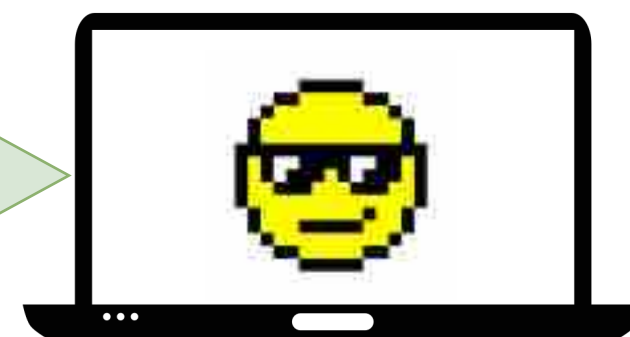
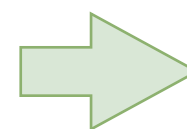
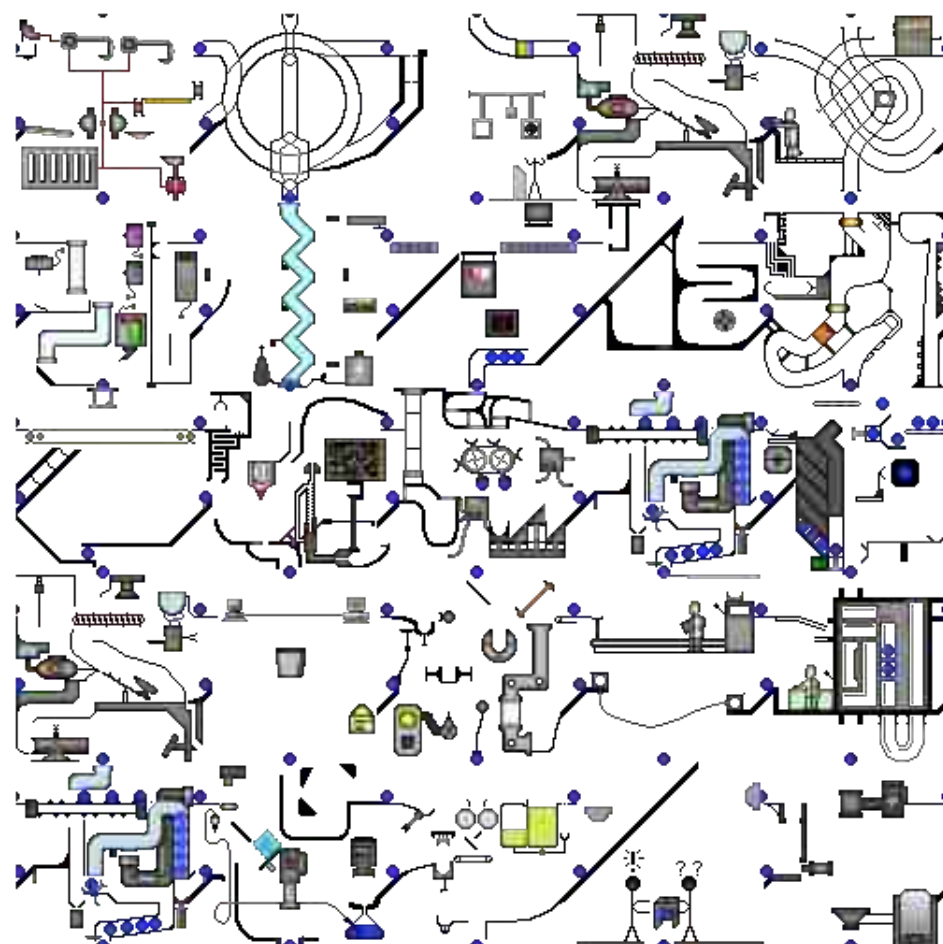
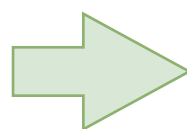
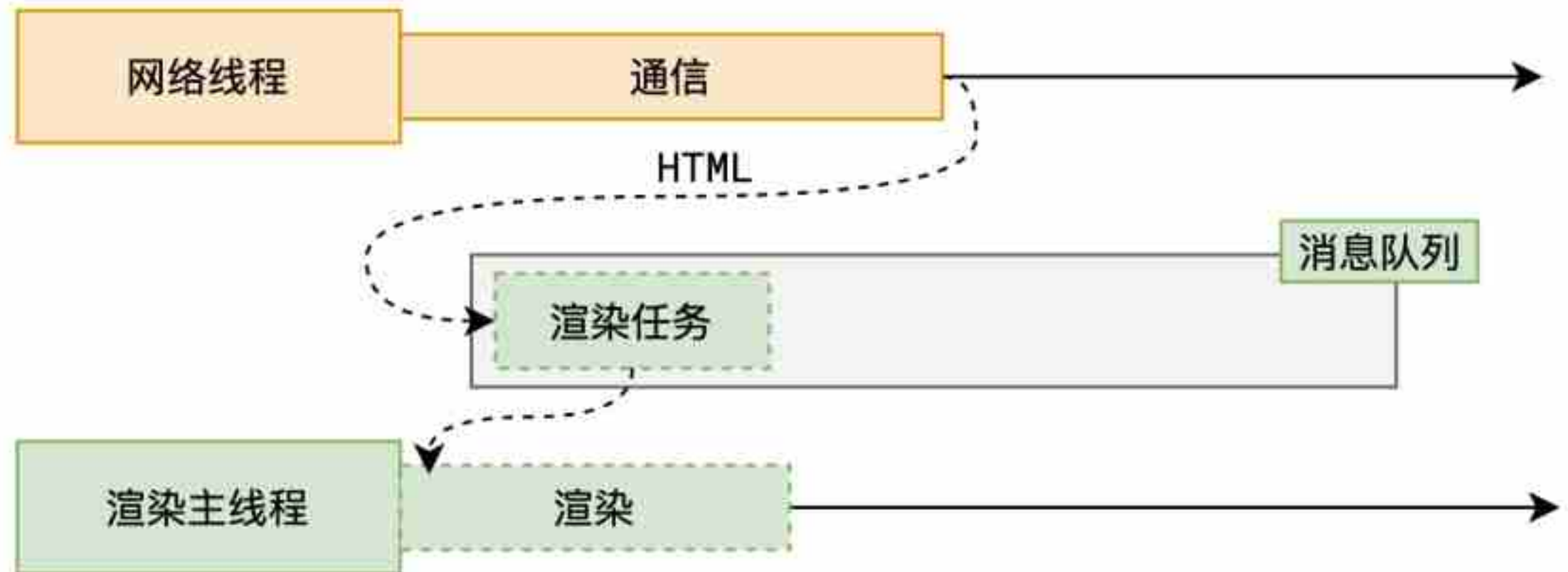


浏览器渲染原理

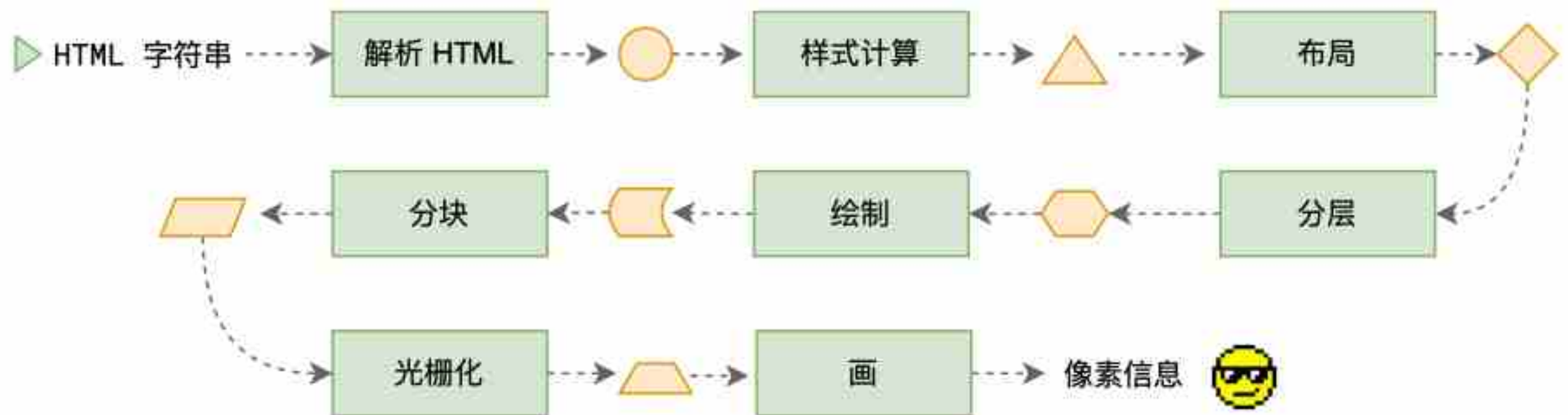
```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="us...>  
    <title>Page name</title>  
    <meta name="descri...>  
