width和height无效，margin和padding不影响其他元素 |
| 块级 | div ul ol li dl dt dd h p | 换行，可包含块级元素， |

# 5. 为什么有时候a能包含div

* 涉及H5的规范
* H5规定，元素有两种特性
  + 种类：
    - Metadata(改变页面行为和显示的link script)
    - Flow(放在body内部的)
    - Sectioning(改变页面结构的section nav aside)
    - Heading(h家族)
    - Phrasing （可以放在p元素中构成段落的，audio,img等，约等于h4中的行内)
    - Embedded(嵌入外部资源的iframe canvas video audio等)
    - Interactive(会发生交互的，a input textarea等)
  + 
  + 种类之间存在嵌套包含关系
  + 内容模型（何种子元素为何种类型时合法）
    - 特殊的一种：transprant(透明)取决于父级元素的内容模型，a的内容模型为transprant
* 综上，鉴于a标签内容模型为透明，故当父级元素能够容纳div时，a即可容纳div

# 6. 空元素是什么，有哪些

* 不包含内容，从开始标签就关闭
* br hr img input link meta

# 7. link 和 @import 的区别

* link可引入文件不止CSS
* @import 是CSS3的语法
* 加载顺序：link 引入的css将同时加载，而@import则会在页面加载完成后加载

# 8. 什么是浏览器

浏览器主要负责从服务器请求资源，然后渲染呈现web页面内容，包括：

* Shell ：浏览器外壳（界面
* 内核 ：基于标记语言展示内容

# 9. 浏览器内核的理解

浏览内核分为渲染引擎和 JS 引擎

* 渲染引擎 ：负责渲染html, xml, pdf等文档
* jS 引擎：负责解析执行 JS 脚本

# 10. 浏览器渲染原理

1. 解析文档，生成DOM树
2. 解析CSS，生成CSSOM树
3. 根据DOM树和CSSOM树，生成渲染树，其中节点称为，渲染节点
4. 每个渲染节点被加入渲染树时，浏览器对页面布局进行排列，称为回流或自动重排
5. 布局结束后，调用渲染节点的paint方法进行绘制

# 11. 渲染过程中遇到JS脚本的解析方式(async defer)

* 不使用async defer关键字时：阻塞文档解析，先对JS脚本进行解析执行，再继续文档解析
* 使用async：文档解析与JS脚本解析同步进行，异步执行JS脚本内容
* 使用defer：文档解析与JS脚本解析同步进行，待文档解析完成后（DOMContentLoaded) 执行JS脚本内容

# 12. 什么是文档的预解析

* 当浏览器执行 JS 脚本时，另一个线程继续对文档进行解析，并通过网络加载外部资源
* 预解析仅解析外部脚本，样式，不涉及DOM树

# 13. 浏览器渲染过程中的问题

* FOUC：样式闪烁
  + 现象：先显示了HTML但没有加载样式
  + 原因：CSS加载时间太长，或被放在了底部
* 白屏：不显示
  + 原因：
    - 部分浏览器渲染时要先构建DOM树和CSSOM树，构建完成才能渲染，如果CSS放在底部就会造成浏览器迟迟不渲染
    - js文件在头部，阻塞文档解析

# 14. 如何优化渲染关键路径

* 减少关键资源数量（关键资源指会阻止网页首次渲染的资源
* 优化关键字节数，减少下载时间
* 优化加载顺序

# 15. 什么是重绘和回流

* 重绘：更新渲染树中元素的一些属性，这些属性只改变元素的颜色、外观等
* 回流：渲染树中的对象改变了尺寸、布局、隐藏等，需要进行重新布局排列
* 触发回流的常见操作：
  + 改变大小
  + 添加，删除或操作DOM
  + 内容变化，在input中输入
  + 操作class内容
  + 激活伪类
  + offset相关属性的计算
  + 设置内联样式

# 16. 如何减少回流

* 用visibility 代替display:none
* 不要在循环里获取节点属性，如获取offset系列，每次都会回流
* 尽量不使用table布局，小改动会造成整个table重绘
* 把dom离线后操作，使用documentFragment对象在内存里操作
* 使用预先定义好的class替换，不要一条条修改css

# 17. H5的新元素新特性

* 新元素
  + 绘画canvas
  + 媒介audio video
  + 语义化更好的内容元素article section footer header
  + 表单控件 calendar date time email url search
* 新特性
  + 本地存储localstroage
  + sessionStroage
  + webworker
  + websocket

# 18. 语义化理解

* 用正确的标签做正确的事
* 使页面结构更清晰
* 有利于搜索引擎

# 19. seo注意事项

seo: 搜索引擎优化

* 合理title
* 语义化标签
* 重要内容放前面
* 重要内容不用js输出
* 少用iframe
* 非装饰性图片要有alt

# 20. 什么是WebSocket

WebSocket协议为web应用程序客户端和服务端之间提供了一种全双工通信机制。

特点：

* 握手阶段采用 HTTP 协议，默认端口是80和443
* 建立在 TCP 协议基础之上，和 http 协议同属于应用层
* 可以发送文本，也可以发送二进制数据。
* 没有同源限制，客户端可以与任意服务器通信。
* 协议标识符是 ws（如果加密，为 wss），如 ws://localhost:8023

# 21. 什么是WebWorkers

Web Worker 的作用，就是为 JavaScript **创造多线程环境**，允许主线程创建 Worker 线程，将一些任务分配给后者运行。在主线程运行的同时，Worker 线程在后台运行，两者互不干扰。等到 Worker 线程完成计算任务，再把结果返回给主线程。