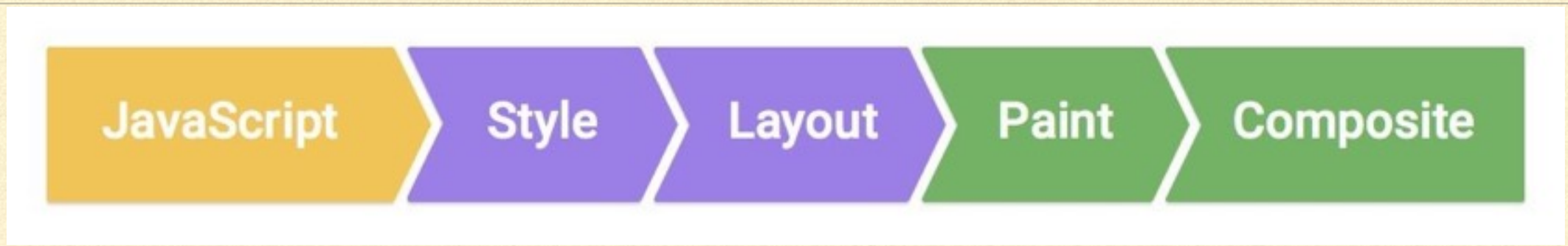
渲染原理与高性能动画

前端工程师 刘斌

如何实现动画

- jquery animation: setTimeout, top/left
- animation, transition
- transform
- requestAnimationFrame
- GPU acceleration