    <link href="assets/...>  
    <link rel="shortcut...>  
  </head>
```



渲染时间点

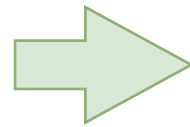


渲染流水线

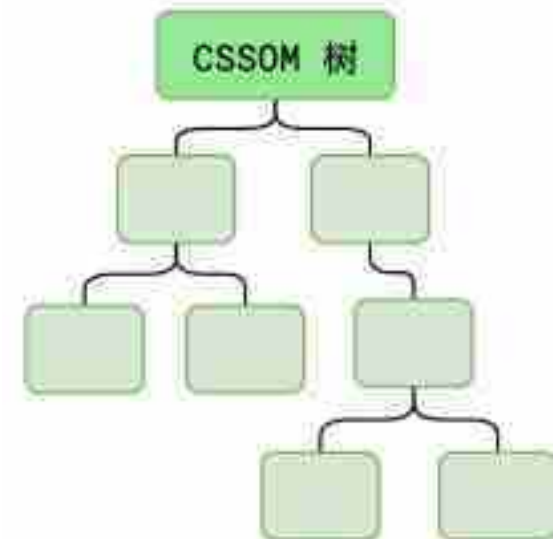
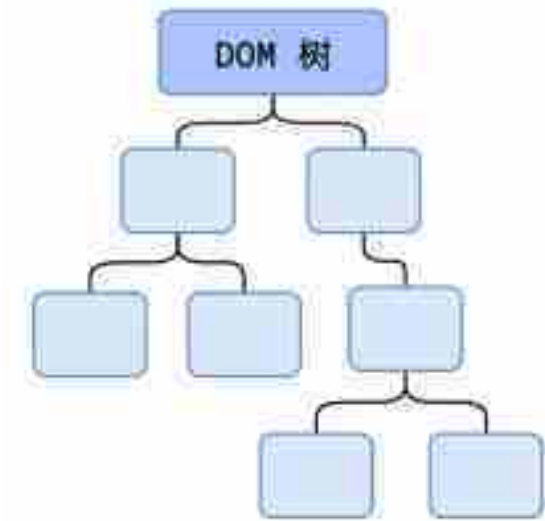
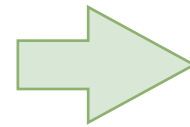


1. 解析 HTML - Parse HTML

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="us">  
    <title>Page name</title>  
    <meta name="descri">  
