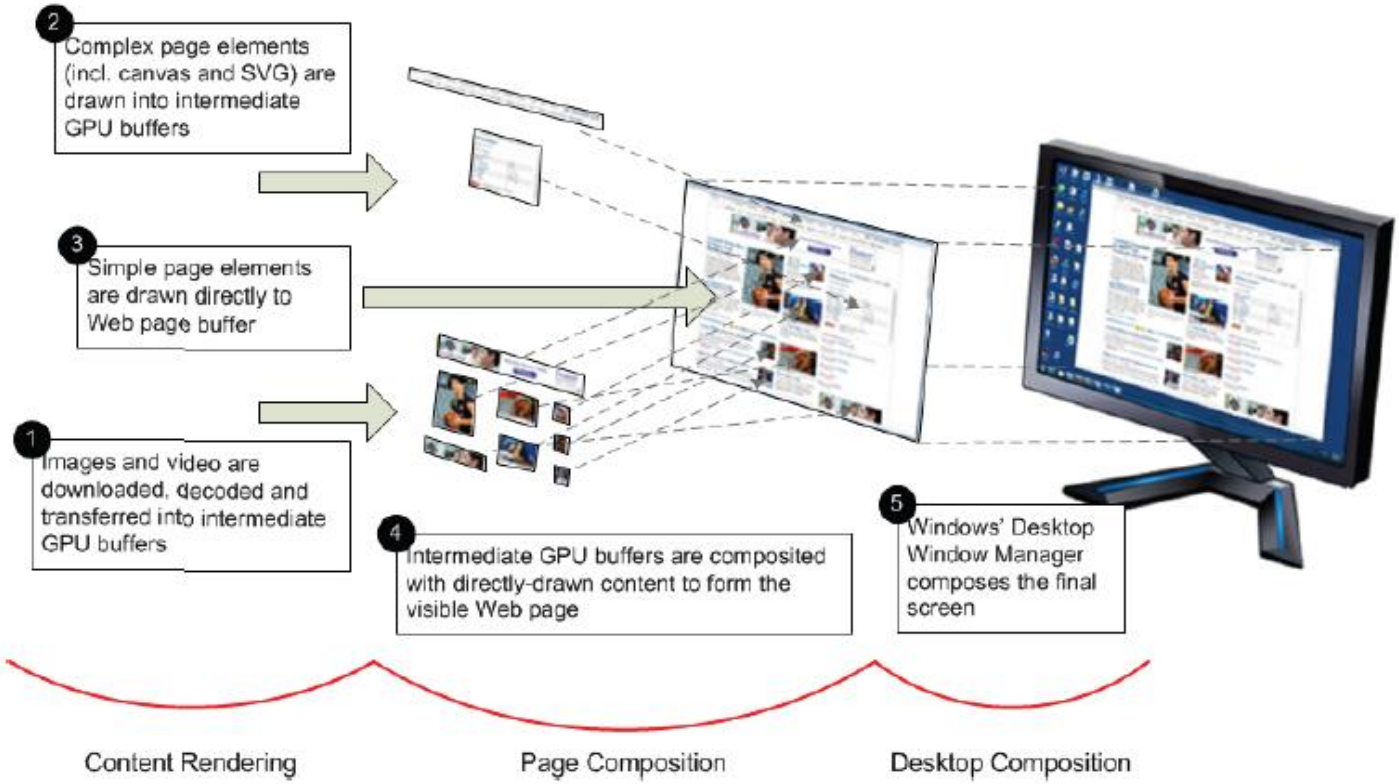
浏览器工作原理

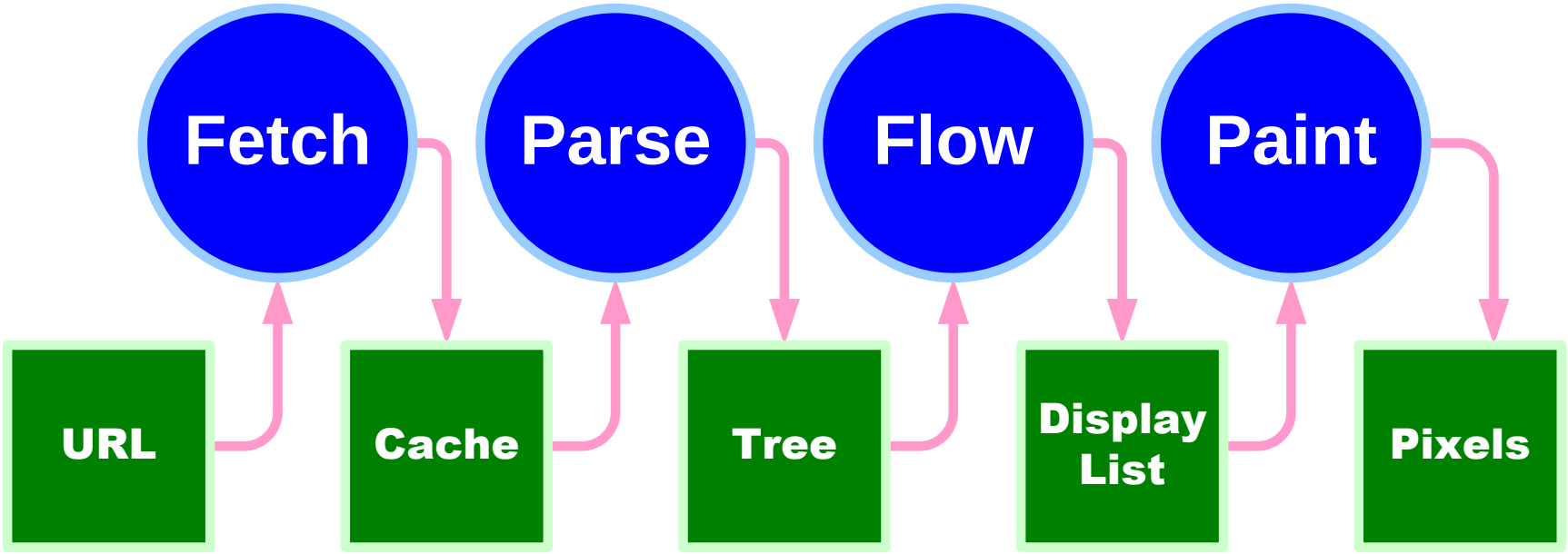
lilyH/李智杰 @奇舞团

浏览器

- Load
- Parse
- Render
- Layout
- Paint

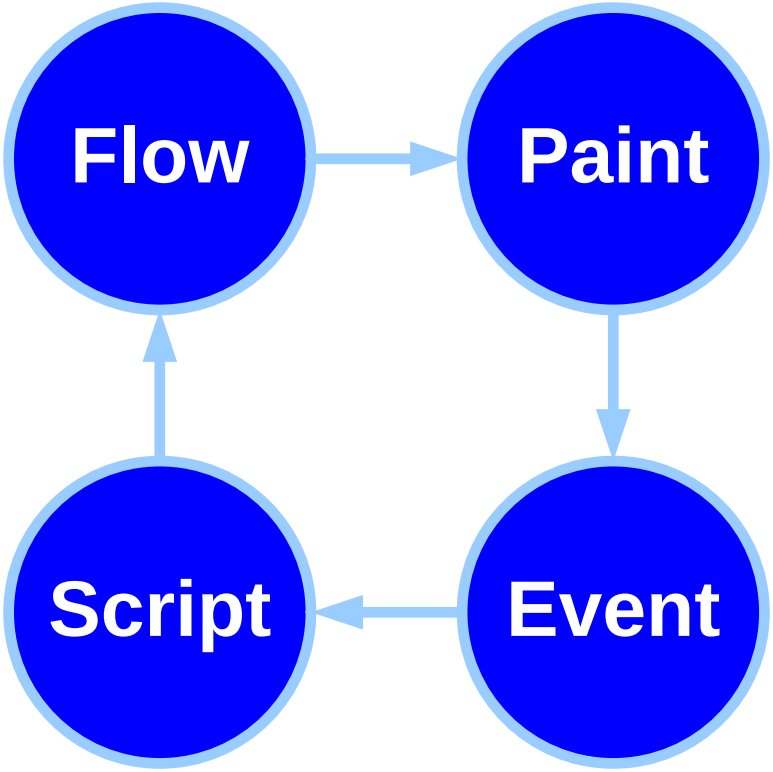


浏览器



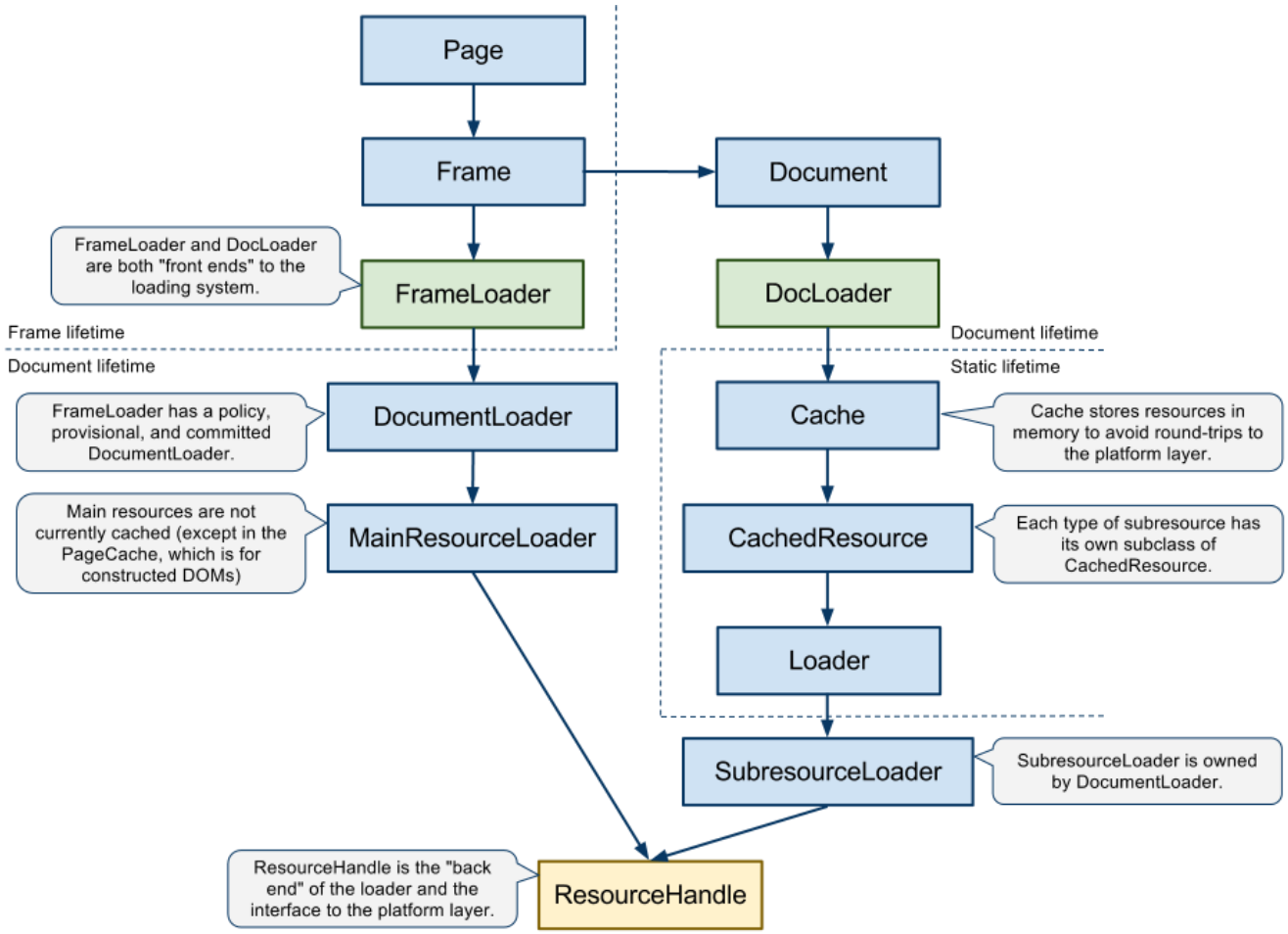
** 来自 Douglas Crockford's Lecture on "Theory of DOM"*

浏览器