浏览器渲染原理

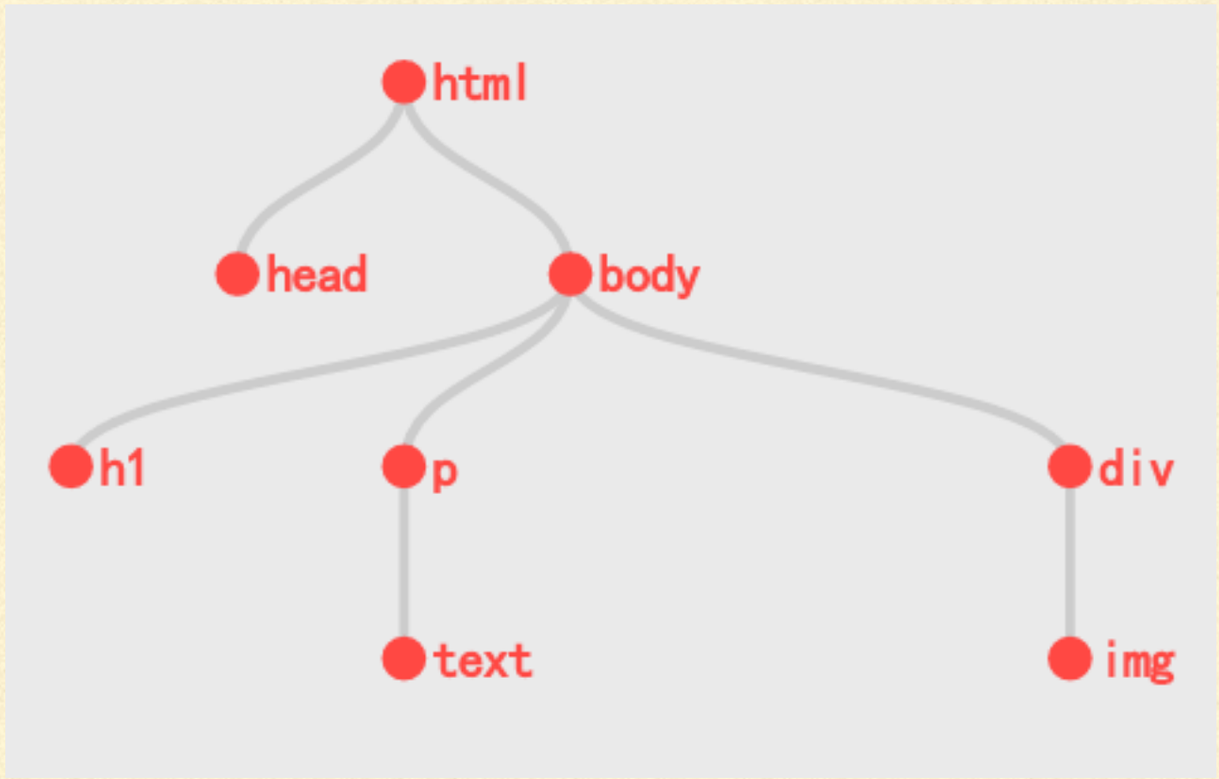


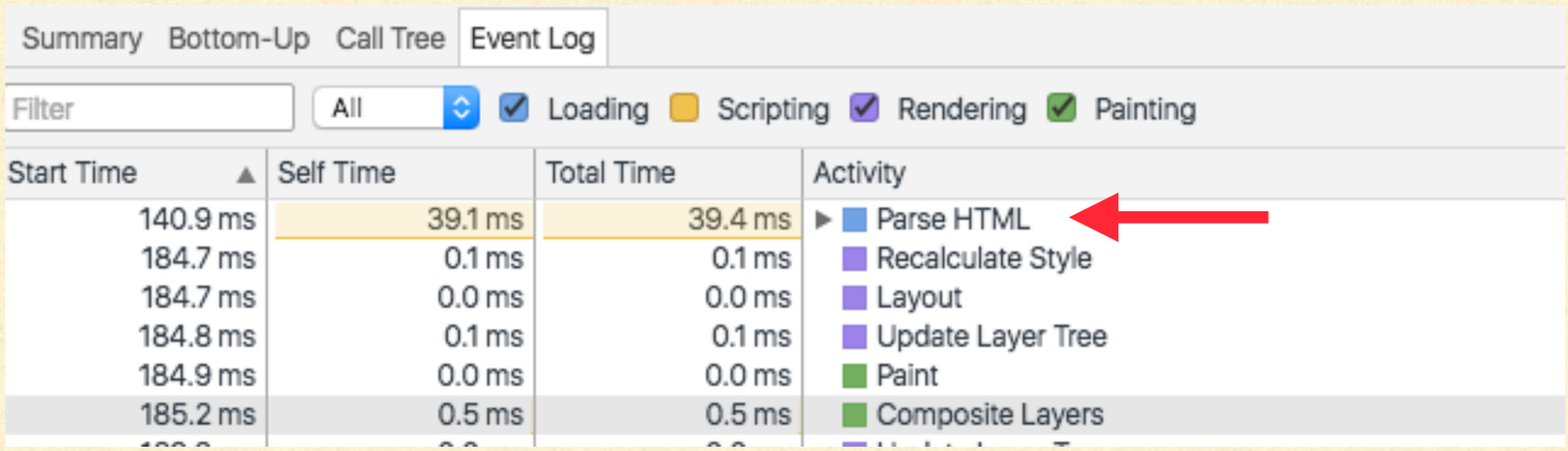
- **JavaScript**。JS对元素样式的修改
- **计算样式**。DOM元素匹配对应的CSS样式
- **布局**。计算每个DOM元素最终在屏幕上显示的大小和位置
- **绘制**。本质上就是填充像素的过程。包括绘制文字、颜色、图像、边框和阴影等
- **渲染层合并**。浏览器将所有层按照合理顺序合并成一个图层，然后显示在屏幕上

HTML

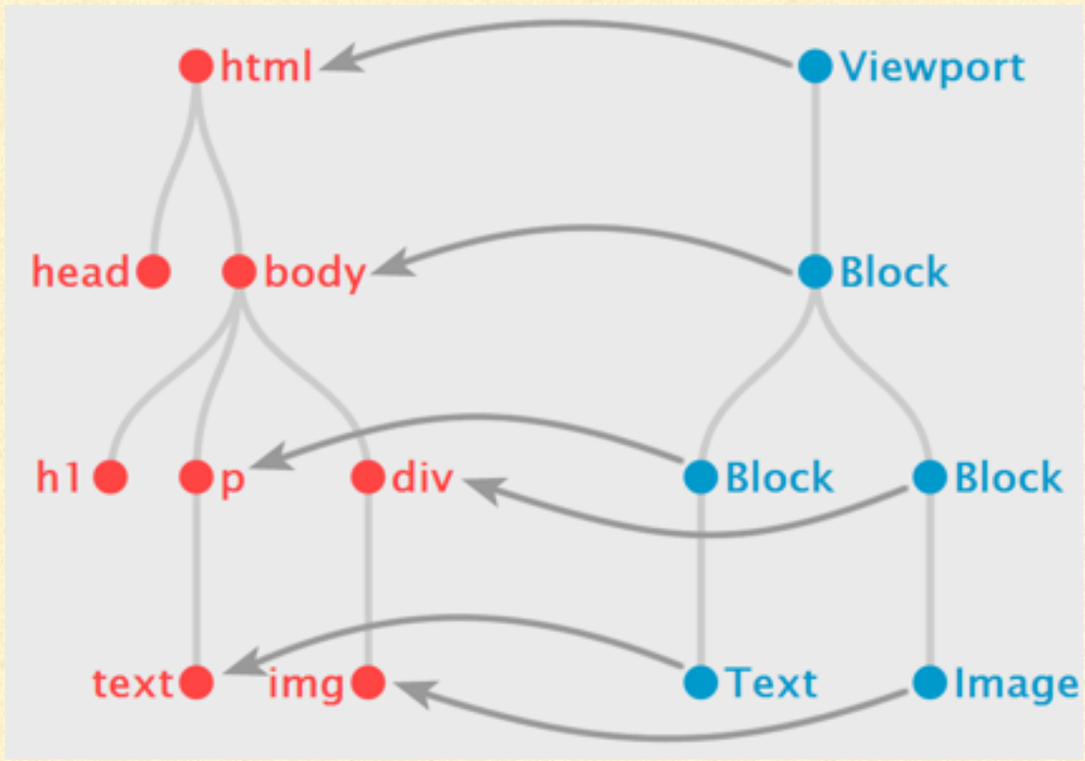
```
<html>
    <body>
        <h1 style="display:none">Article</h1>
        <p>Hi, I'm Banana</p>
        <div>
            <img src="example.png" alt="" />
        </div>
    </body>
</html>
```