    <link href="assets/">  
    <link rel="shortcut">  
  </head>
```

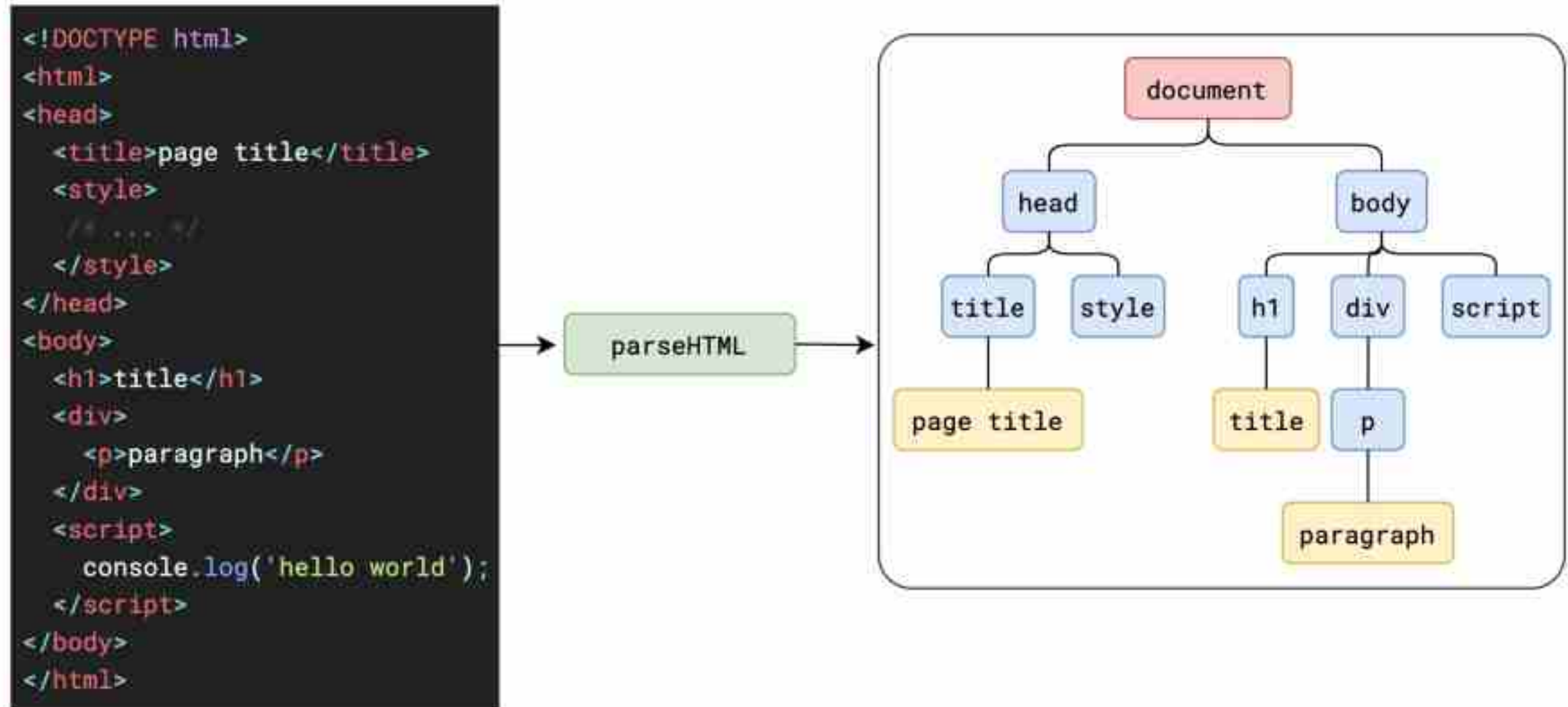


parseHTML



1. 解析 HTML - Parse HTML

Document Object Model

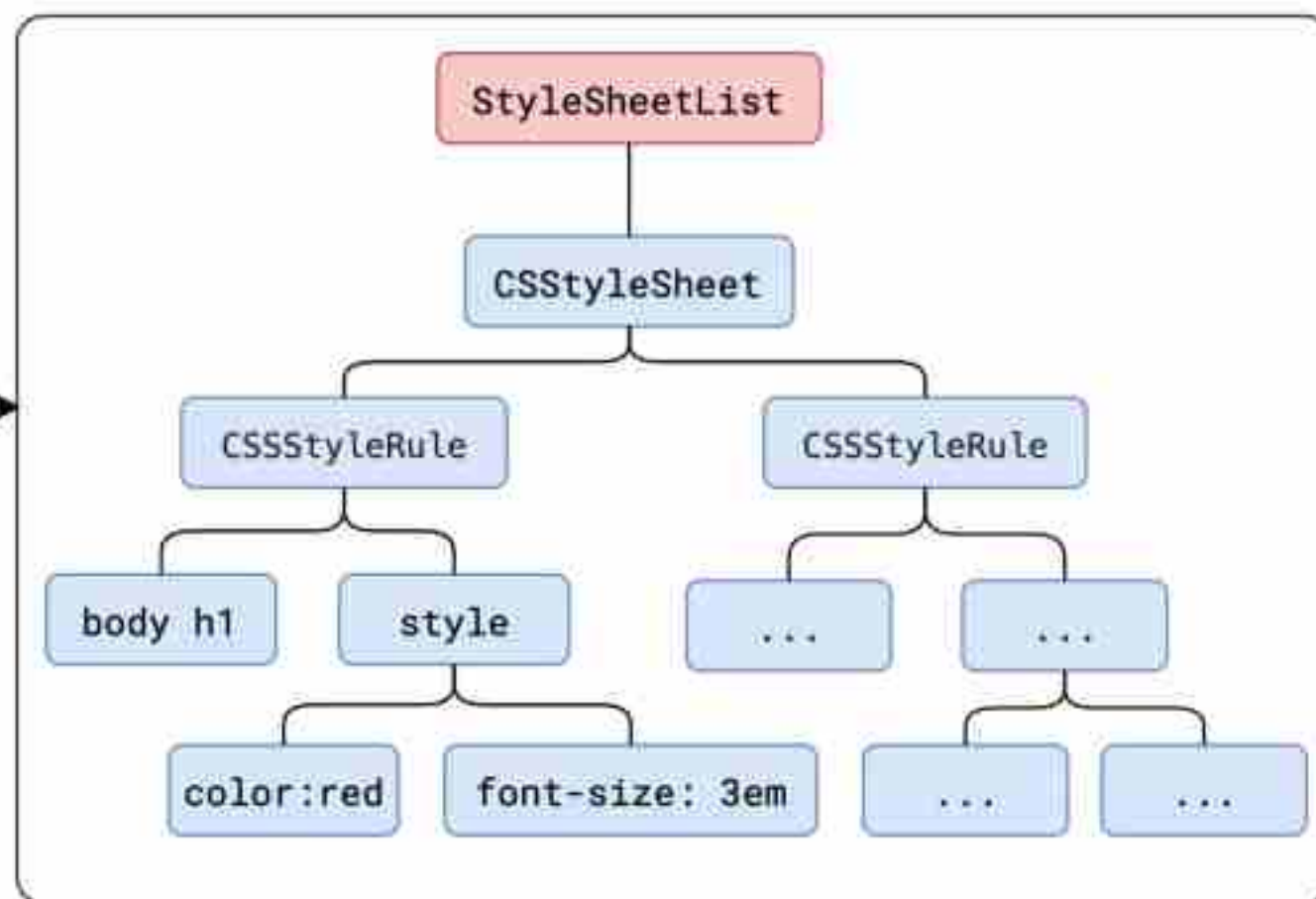


1. 解析 HTML - Parse HTML

CSS Object Model

```
body h1 {  
  color: red;  
  font-size: 3em;  
}  
  
div p {  
  margin: 1em;  
  color: blue;  
}
```

解析 CSS