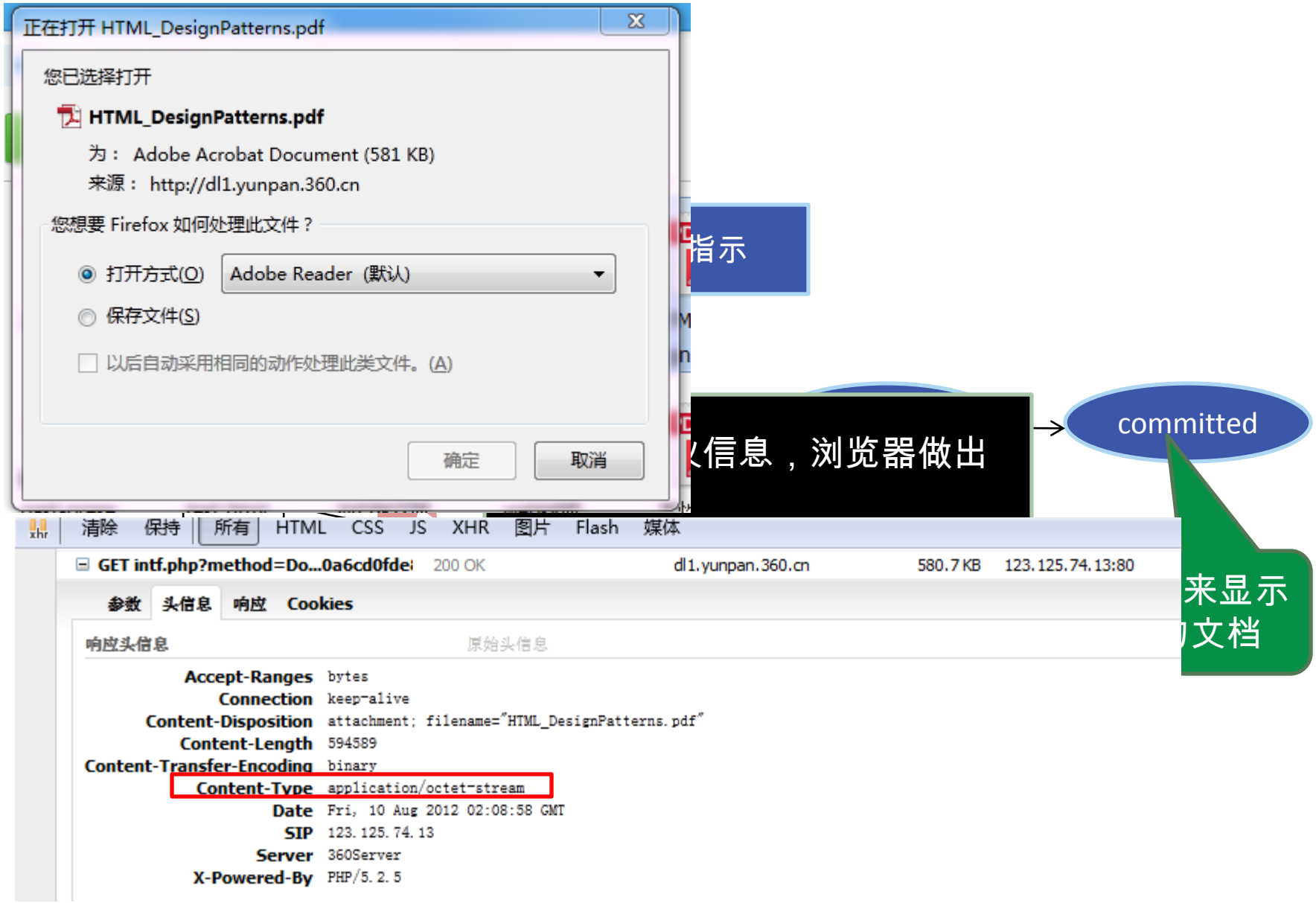


浏览器
事件驱动，
单线程，
异步编程模型

加载

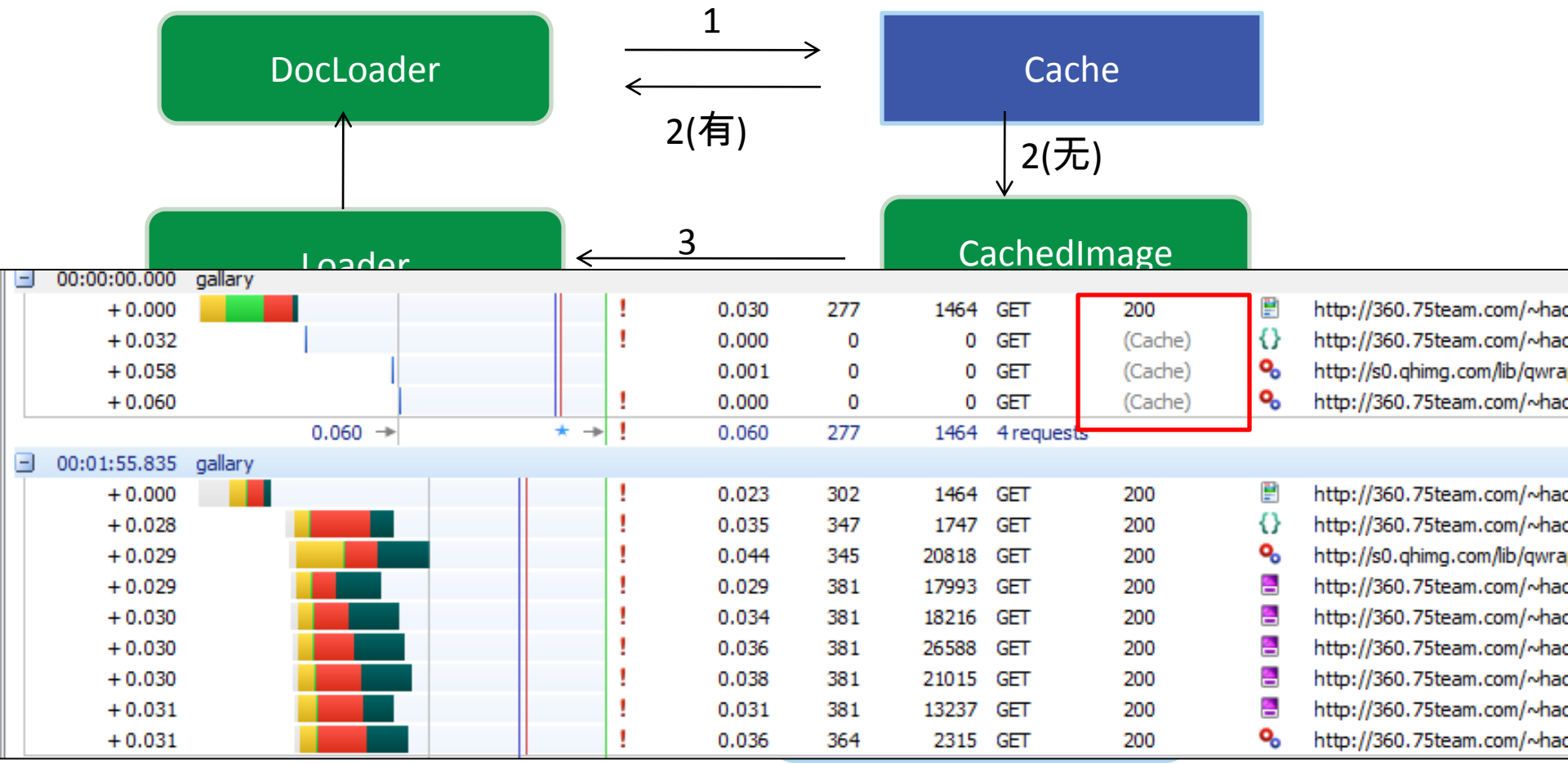


** 来自 Webkit blog “how webkit loads a web page”*



Webkit 框架文件加载流程

加载资源



Webkit 资源文件加载流程

<script></script>

- <script src>标签尽量放在底部（css <link>尽量放在页面顶部）
- 打包压缩script文件
- 尽量减少script的数量
- Cache

