浏览器的内核，也就是我们所说的渲染引擎，是用来将HTML文档，转化成可视化的图像结果。所谓渲染就是，就是根据描述或者定义构建数学模型，通过模型生成图像。浏览器引擎就是将HTML/CSS/JS文本图像转换成可视化页面。

通过上面描述的功能，我们可以知道，浏览器的渲染引擎都有哪些模块，

html解释器：负责解析html文档，并生成DOM树，

css解释器：解析css，为DOM树中的每个元素计算样式信息，为后面网页的布局提供基础。

布局：DOM树构建完成之后，联合解析成功的css样式信息，计算他们的大小位置信息，形成一个能够表示所有信息的内部模型。

js引擎 ：负责解析js代码。

上面这些模块依赖，网络，存储，2D/3D图像，音频和视频解码，图像解码。

网页的结构有哪些？

1.框结构：所有的元素都是按照框的结构去设计的，所有的元素按照，块级元素和内联元素区分，在布局的时候根据本身的特性加上css的改变去布局。框结构可以对网页进行布局，将网页分隔成几个框，也可以在网页嵌入其他网页。

2.层次特性：网页中的元素可能分布在不同的层次中，某些元素可能分布在不同于父元素的层次中，影响层级的因素有css的作用，当然还有元素本身特性，video，canvas，就不会和父元素在一个层级，因为video用来播放视频，为它创建一个新的层级便于浏览器的解码和渲染交互问题。canvas元素有复杂的2D/3D操作，所以会有自己的层级。

WEBKIT的渲染过程

1.当用户输入URL的时候，浏览器调用资源加载器去加载对应的网页；

2.加载器依赖网络模块建立连接，发送请求，这里需要TCP连接；

3.WEBKIT接收到各种网页或者资源数据，其中某些资源可能是异步或者同步获取的；

4.网页被交给HTML解释器生成一系列的词语TOKEN,

5.解释器根据词语构建节点(NODE),形成DOM树，

6.如果节点是JS，调用js引擎去执行，

7.如果节点需要其他资源，例如图片,css,视频，浏览器调用资源加载器去加载资源，但是他们是异步的，在加载的时候不会阻塞当前DOM树的构建，如果遇到没有标记异步加载的scripr标签，则停止DOM树的创建，直到js文件被加载执行完毕，这个行为是浏览器默认的行为，因为在js代码中可能存在改变DOM树结构的代码。

DOM树构建完成，触发DOMCONENT事件，资源加载完毕并且DOM树构建完毕，触发onload事件。

renderObject：

1.css文件解被解析完成之后，构建一个内部结构，

2.css解析完之后，再DOM树上附加解释后的样式信息，值就是renderObject;

3.renderObject在创建的同时，渲染引擎会根据网页的层次结构构建renderLayer树，

4.最后就是根据绘图上下文去绘图，这一过程依赖2D/3D图形库，浏览器在绘图的时候为了安全和高效，会引入复杂的机制，从单纯但软件绘图到GPU加速渲染，混合模式渲染。（

CSS animations, transforms 以及 transitions 不会自动开启GPU加速，而是由浏览器的缓慢的软件渲染引擎来执行。那我们怎样才可以切换到GPU模式呢，很多浏览器提供了某些触发的CSS规则。

现在，像Chrome, FireFox, Safari, IE9+和最新版本的Opera都支持硬件加速，当它们检测到页面中某个DOM元素应用了某些CSS规则时就会开启，最显著的特征的元素的3D变换

可是在一些情况下，我们并不需要对元素应用3D变换的效果，那怎么办呢？这时候我们可以使用个小技巧“欺骗”浏览器来开启硬件加速。

虽然我们可能不想对元素应用3D变换，可我们一样可以开启3D引擎。例如我们可以用transform: translateZ(0); 来开启硬件加速 。

）

资源加载和网络栈

网络资源加载是页面渲染的第一步，也是必不可少的一步。网页本身就是一种资源，而且网页本身还要依赖其他类型的资源，比如图片，视频，因为资源加载涉及网络和资源缓存等机制，HTML的资源包括下面的类型:

HTML,JS,IMAG,CSS,SVG,字体文件，XSL,

上面的资源在WEBKIT中有不同的类来表示，公共基类是CachedResource，

子类是：CachedcssStyle, Cachedscript,Cachedimage,CachedFont,.......