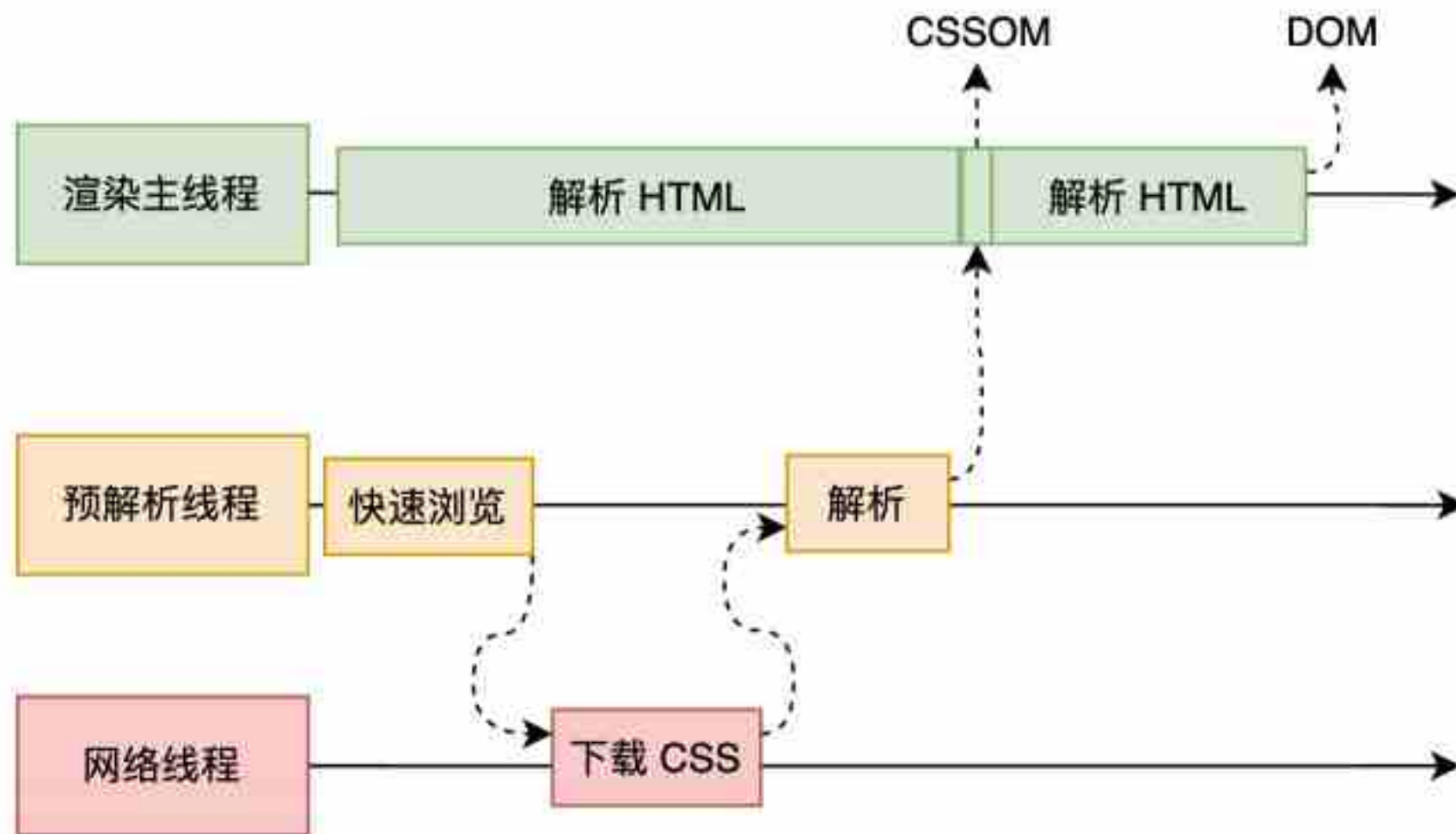


1. 解析 HTML - Parse HTML

常见问题

HTML 解析过程中遇到 CSS 代码怎么办？

为了提高解析效率，浏览器会启动一个预解析器率先下载和解析 CSS



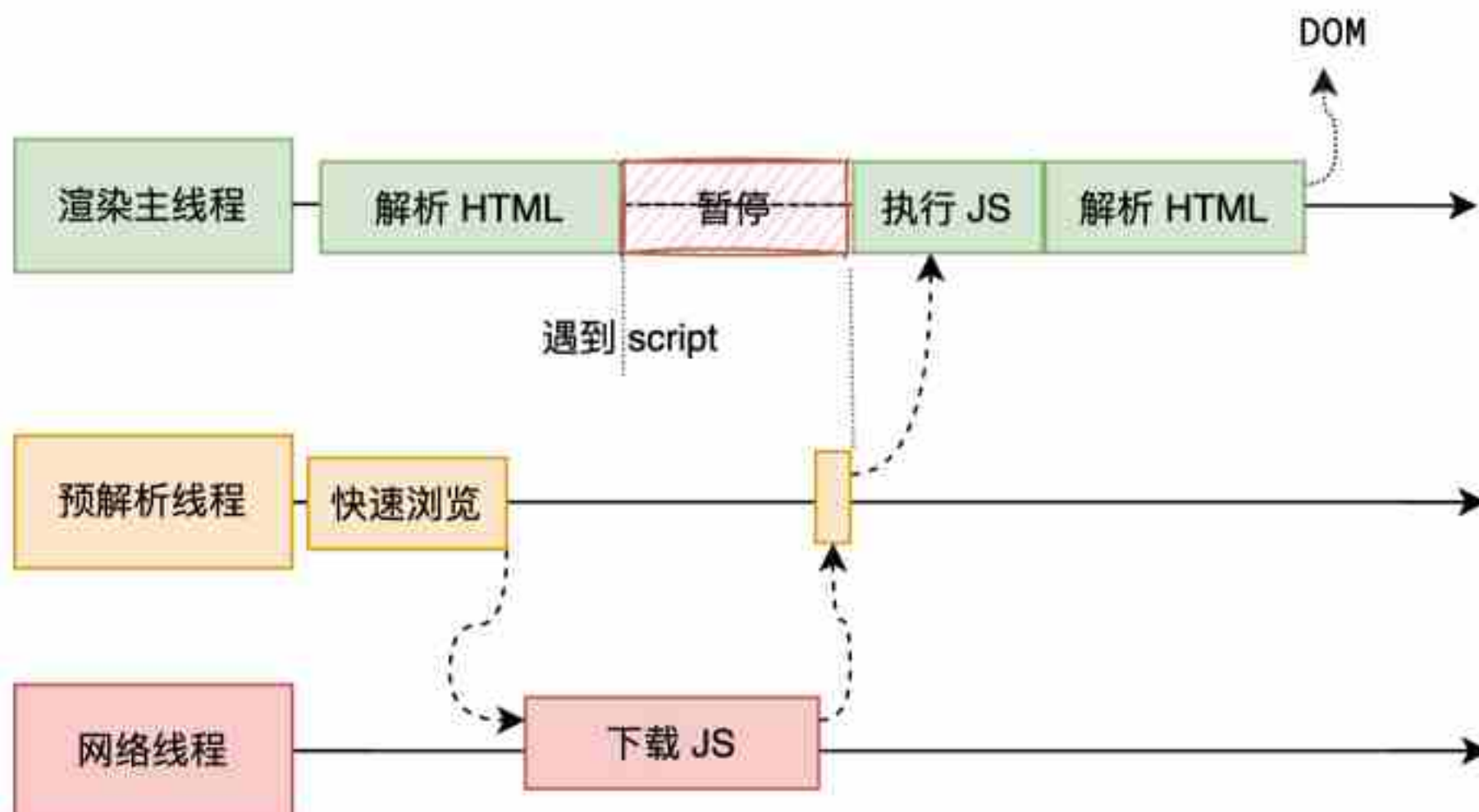
1. 解析 HTML - Parse HTML

常见问题

HTML 解析过程中遇到 JS 代码怎么办？

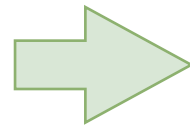
渲染主线程遇到 JS 时必须暂停一切行为，等待下载执行完后才能继续

预解析线程可以分担一点下载 JS 的任务

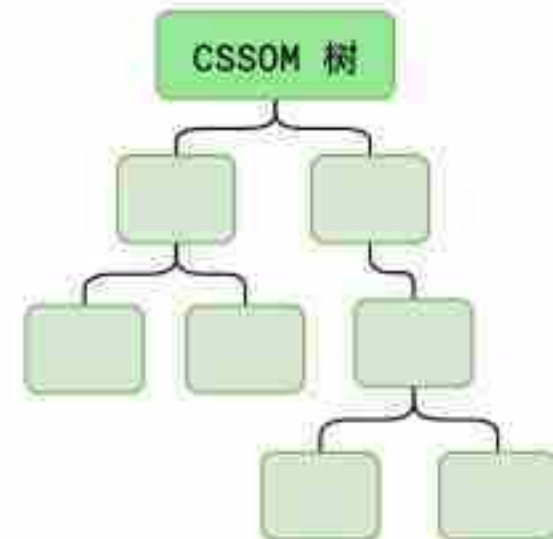
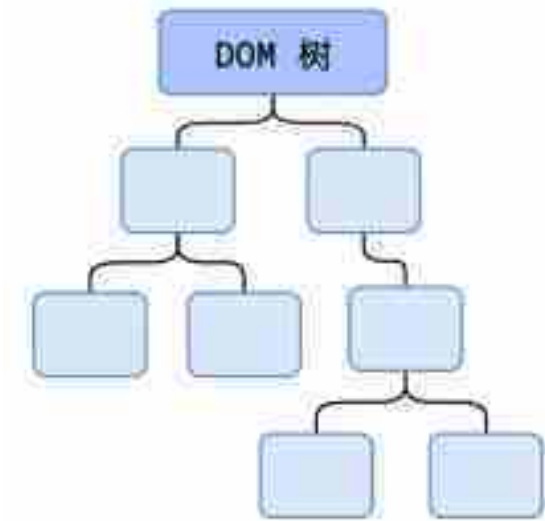