*来自Asynchronous and deferred JavaScript execution explained

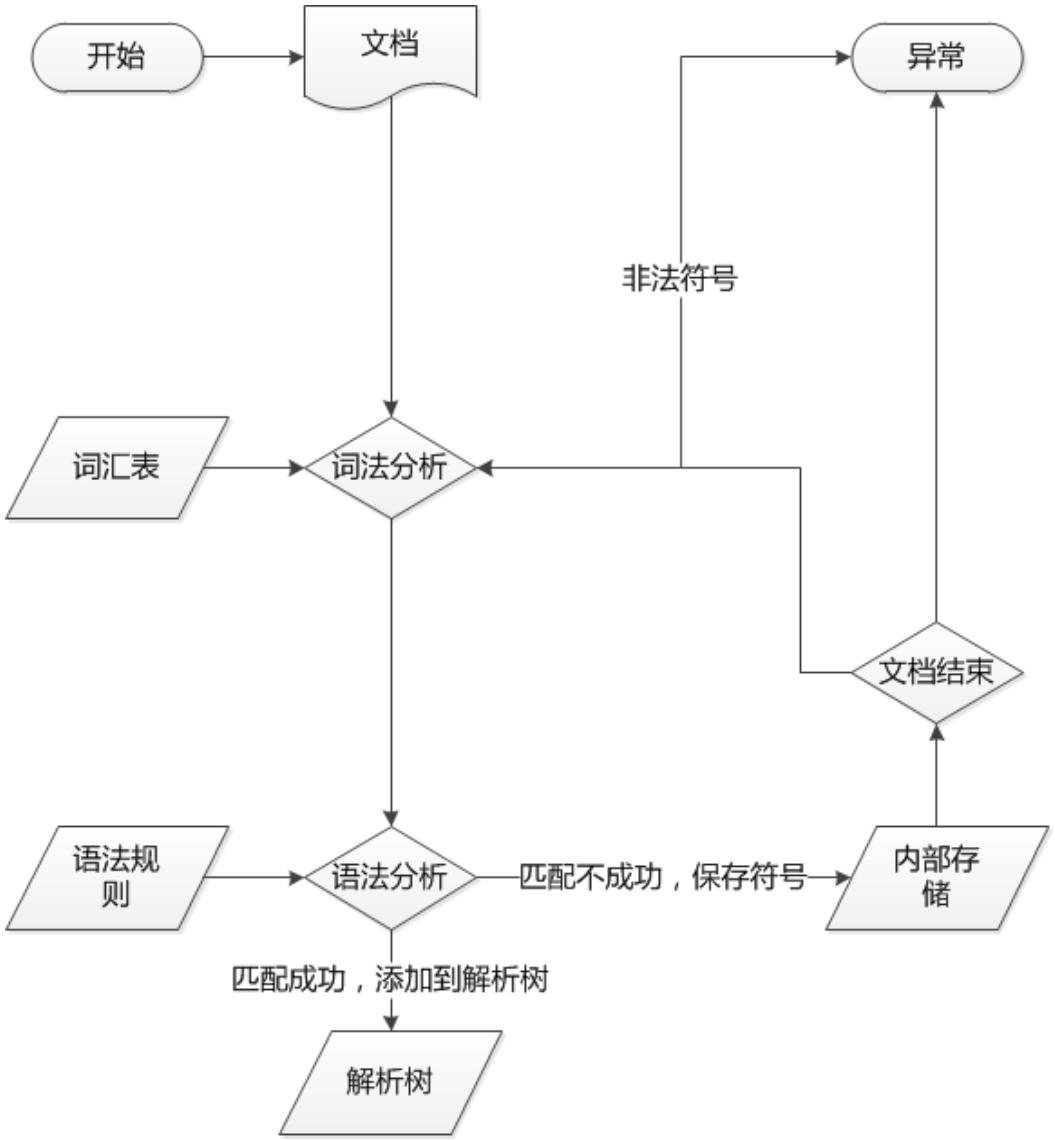
document.write()

- 允许JavaScript创建HTML文本
- onload前：插入HTML的文本到文档中
- onload后：替换文档中HTML的文本
- 会把输出写入到脚本文档所在的位置，浏览器解析完documemt.write()所在文档内容后，继续解析document.write()输出的内容，然后再继续解析HTML文档

HTML解析

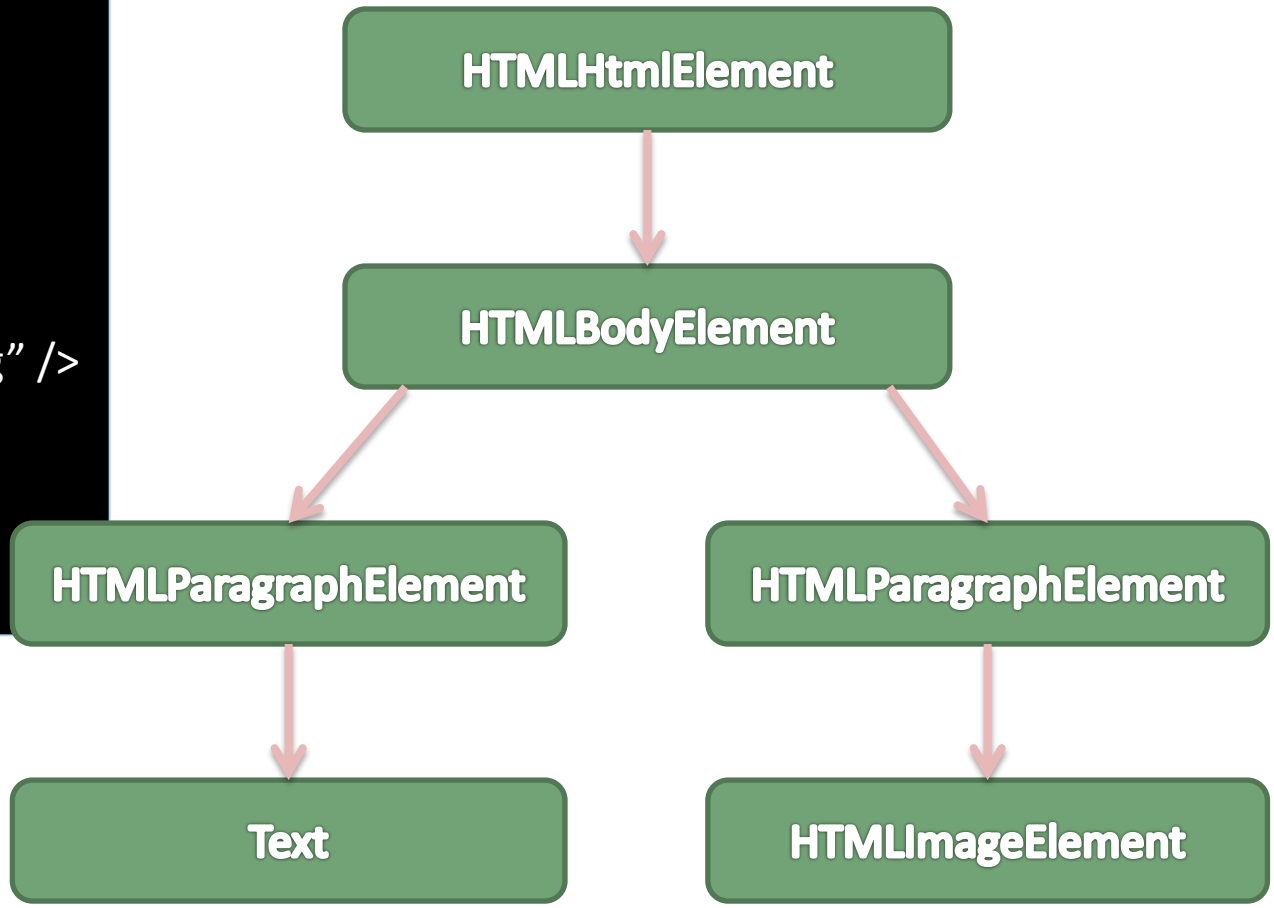
非上下文无关文法

从源文档到解析树


开始
文档
异常
非法符号
词汇表
词法分析
文档结束
语法规则
语法分析
匹配不成功，保存符号
内部存储
匹配成功，添加到解析树
解析树

HTML解析树——DOM树

```
<html>
  <body>
    <p>
      Hello DOM
    </p>
    <div>
    <img src="example.png" />
    </div>
  </body>
</html>
```



