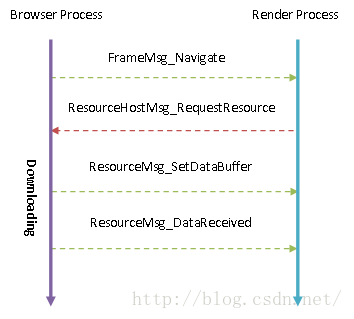
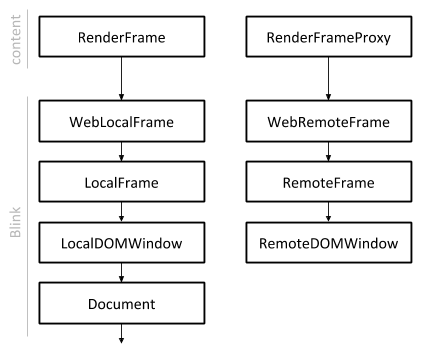
1. WebContentsImpl是一个网页的基本，里面包含一个FrameTree对象frame\_tree\_，frame\_tree\_对象创建一个FrameTreeNode 对象root\_。Frame\_tree\_是一个只有根节点的Frame Tree。根节点描述的网页就是接下来要进行加载的。根节点描述的网页加载完成之后，就会进行解析。在解析的过程中，如果碰到iframe标签，那么就会创建另外一个子节点，并且添加到当前正在创建的Frame Tree中去。
2. Chromium在Browser进程中为网页创建了一个Frame Tree之后，会将网页的URL发送给Render进程进行加载。Render进程接收到网页URL加载请求之后，会做一些必要的初始化工作，然后请求Browser进程下载网页的内容。Browser进程一边下载网页内容，一边又通过共享内存将网页内容传递给Render进程解析，也就是创建DOM Tree。本文接下来就分析网页URL的加载过程。
3. 
4. RenderFrameImpl类的成员变量frame\_指向的是一个WebLocalFrameImpl对象。如果当前正在处理的RenderFrameImpl对象还没有加载过URL，并且当前要加载的URL不为空，RenderFrameImpl类的成员函数OnNavigate会调用成员变量frame\_指向的WebLocalFrameImpl对象的成员函数loadRequest加载指定的URL。
5. WebLocalFrameImpl类的成员函数loadRequest首先调用成员函数frame获得成员变量m\_frame描述的一个LocalFrame对象，接着又调用这个LocalFrame对象的成员函数loader获得其成员变量m\_loader描述的一个FrameLoader对象。有了这个FrameLoader对象之后，就调用它的成员函数load加载参数request描述的URL。
6. FrameLoader类的成员函数loadWithNavigationAction首先调用成员函数client获得一个FrameLoaderClientImpl对象，接着再调用这个FrameLoaderClientImpl对象的成员函数createDocumentLoader为参数action描述的URL创建了一个WebDataSourceImpl对象，并且保存在成员变量m\_policyDocumentLoader中。关于FrameLoader类的成员函数client和FrameLoaderClientImpl类的成员函数createDocumentLoader的实现，可以参考前面[Chromium Frame Tree创建过程分析](http://blog.csdn.net/luoshengyang/article/details/50450100" \t "_blank)一文。
7. FrameLoader类的成员函数loadWithNavigationAction接下来又将成员变量m\_policyDocumentLoader描述的WebDataSourceImpl对象转移到另外一个成员变量m\_provisionalDocumentLoader中，最后调用这个WebDataSourceImpl对象的成员函数startLoadingMainResource加载参数action描述的URL。
8. DocumentLoader类的成员变量m\_fetcher描述的是一个ResourceFetcher对象，DocumentLoader类的成员函数startLoadingMainResource调用这个ResourceFetcher对象的成员函数fetchMainResource请求加载本地变量cachedResourceRequest描述的资源。这个资源描述的即为上一步指定要加载的URL。ResourceFetcher类的成员函数fetchMainResource执行结束后，会返回一个RawResource对象。这个RawResource对象保存在WebDataSourceImpl类的成员变量m\_mainResource中。这个RawResource对象描述的是一个异步加载的资源，DocumentLoader类的成员startLoadingMainResource调用它的成员函数addClient将当前正在处理的DocumentLoader对象添加到它的内部去，用来获得异步加载的资源数据，也就是本地变量cachedResourceRequest描述的URL对应的网页内容。
9. 有的Render Frame只是一个Proxy，称为Render Frame Proxy。Render Frame Proxy描述的是在另外一个Render进程中进行加载和渲染的网页。这种网页在WebKit里面对应的Frame和DOM Window分别称为Remote Frame和Remote DOM Window。由于Render Frame Proxy描述的网页不是在当前Render进程中加载和渲染，因此它是没有Document的。

       相应地，Render Frame描述的是在当前Render进程中进行加载和渲染的网页，它是具有Document的，并且这种网页在WebKit里面对应的Frame和DOM Window分别称为Local Frame和Local DOM Window。

       从图1我们还可以看到，在Render Frame和Local Frame之间，以及Render Frame Proxy和Remote Frame之间，分别存在一个Web Local Frame和Web Remote Frame。Web Local Frame和Web Remote Frame是属于WebKit Glue层的概念。从前面[Chromium网页加载过程简要介绍和学习计划](http://blog.csdn.net/luoshengyang/article/details/50414848" \t "_blank)一文可以知道，WebKit Glue层的作用是将WebKit的对象类型转化为Chromium的对象类型，这样Chromium的Content层就可以用统一的、自有的方式管理所有的对象。关于Chromium的层次划分和每一个层次的作用，可以参考前面[Chromium网页加载过程简要介绍和学习计划](http://blog.csdn.net/luoshengyang/article/details/50414848" \t "_blank)一文。

1. 
2. Browser进程一边下载网页的内容，一边将下载回来的网页交给Render进程的Content模块。Render进程的Content模块经过简单的处理之后，又会交给WebKit进行解析。WebKit是从ResourceLoader类的成员函数didReceiveData开始接收Chromium的Content模块传递过来的网页内容的，因此我们就从这个函数开始分析WebKit解析网页内容的过程，也就是网页DOM Tree的创建过程。
3. DocumentWriter类的成员变量m\_parser指向的是一个HTMLDocumentParser对象，DocumentWriter类的成员函数addData调用这个HTMLDocumentParser对象的成员函数appendBytes对下载回来的网页数据进行解析。
4. HTMLDocumentParser类的成员函数pumpTokenizerIfPossible首先调用成员函数isScheduledForResume判断当前正在处理的HTMLDocumentParser对象是否处于等待重启继续解析网页内容的状态中。如果是的话，等到定时器超时时，当前线程就会自动调用当前正在处理的HTMLDocumentParser对象的成员函数pumpTokenizer对剩下未解析的网页内容进行解析。这种情况必须要确保参数mode的值为AllowYield，也就是确保当前正在处理的HTMLDocumentParser对象使用异步方式解析网页内容。

       如果当前正在处理的HTMLDocumentParser对象是以同步方式解析网页内容，那么HTMLDocumentParser类的成员函数pumpTokenizerIfPossible接下来就会马上调用成员函数pumpTokenizer对刚才下载回来的网页内容进行解析。

1. HTMLDocumentParser类的成员函数constructTreeFromHTMLToken所做的事情就是根据参数rawToken描述的一个Token来不断构造网页的DOM Tree。这个构造过程是通过调用成员变量m\_treeBuilder描述的一个HTMLTreeBuilder对象的成员函数constructTree实现的。