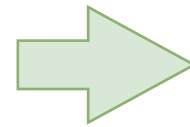


1. 解析 HTML - Parse HTML

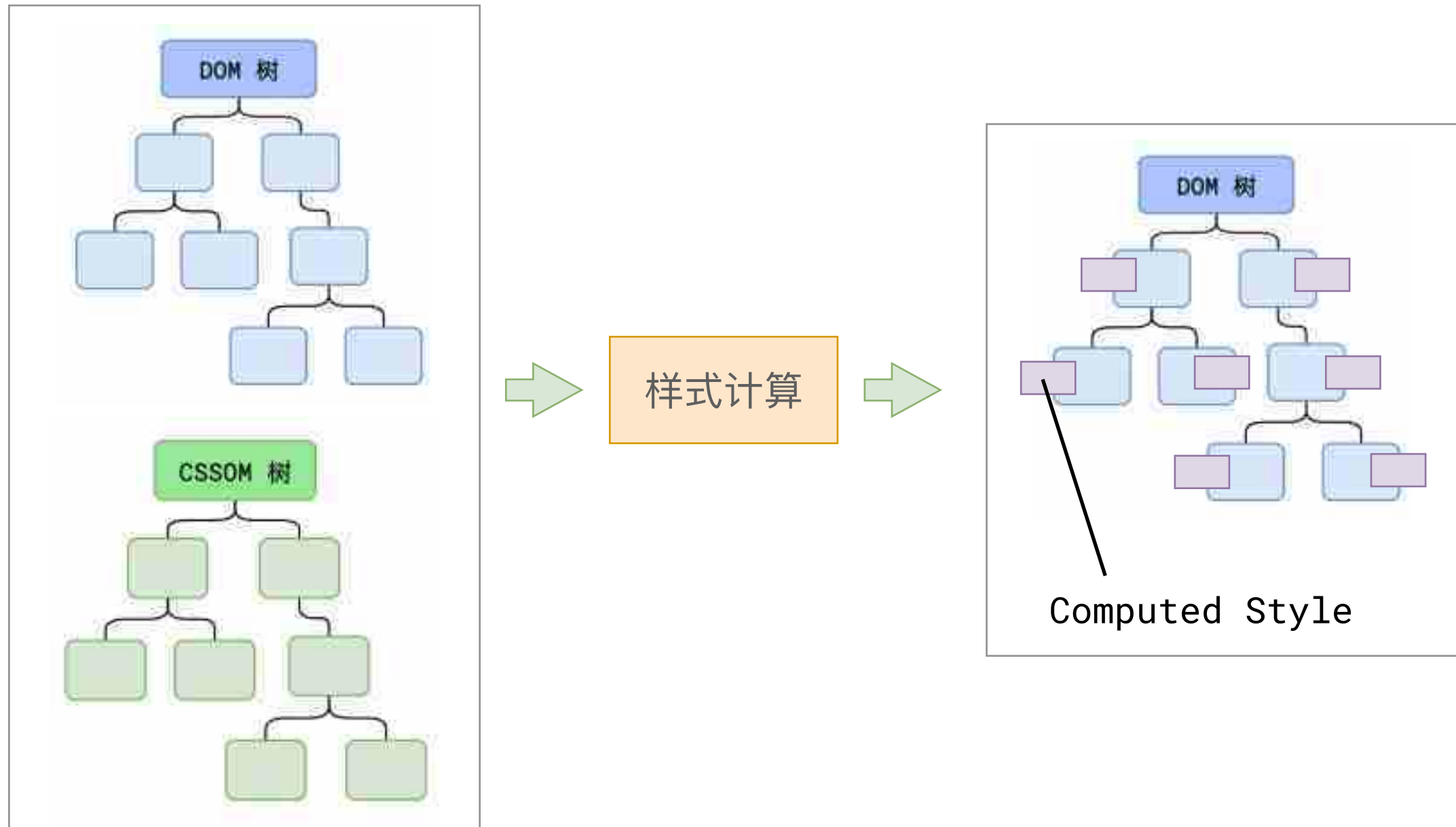
```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="us">  
    <title>Page name</title>  
    <meta name="descri">  
    <link href="assets/">  
    <link rel="shortcut">  
  </head>
```



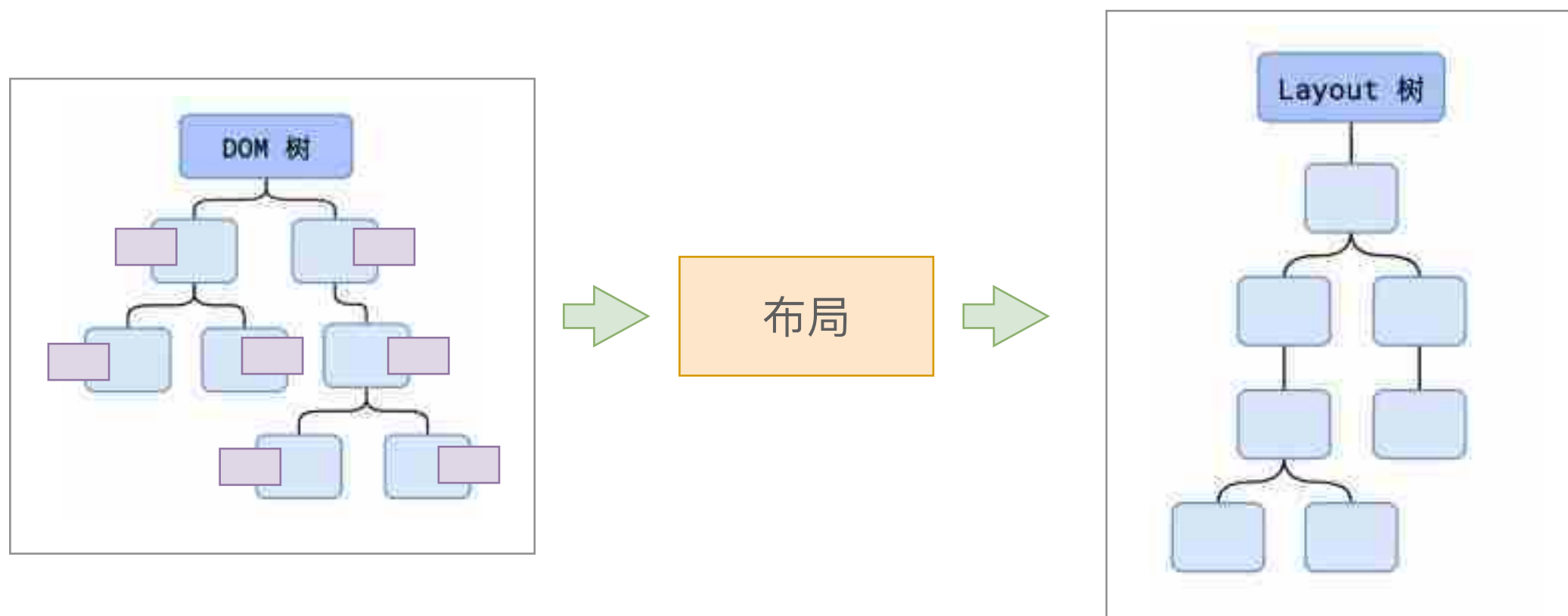
parseHTML



2. 样式计算 - Recalculate Style