DTD定义了HTML的解析语法

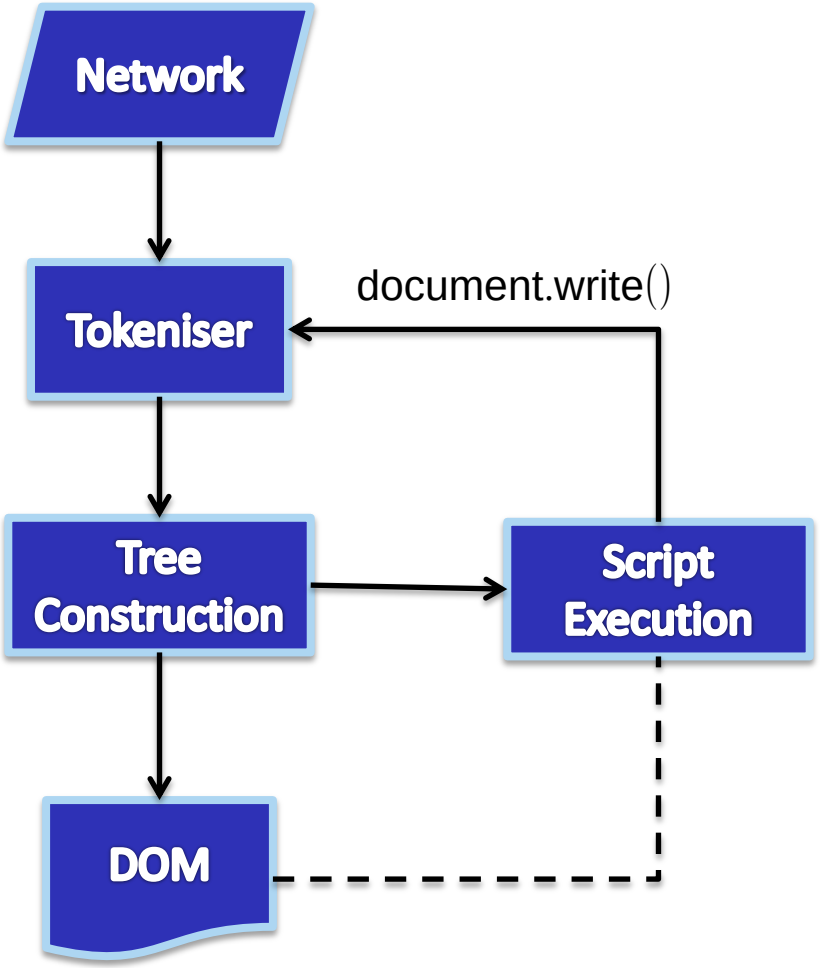
专属解析器

hmtl不能被一般的自顶向下或自底向上的解析器所解析。
原因是：

1. 这门语言本身的宽容特性

2. 浏览器对一些常见的非法html有容错机制

3. 解析过程是往复的，通常源码不会在解析过程中发生改变，但在html中，脚本标签包含的“document.write ”可能添加标签，这说明在解析过程中实际上修改了输入。

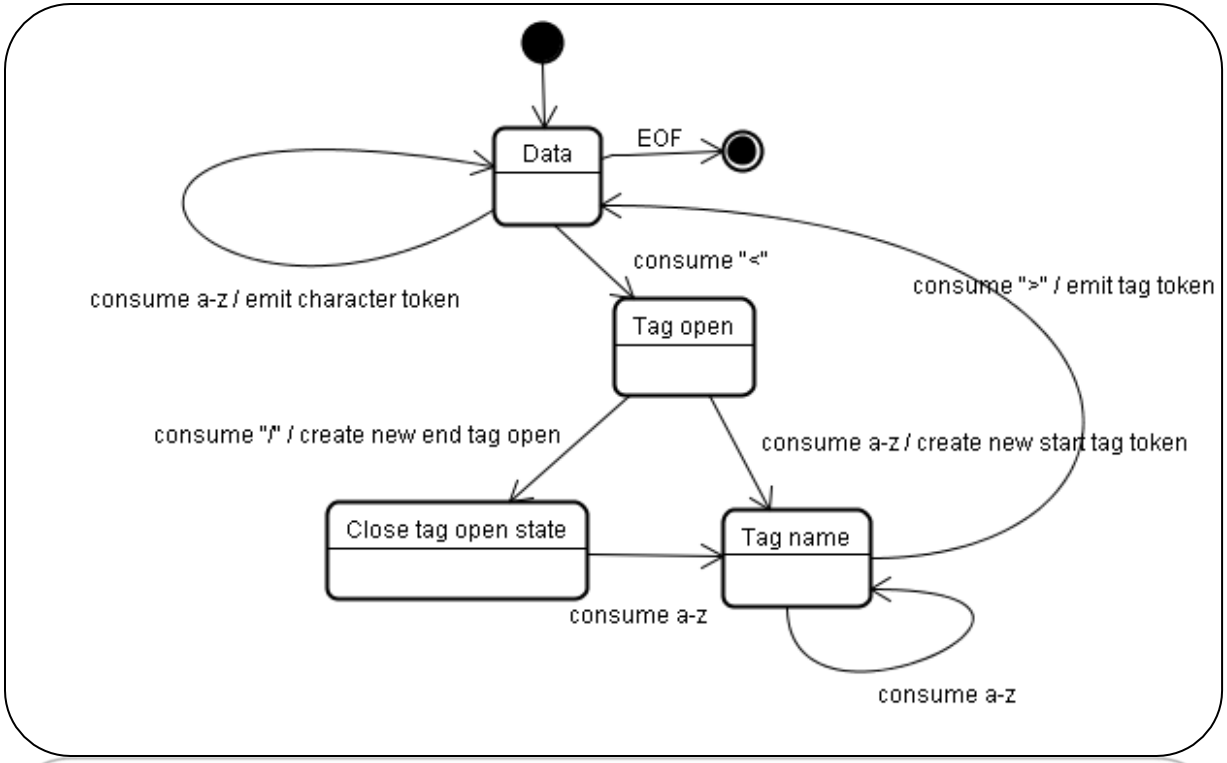


HTML解析流程

符号化：词法分析

基本示例——符号化下面的html：

```
<html>
    <body>
        Hello world
    </body>
</html>
```



符号化是词法分析的过程，将输入解析为符号，html的符号包括开始标签、结束标签、属性名及属性值

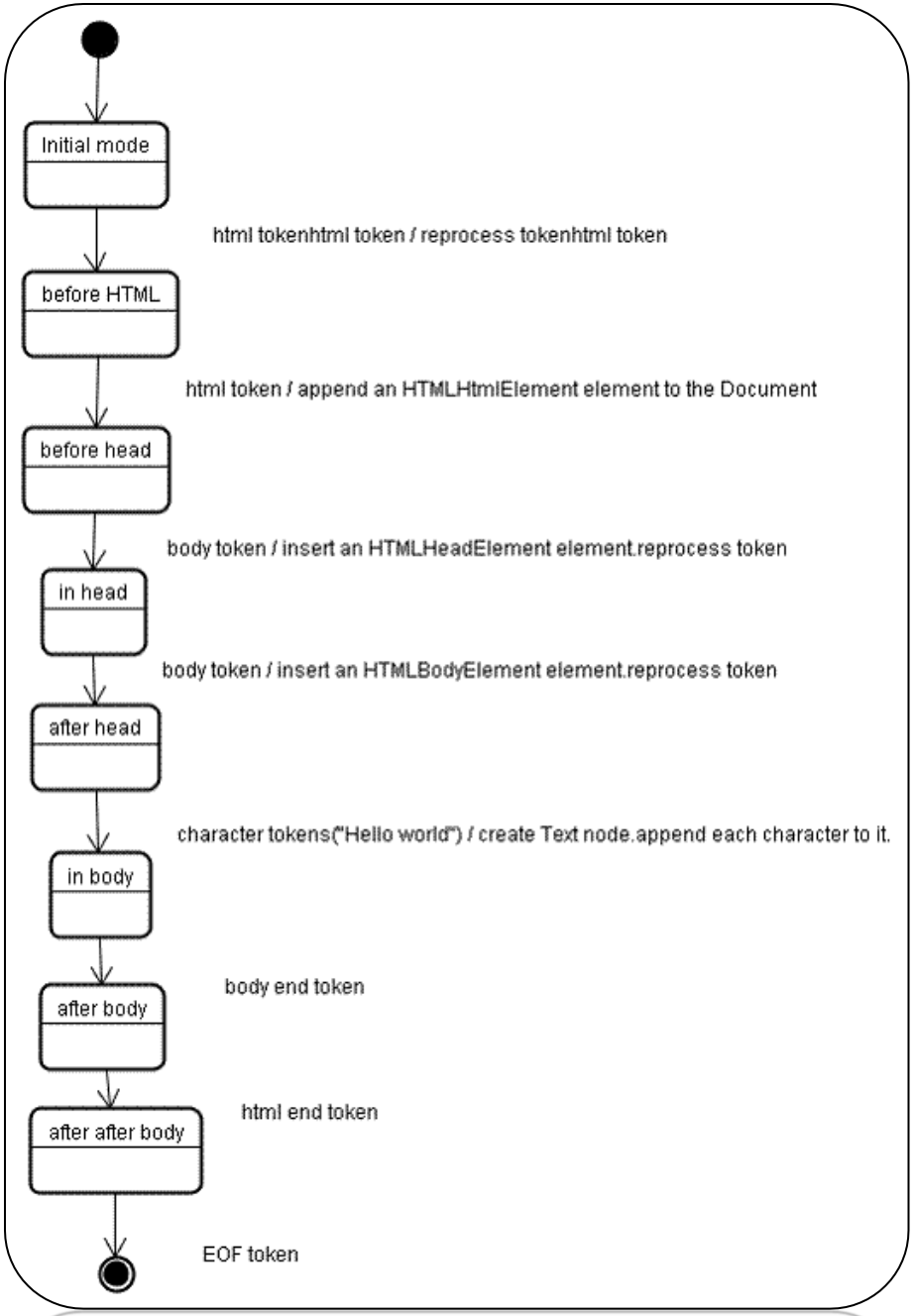
构建树：语法分析

来看一下示例中树的创建过程：

```
<html>
    <body>
        Hello world
    </body>
</html>
```