之所以每个子类都有Cached，这是因为效率问题引入的缓存机制，所有资源的获取都要先去请求缓存中的信息，以便决定是否向服务器提出资源请求。

资源缓存

资源缓存机制是提高资源使用率的有效方法，他的基本思想是建立一个资源缓存池，当请求资源的时候，先去资源缓存池查看是否有相应的资源，有则用，没有则去发送请求给网络模块去请求服务器。这里的缓存是内存缓存，不同于网络栈中的磁盘缓存。

wibket从缓存池中寻找资源，是根据url关键字，因为标记资源的唯一标识符就是URL。

资源加载器

资源加载器有三种，1.特定的资源加载器，例如image对应imageLoader，css的font对应FontLoader,

2.资源缓存机制加载器加载资源，所有的特定加载器都共享它来查找并插入缓存资源。

3.通用资源加载器，当需要从文件系统或者网络获取资源的时候使用该类去加载。

鉴于网络加载资源是一个耗时的操作，通常一些资源的加载是异步的，也就是说资源的加载不阻塞页面的渲染，但是遇到js代码的时候，会阻塞页面的渲染，这个时候解析没办法继续执行，对后面要用的资源也不知道，不能发送请求。当遇到这种情况时，浏览器启用另外一个线程，去遍历后面的HTML网页，收集要发送的URL然后发送请求，这样就避免被阻塞。与此同时，webkit能够并发的去下载这些资源，提升网页的加载速度。

资源的生命周期

资源池中的资源不能无限大，需要用相应的机制来管理其中的资源，管理资源池使用LRU(least recent used).资源加载后会被放进资源池中。但是页面刷新的时候，并不是去完全请求资源缓存池中的内容，而是根据资源的http协议，去决定是否要去请求最新的资源。如果从缓存中取资源，则状态码是304.

这样减少http请求，减少网络负载。

网络栈

网络栈的基本组成http,dns，websocket,还有一些新技术，为了减少网络延迟而引进。太难理解了所以放弃了原理部分。

域名解析(DNS)

用户通过域名来访问网络资源，所以在建立TCP连接时要解析域名，调用一个阻塞函数getaddrinfo(),该函数是阻塞函数，所以使用线程来处理。

同样为了考虑效率，使用HostCache来保存解析后的域名，域名解析看似没什么特殊的地方，但是同样存在可优化的地方。

高性能网络栈

         DNS预取和tcp连接

    一次DNS的查询时间大概是60~120ms，而tcp连接时间大概也就是十几毫秒，看似十分简短的时间，对网页来说是一个非常长的时间，那么如何有效的减少这段时间----DNS预取和tcp连接。

首先是DNS预取技术，它主要的思想是利用现有的DNS机制，提前解析网页中可能的网络连接，具体来讲，当用户在浏览具有超链接的网站时，提取所有的超链接，将域名抽取出来，利用较少的cpu和网络带宽解析这些域名的ip地址，这样一来用户不会察觉到这一过程，特别是域名解析较慢的时候，效果特别明显。只能是DNS预取，不能提前建立TCP连接，

HTTP管线化

        管线化，需要浏览器和服务器的配合，才能正常的工作，原理是同时将多个HTTP请求一次性的提交给服务器，因此无需等待服务器的回复，因为它能将多个http请求填充在一个tcp数据包内，管线化需要在网络上传输较少的数据包。管线化机制需要通过永久链接，并且只有get和head请求可以管线化。

HTML解释器和DOM模型

DOM的全称是文档对象模型，它可以以一种独立于平台和语言的方式访问系应该一个文档的内容和结构。DOM定义的是一组与平台，语言无关的接口，该接口允许编程语言动态访问和更改结构化文档。

DOM结构构成的基本要素是”节点“，而文档的DOM结构就是由层次化的节点组成，DOM模型中， 节点的概念很宽泛，整个文档是一个节点，称为文档节点，HTML中的标记(tag)也是一种节点，称为元素节点，还有一些其他类型的节点，例如属性节点，文本节点，注释节点，

事件的工作过程

事件在工作过程中使用两个主体，第一是事件(event),第二是事件目标，每个事件都有事件属性来标记该事件的事件目标，当事件到达事件目标时，在这个目标上注册的事件将会被触发，监听者调用的顺序是不固定的，所以不能根据代码逻辑注册顺序来决定触发顺序。

事件处理最重要的部分就是事件捕获和事件冒泡两种机制，当渲染引擎接收到一个事件后，会检查哪个元素是直接的事件目标。事件