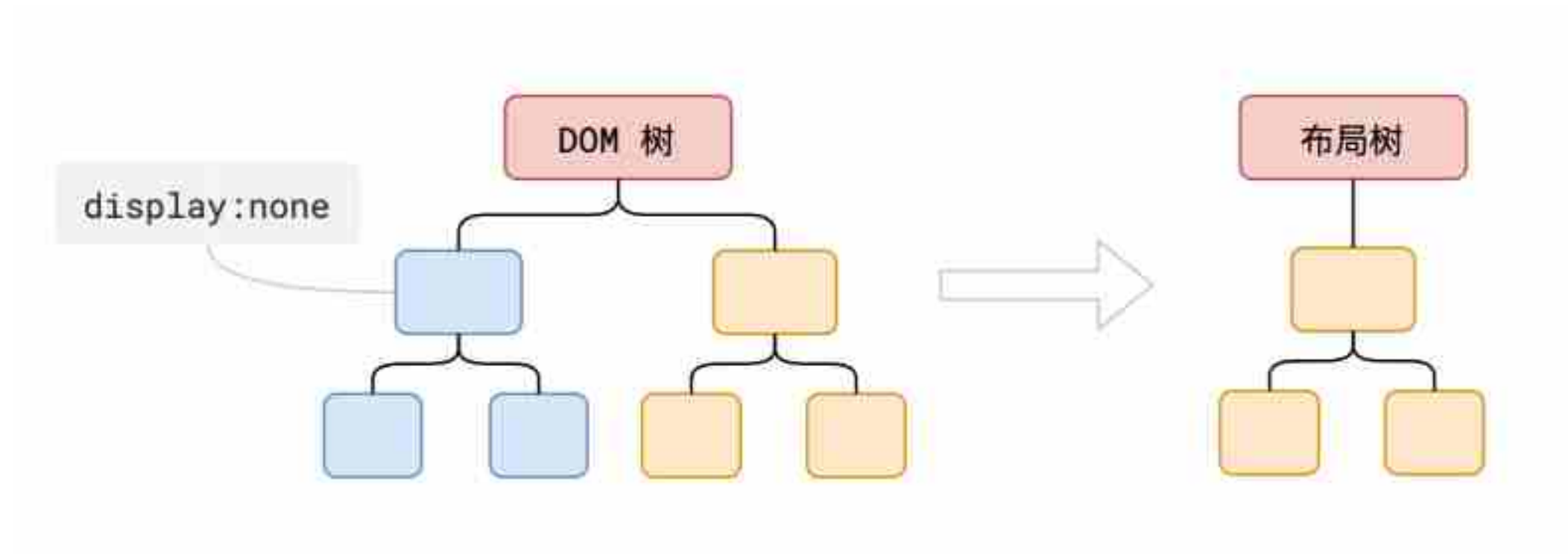
3. 布局 - Layout



3. 布局 - Layout

DOM 树 和 Layout 树

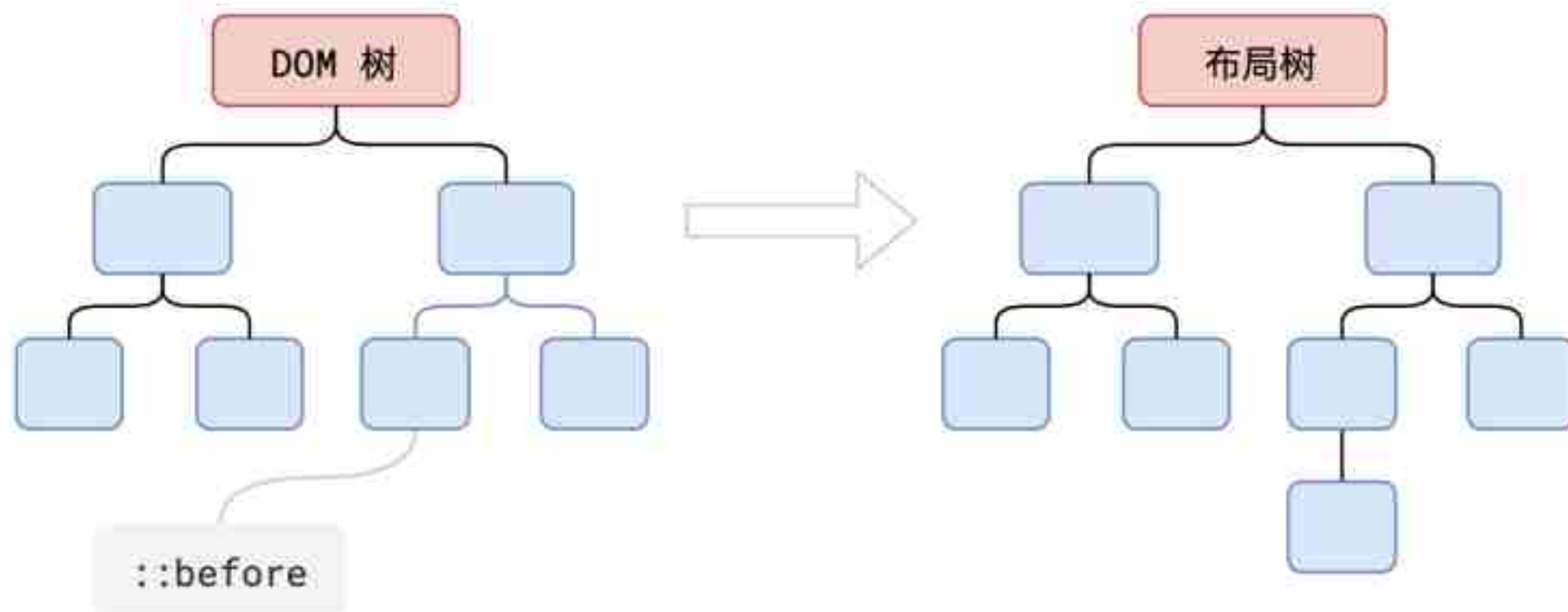
DOM 树 和 Layout 树不一定是一一对应的



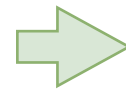
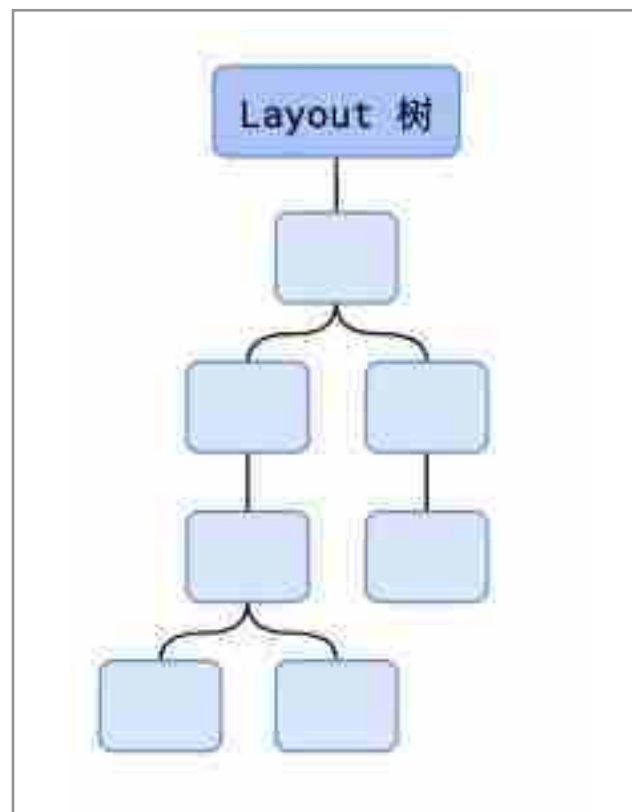
3. 布局 - Layout

DOM 树 和 Layout 树

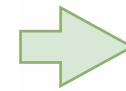
DOM 树 和 Layout 树不一定是一一对应的



4. 分层 - Layer



分层