解析结束时的处理

Dom.ready()



浏览器容错

以下面这段html为例：

```
<html>
    <mytag></mytag>
    <div>
    <p>
    </div>
        Really lousy HTML
     </p>
     <div>
</html>
```

容错的例子：

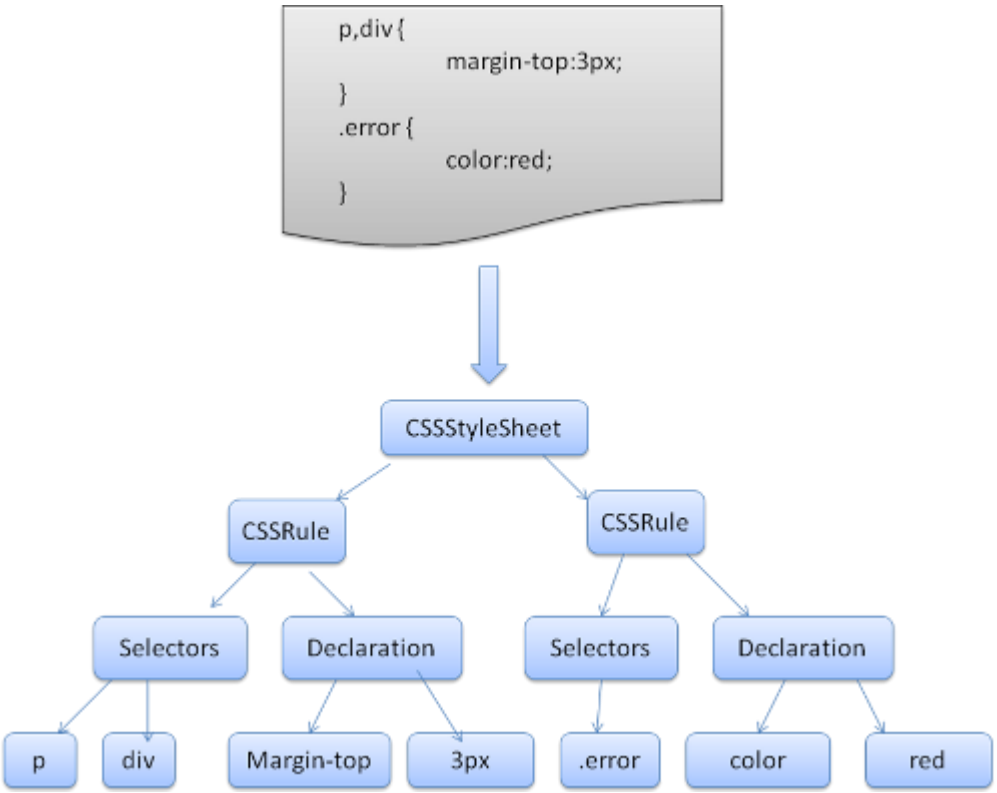
- 标签未关闭
- 标签嵌套错误
- 标签错误
- 遗漏标签 ,<head> <title> <body>
- 太深的标签继承，最多只允许20个相同类型的标签嵌套。

CSS解析

上下文无关文法

CSS解析器

- Webkit使用自底向上的解析器，Gecko使用自顶向下的解析器
- 它们都是将每个css文件解析为样式表对象，每个对象包含css规则，css规则对象包含选择器和声明对象，以及其他一些符合css语法的对象。



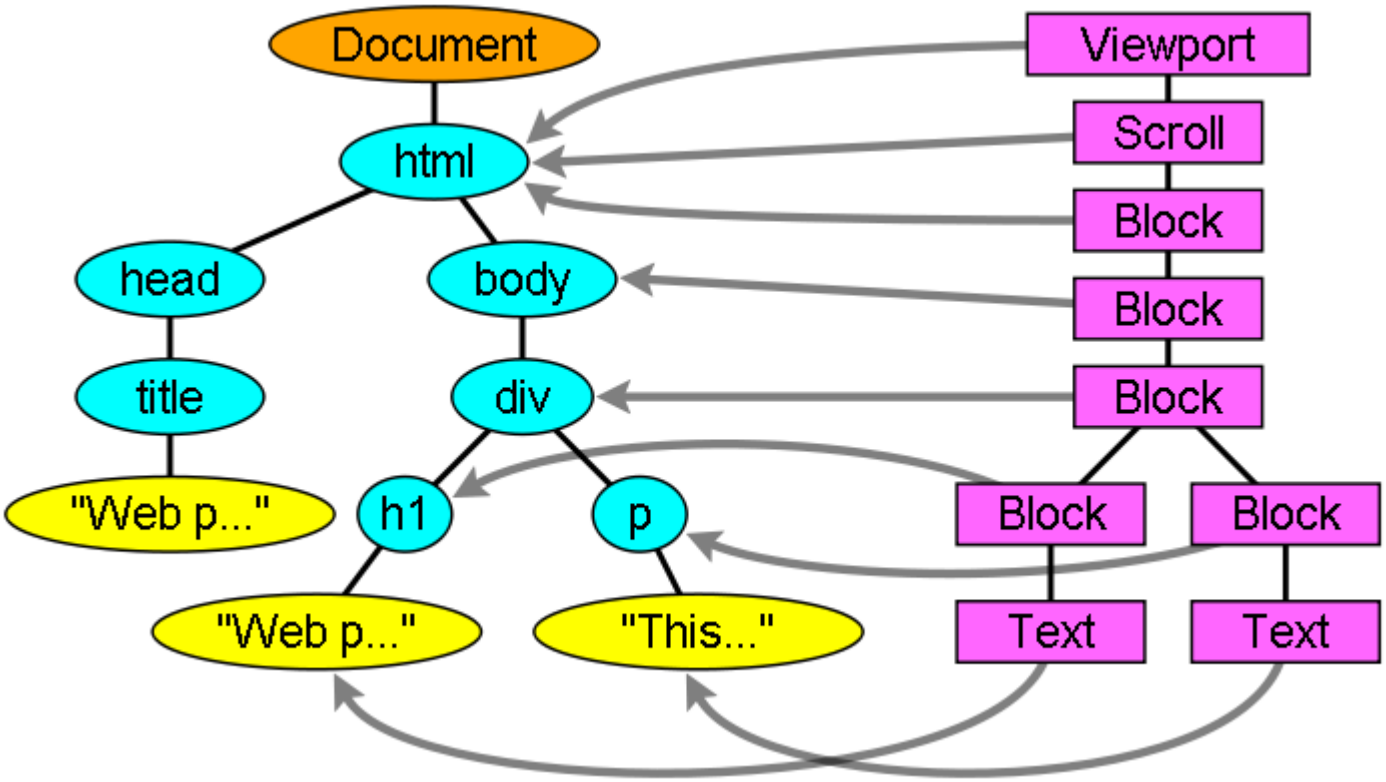
渲染树的构造

渲染引擎

- 渲染对象和DOM元素相对应，但这种对应关系**不是一对一**的，**不可见的Dom元素不会被插入渲染树**，例如head元素。另外，**display属性为none的元素也不会在渲染树中出现**（visibility属性为hidden的元素将出现在渲染树中）。
- 当文本因为宽度不够而**折行**时，新行将作为额外的渲染元素被添加。
- 根据css规范，一个行内元素只能仅包含行内元素或仅包含块状元素，在存在混合内容时，将会**创建匿名的块状渲染对象包裹住行内元素。**

渲染树和DOM树的关系

float 和 **absolute** 的元素，在两棵树上的位置不同，渲染树上标识出真实的结构，并用一个**占位结构**标识出它们原来的位置

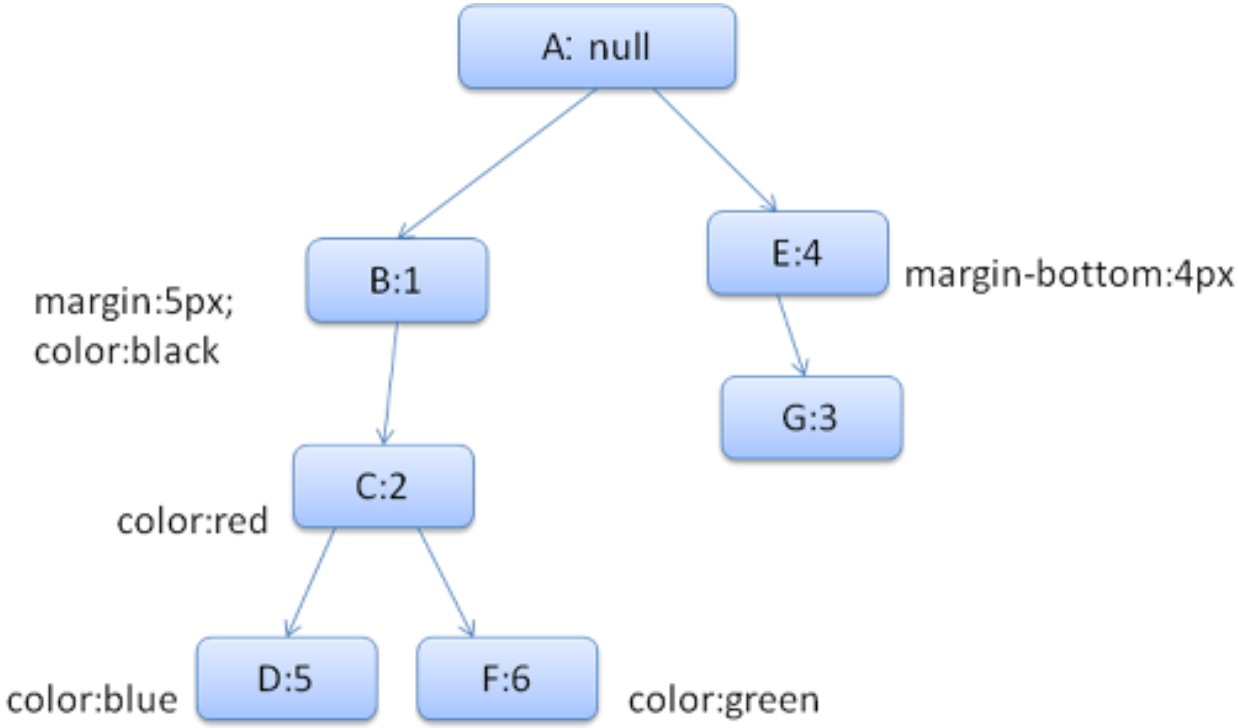


假设有下面这段html

```
<html>
 <body>
  <div class="err" id="div1">
    <p>this is a
      <span class="big">
            big error
      </span>
      this is also a
      <span class="big">
            very  big  error
      </span>
      error
    </p>
  </div>
  <div class="err" id="div2">
    another error
  </div>
 </body>
</html>
```

有下面这些规则

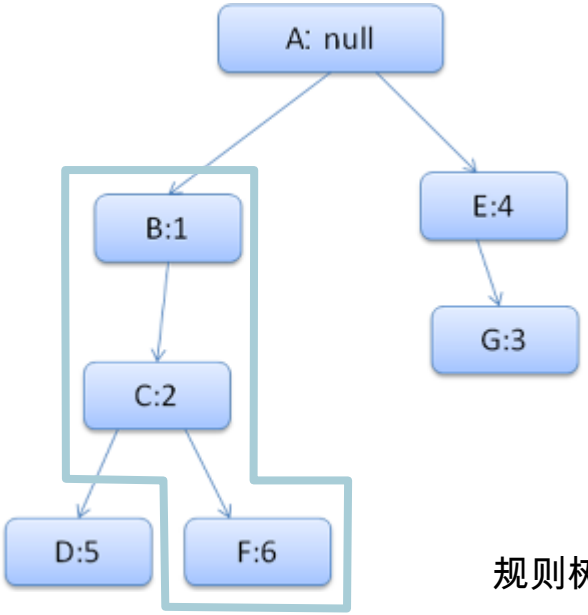
1. div {margin:5px;color:black}
2. .err {color:red}
3. .big {margin-top:3px}
4. div span {margin-bottom:4px}
5. #div1 {color:blue}
6. #div2 {color:green}



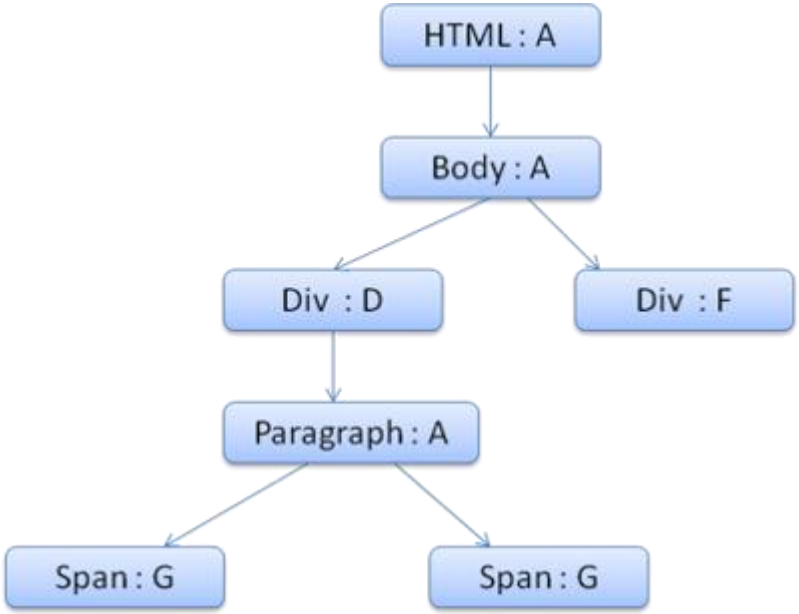
规则树

缓存例子

another error



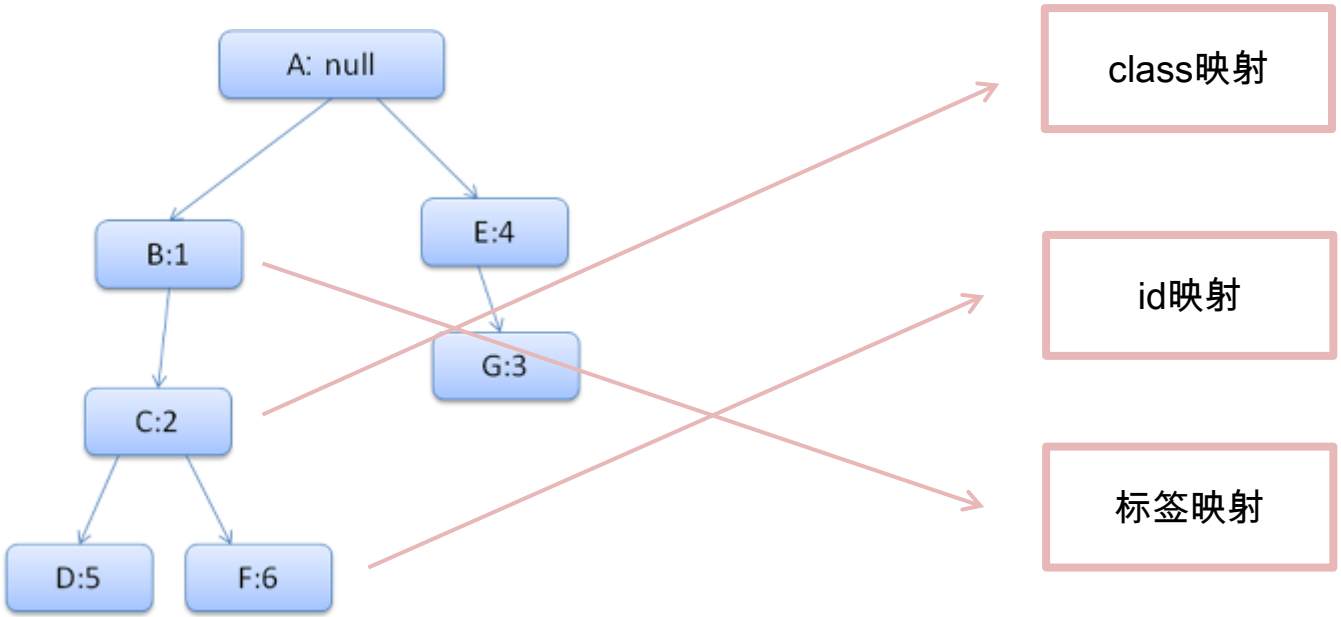
规则树



margin-top:5px;
margin-bottom:5px;
margin-left:5px;
margin-right:5px;
color:#00FF00;

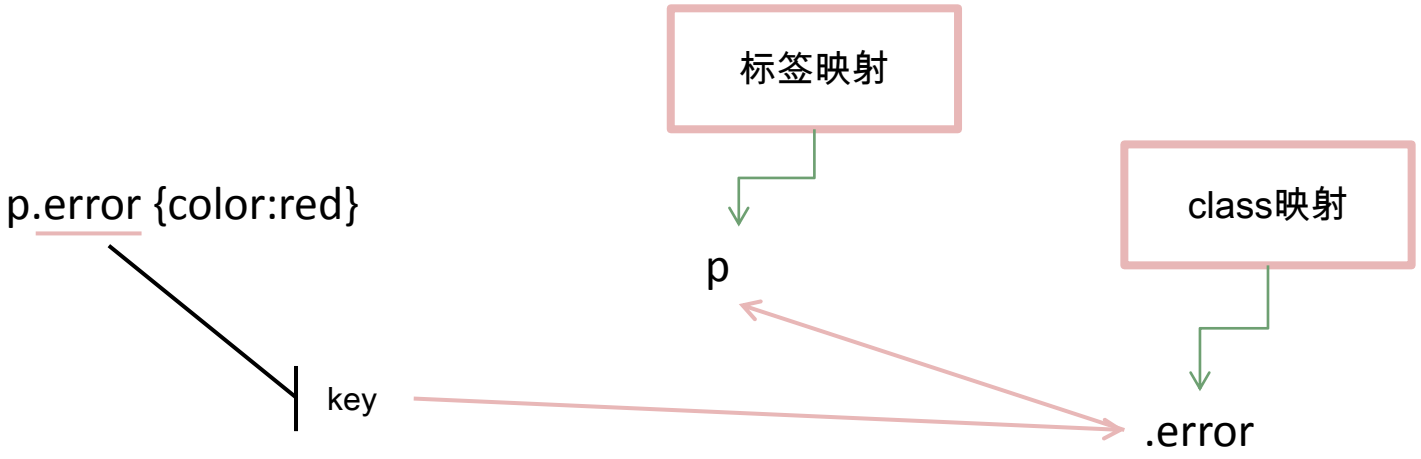
上下文树

通过映射提高匹配速度



这个优化使在进行规则匹配时减少了95 + %的工作量

映射例子



优化方案

- 避免使用全局的
- 不要限定ID选择
- 不要限定类选择符
- 避免使用后代选择符
- 质疑所有的子选择器
- 依赖继承

浏览器从右往左读取选择器！！

选择器效率：

1.id	(#myid)
2.class	(.myclass)
3.标签	(div, h1, p)
4.兄弟	(h1 + p)
5.孩子	(ul > li)
6.子选择器	(li a)
7.通配符	(*)
8.属性	(a[rel="external"])
9.伪类	(a:hover, li:first)