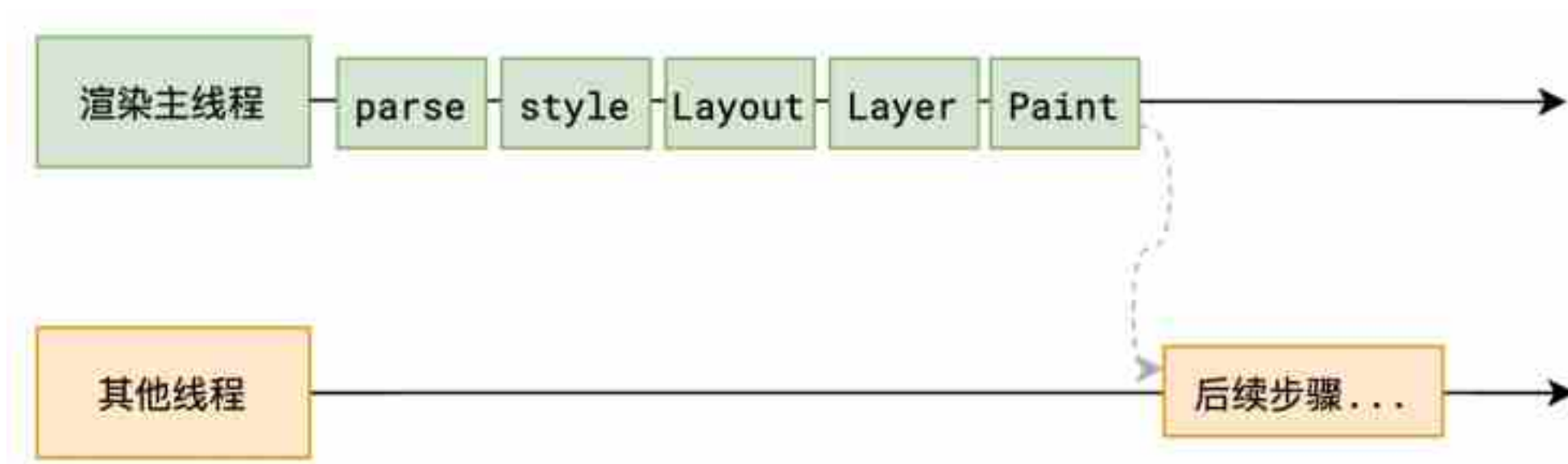
5. 绘制 - Paint

这里的绘制，是为每一层生成如何绘制的指令



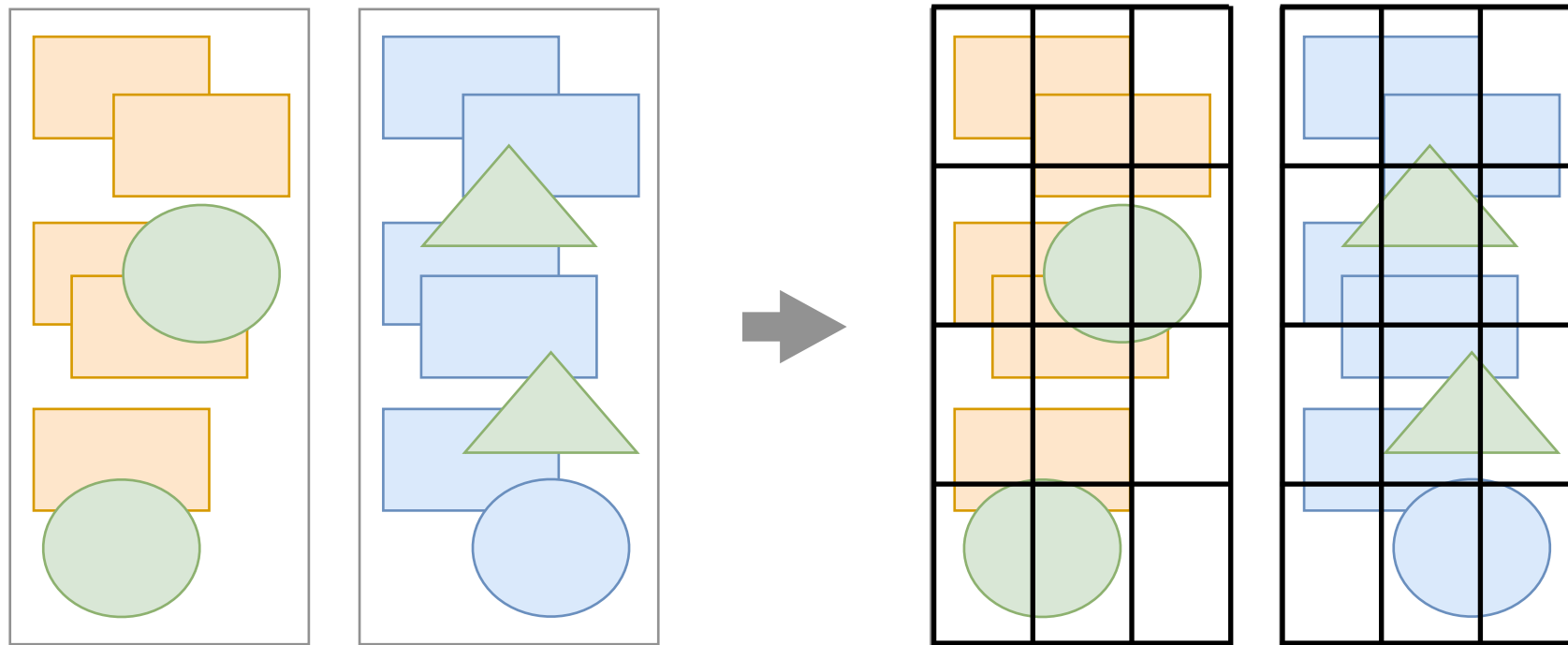
5. 绘制 - Paint

渲染主线程的工作到此为止，剩余步骤交给其他线程完成



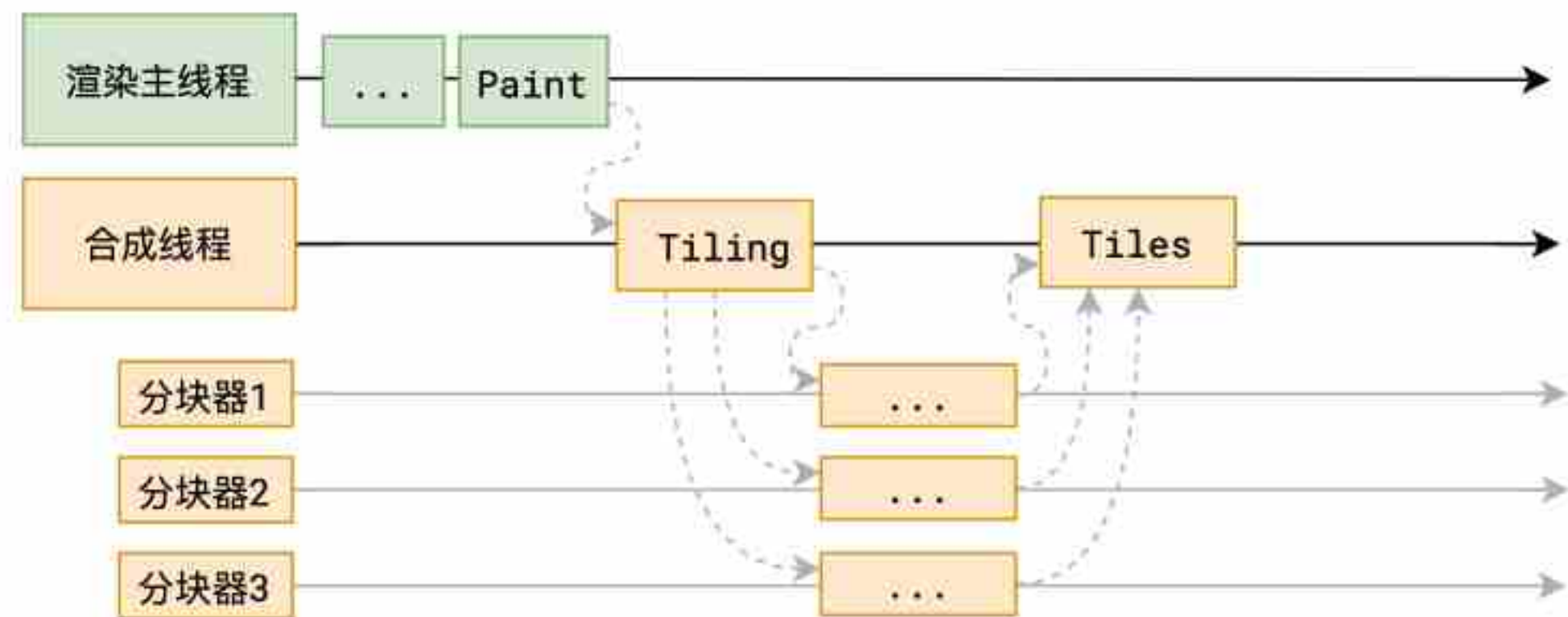
6. 分块 - Tiling

分块会将每一层分为多个小的区域



6. 分块 - Tiling

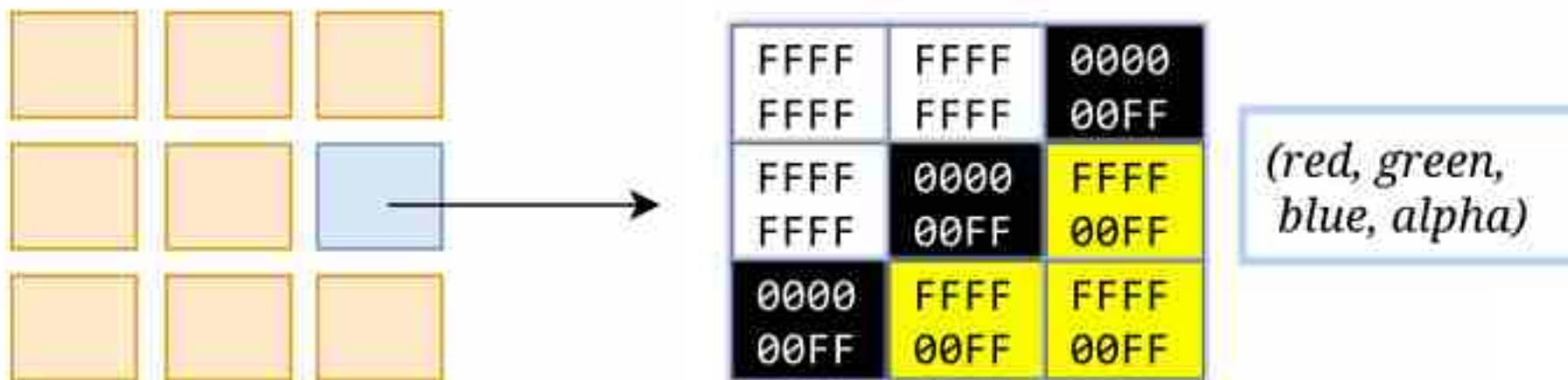
分块的工作是交给多个线程同时进行的



7. 光栅化 - Raster

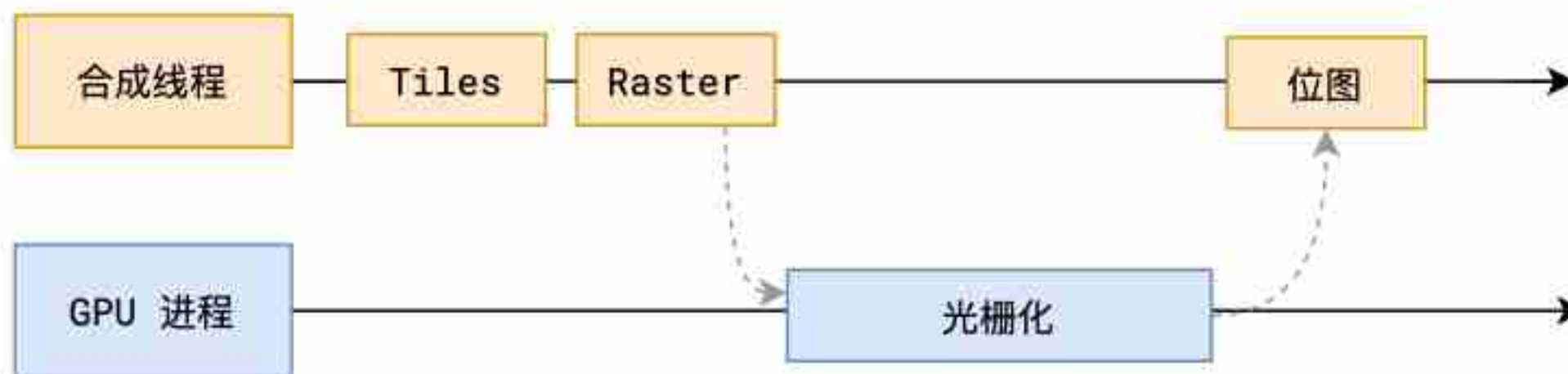
光栅化是将每个块变成位图

优先处理靠近视口的块