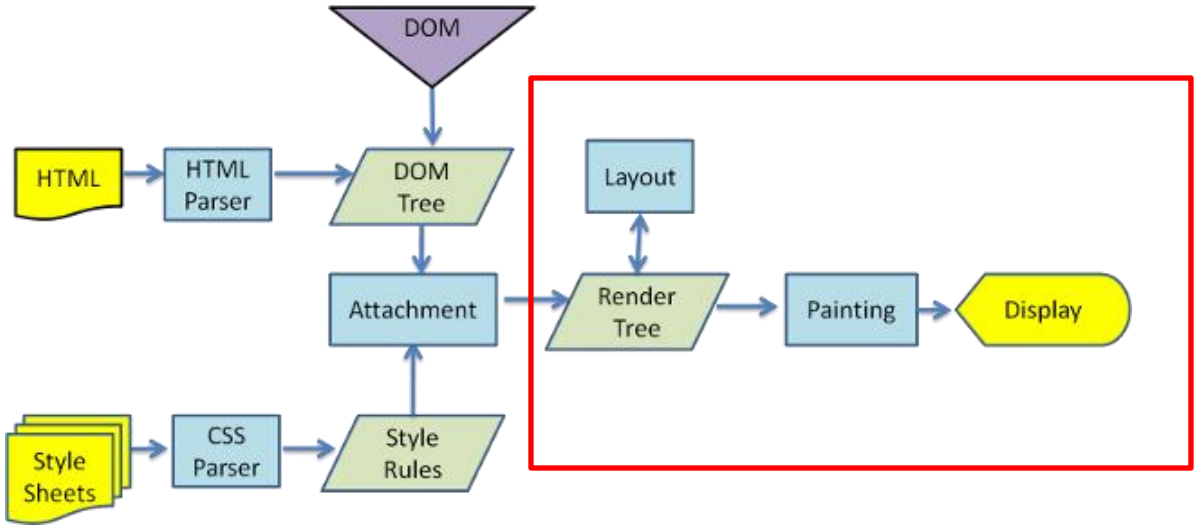
布局/绘制

输入

```
▼<html class>
  <!--<![endif]-->
 ▶<head>…</head>
 ▼<body>
  ▶<div style="display:block;clear:both;float:none;position:absolute;right
  ▶<script>…</script>
  ▶<script>…</script>
  ▶<script>…</script>
  ▶<div class="topbar">…</div>
  ▶<div id="skin-wrap" class="skin-wrap inner-hide" style="display: none; '
  ▶<div id="wrap">…</div>
   <div id="suggests" style="left: 330.5px; top: 165px; width: 438px; "></
  ▶<ul class="more-edition">…</ul>
   <!-- Constructed by 360F2E/75team.com -->
  ▶<script>…</script>
   <!--[if IE]>
   <script charset="utf-8" src="s.php?v5/ieplusv2.js&v=2dc7103658a0a129bfea
   <![endif]-->
   <script charset="utf-8" src="http://h.qhimg.com/magicwand.php"></script
   <script charset="utf-8" src="s.php?v5/biz.js,v5/
   minitip.js&v=1a9818b83b27049ca536c2f3768f72adf1664fced5cfcd26329402ba43
  ▶<div id="mtip-1" class="minitip radius2 right">…</div>
  ▶<div id="mtip-2" class="minitip radius2 right" style="left: 471px; top:
   ">…</div>
  ▶<div id="mtip-3" class="minitip radius2 right">…</div>
   <script src="http://hao.360.cn/css/monitorscript.js?v=20120619.js"></sc
  ▶<script>…</script>
  ▶<script>…</script>
```

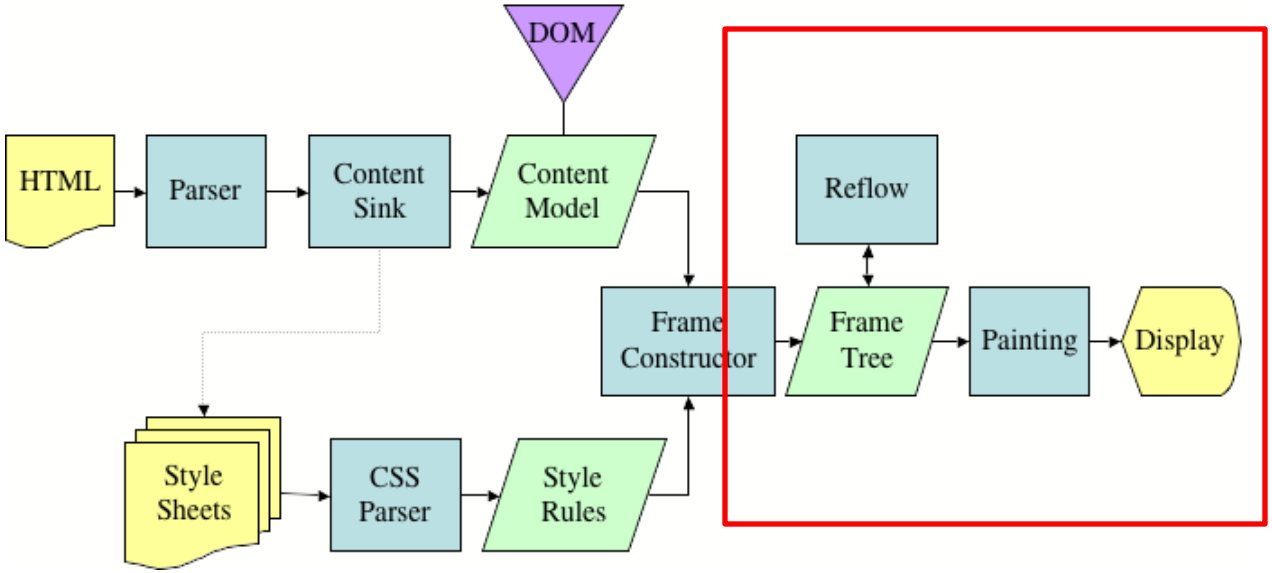
布局/绘制

输出

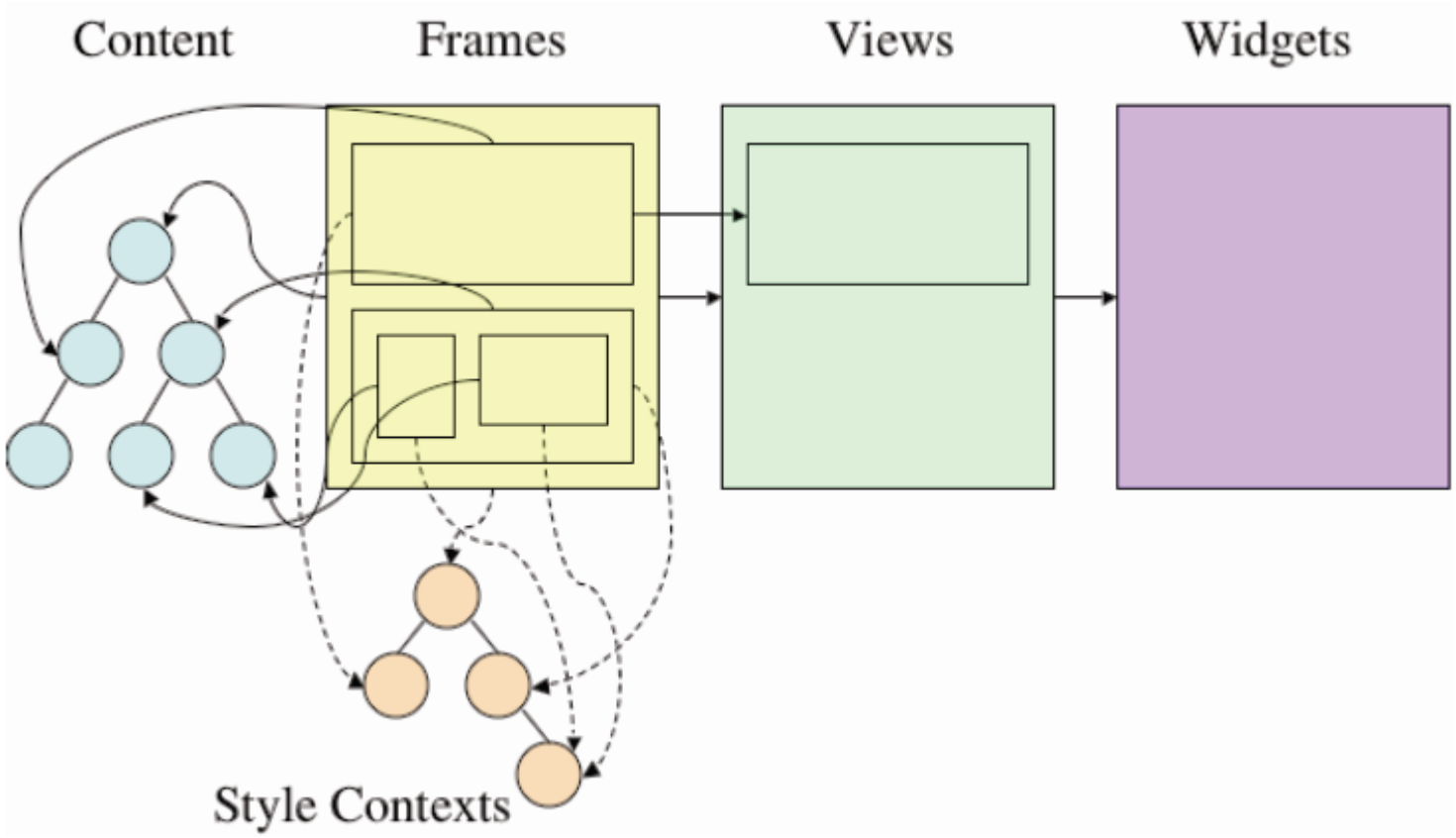


Webkit

Mozilla's Gecko

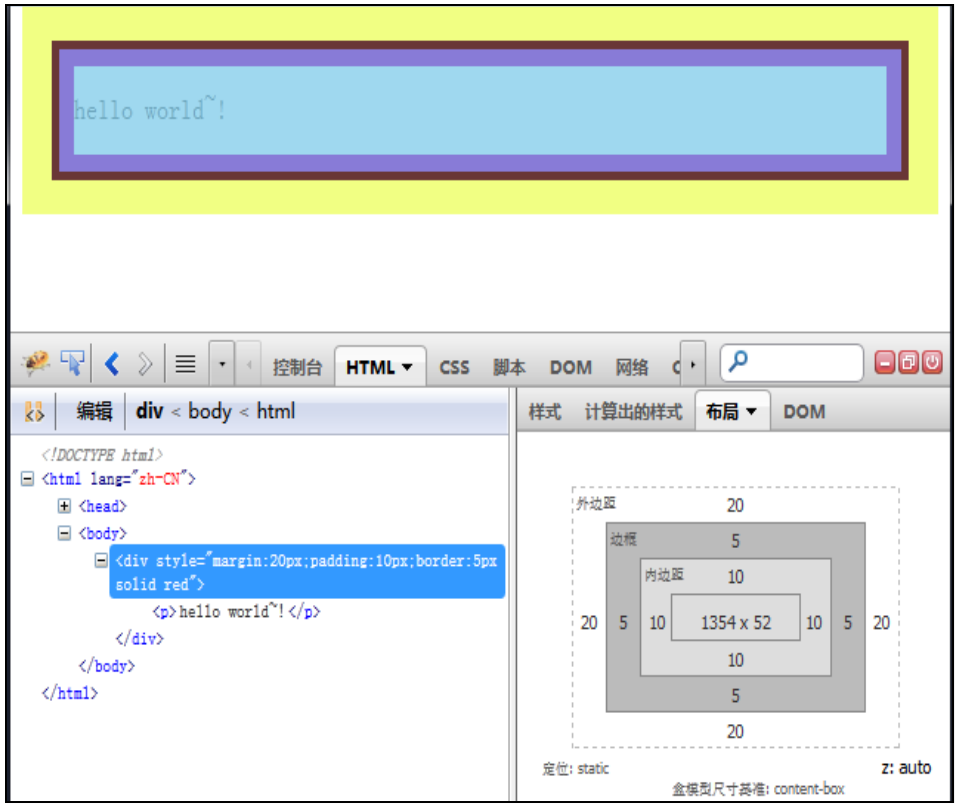


** 来自HOW BROWSERS WORK: BEHIND THE SCENES OF MODERN WEB BROWSERS*



- 计算渲染树中节点之间的位置和大小(CSS的盒模型)

```
<div  style="margin:20px;
padding:10px;  border:5px solid
red">
      <p>hello world~!</p>
</div>
```



BOX

height, width, border, spacing, padding, margin and position…

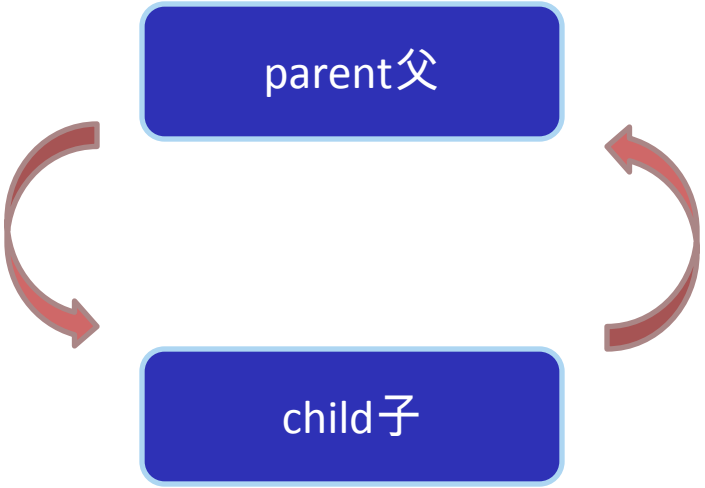
CSS box model

Layout

- 关于元素之间如何摆放的问题（CSS可视化渲染模式）

- 递归的过程

display, position, float



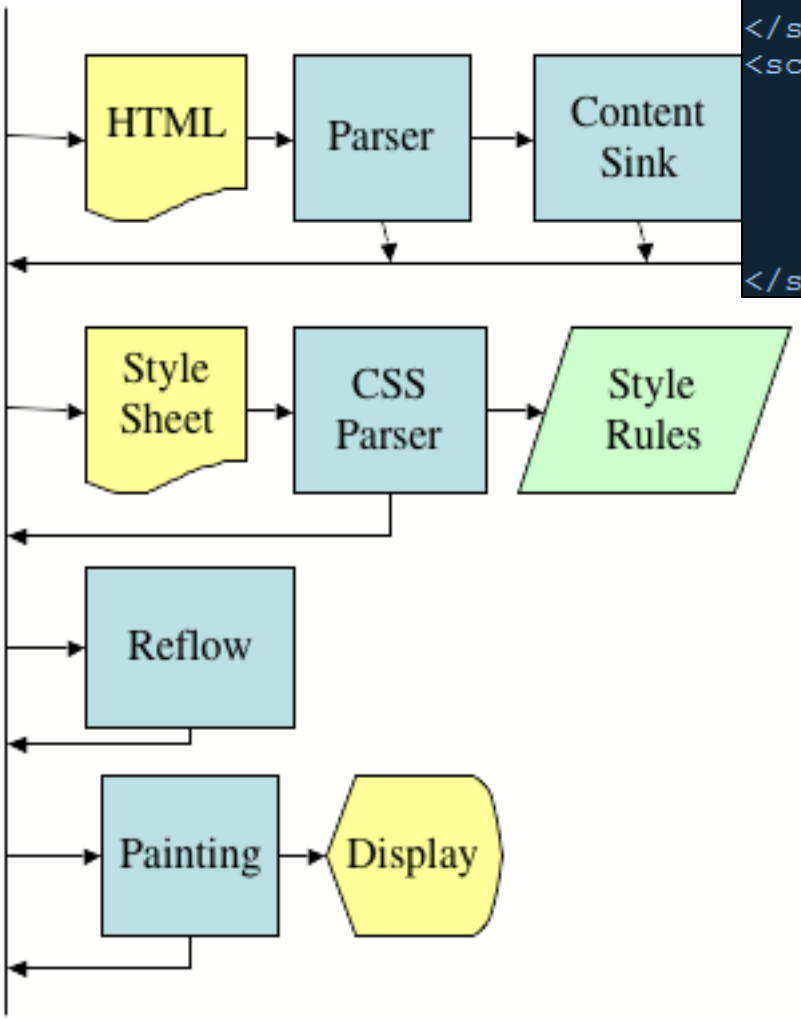
CSS Visual Rendering Model

何时Layout

- Initial 初始化页面的时候
- Incremental DOM的变化 (script脚本、网络接收到更多内容)
- Resize 窗口resize(从上到下的布局框架改变)
- StyleChange 字体大小的变化（全局）
- Dirty 针对child的一些增量布局

影响Layout？

```
<script>
    for(var i=0,len=values.length; i<len; i++){
        ul.innerHTML += "<li>"+values[i]+"</li>";
    }
</script>
<script>
    var itemsHtml = "";
    for(var i=0,len=values.length; i<len; i++){
        itemsHtml += "<li>"+values[i]+"</li>";
    }
    ul.innerHTML = itemsHtml;
</script>
```



- 首屏加载的DOM数
- DOM树的深度
- 频繁的DOM操作
- CSS规则的数量
- CSS规则的效率
- 集中修改样式
- …

Paint

- 实际的绘制其实是图形引擎，它来负责实际的像素的转换或者比如硬件加速的一些事情

- 块元素的绘制顺序：
  背景色、背景图、border、children、outline

The End

参考

- http://peter.sh/experiments/asynchronous-and-deferred-javascript-execution-explained/
- http://www.yuiblog.com/blog/2006/10/20/video-crockford-domtheory/ (Hackday.ppt)
- http://www.webkit.org/blog/1188/how-webkit-loads-a-web-page/
- http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/
- http://www.vineetgupta.com/2010/11/how-browsers-work-part-1-architecture/

使用高效的CSS选择符

- http://www.falconhan.com/webanalytics/Use-efficient-CSS-selectors.htm

新的试验样式scoped

- http://updates.html5rocks.com/2012/03/A-New-Experimental-Feature-style-scoped

接下来...


High Performance Web Sites
14 Steps to Faster-Loading Web Sites
高性能网站
建设指南
前端工程师技能精髓
Steve Souders 著
O'REILLY


Performance Best Practices for Web Developers
Even Faster Web Sites
高性能网站建设
进阶指南
Web开发者性能优化最佳实践
Steve Souders 著
O'REILLY

...