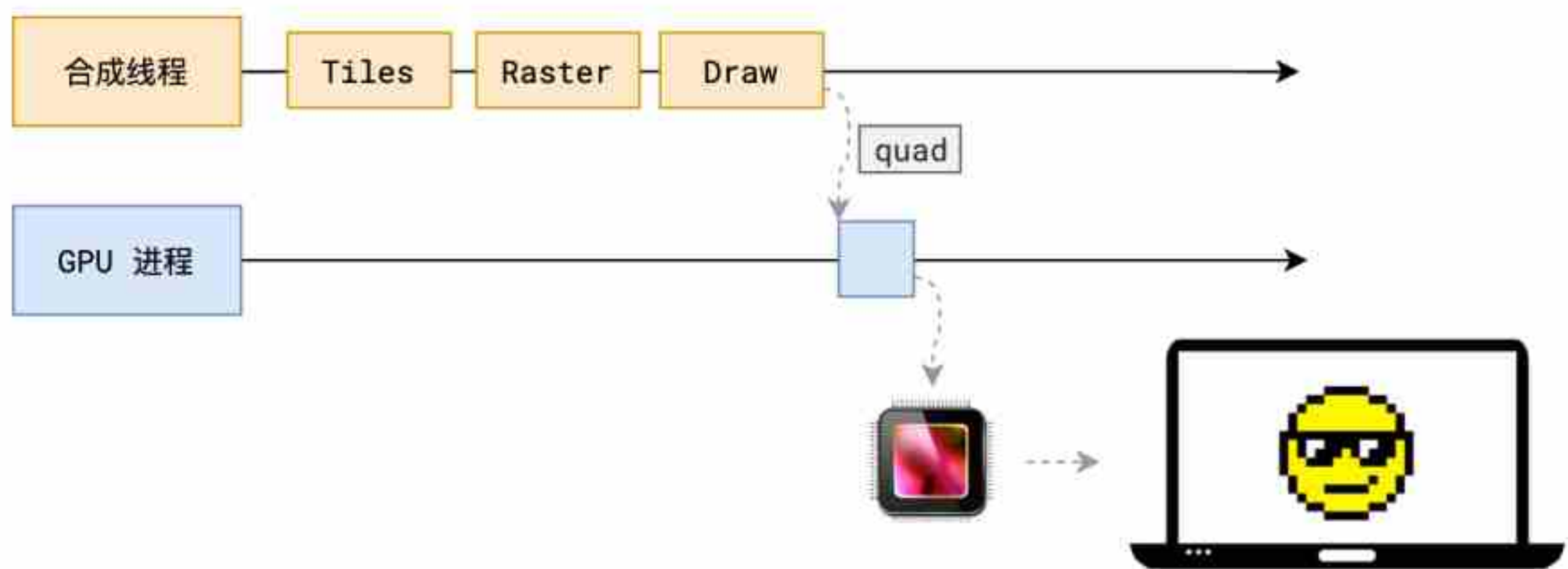
7. 光栅化 - Raster

此过程会用到 GPU 加速

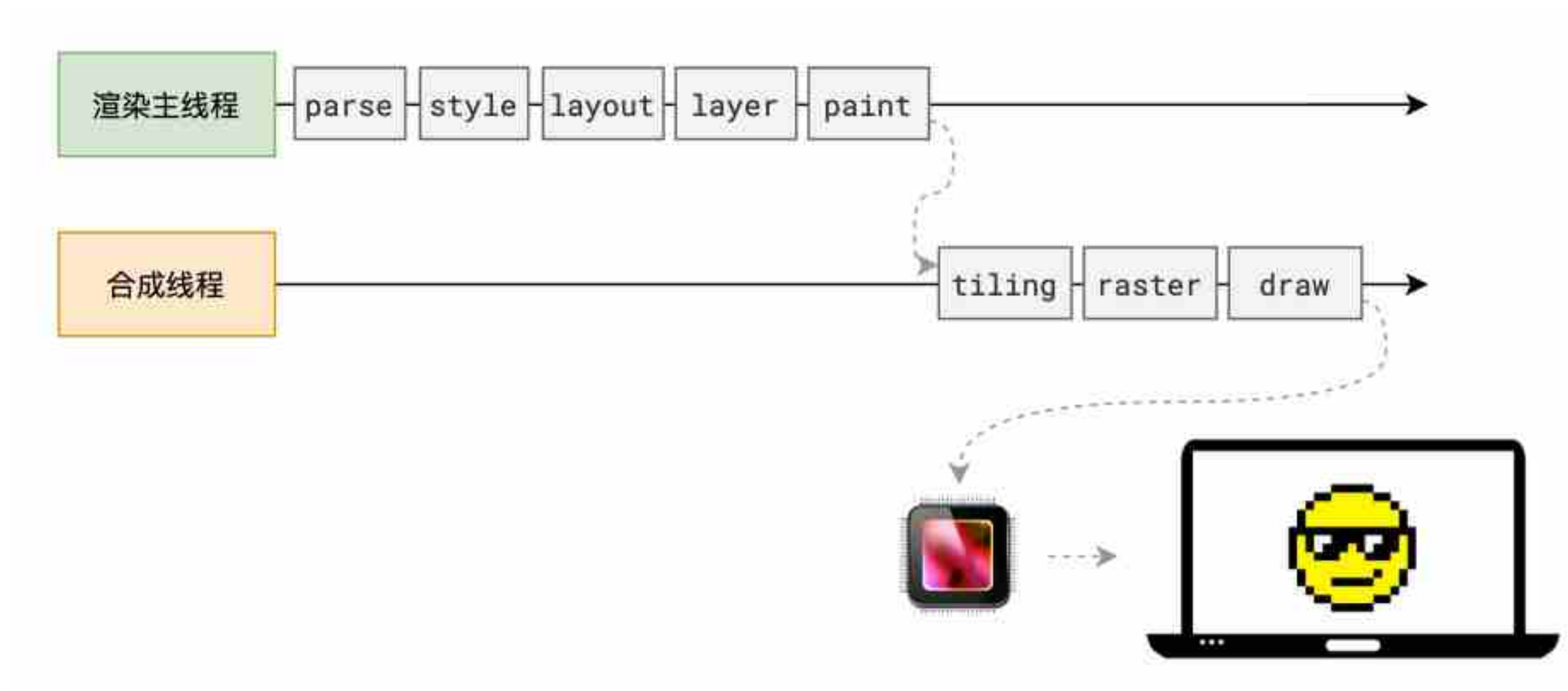


8. 画 - Draw

合成线程计算出每个位图在屏幕上的位置，交给 GPU 进行最终呈现

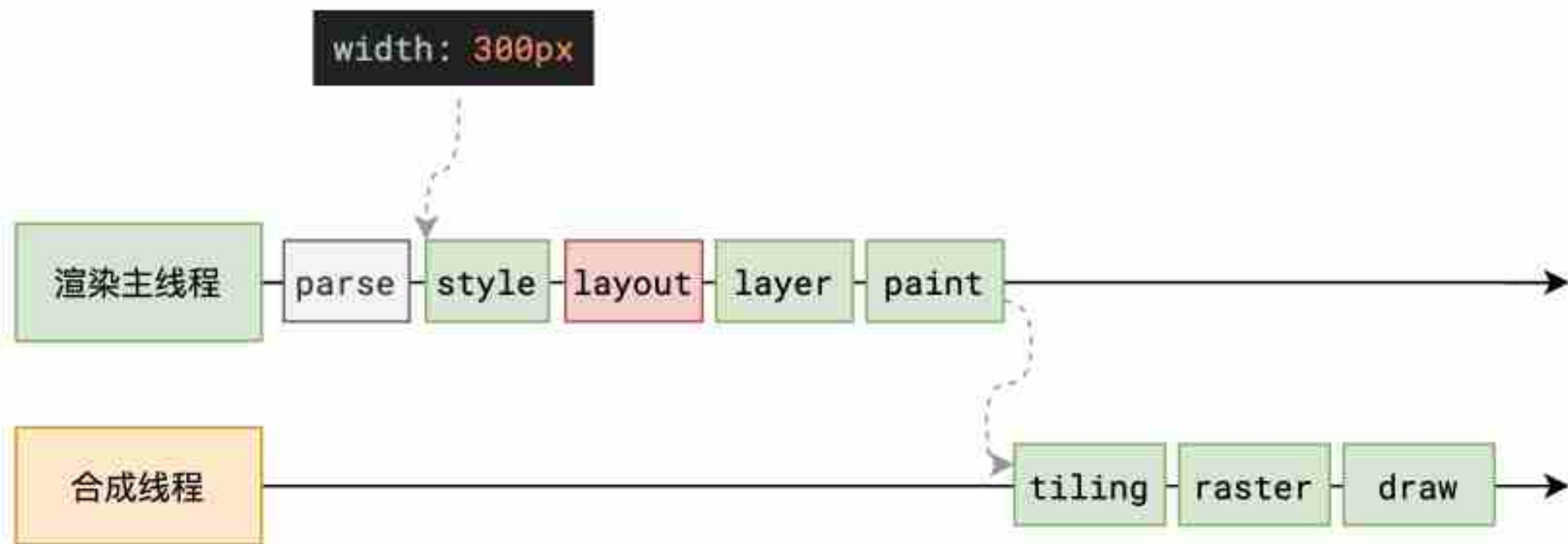


完整过程



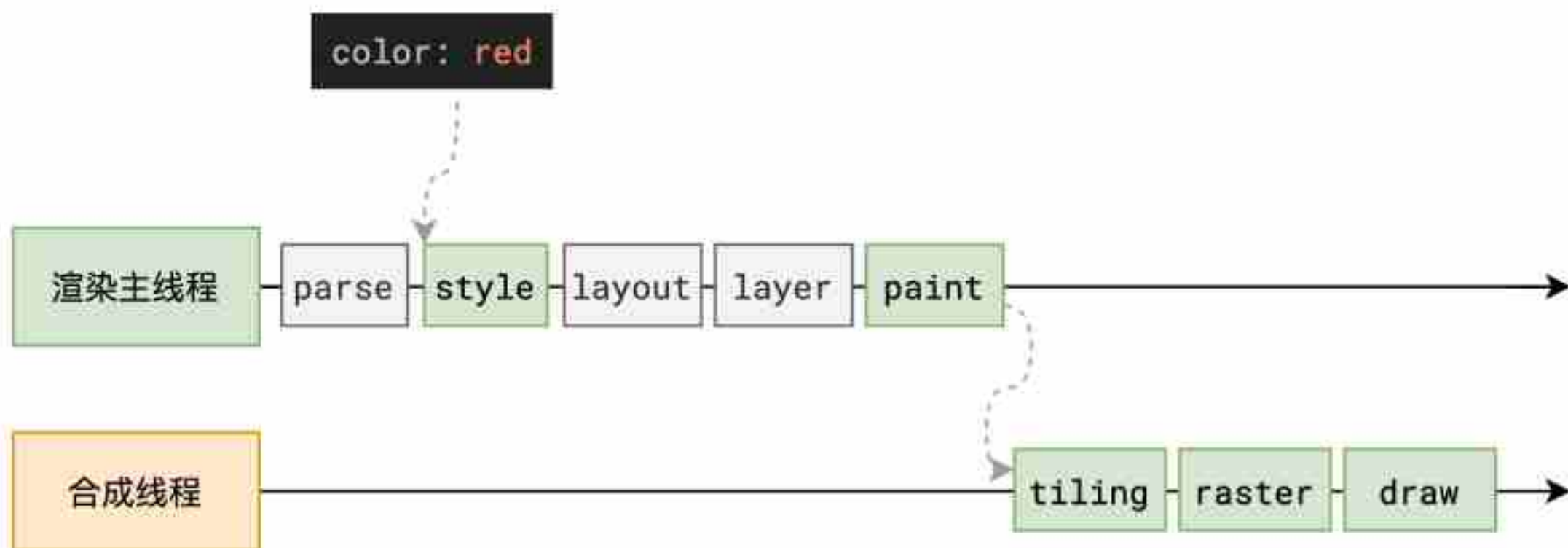
常见面试题

什么是 reflow ?



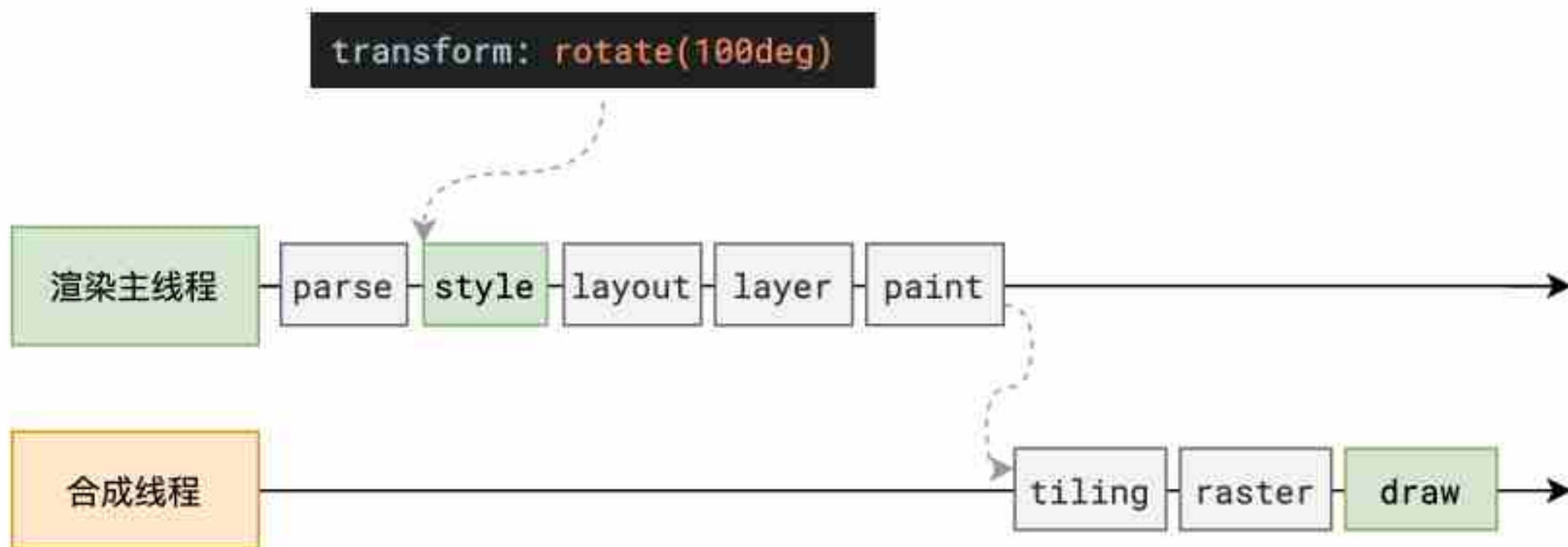
常见面试题

什么是 repaint ?



常见面试题

为什么 transform 效率高?



常见面试题

为什么 transform 效率高?