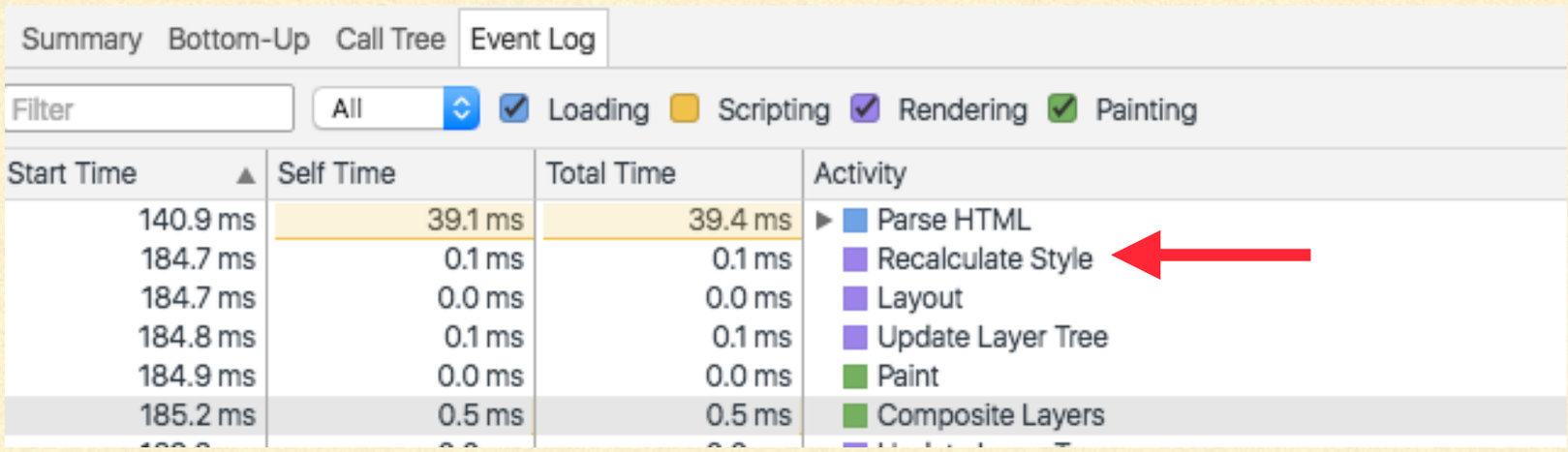
DOM


html
head
body
h1
p
div
text
img


Summary
Bottom-Up
Call Tree
Event Log
Filter
All
Loading
Scripting
Rendering
Painting
Start Time
Self Time
Total Time
Activity
140.9 ms
39.1 ms
39.4 ms
Parse HTML
184.7 ms
0.1 ms
0.1 ms
Recalculate Style
184.7 ms
0.0 ms
0.0 ms
Layout
184.8 ms
0.1 ms
0.1 ms
Update Layer Tree
184.9 ms
0.0 ms
0.0 ms
Paint
185.2 ms
0.5 ms
0.5 ms
Composite Layers

DOM + CSS = RENDER TREE





LAYOUT

Summary | Bottom-Up | Call Tree | Event Log

Filter | All | Loading | Scripting | Rendering | Painting

Start Time	Self Time	Total Time	Activity
140.9 ms	39.1 ms	39.4 ms	Parse HTML
184.7 ms	0.1 ms	0.1 ms	Recalculate Style
184.7 ms	0.0 ms	0.0 ms	Layout
184.8 ms	0.1 ms	0.1 ms	Update Layer Tree
184.9 ms	0.0 ms	0.0 ms	Paint
185.2 ms	0.5 ms	0.5 ms	Composite Layers

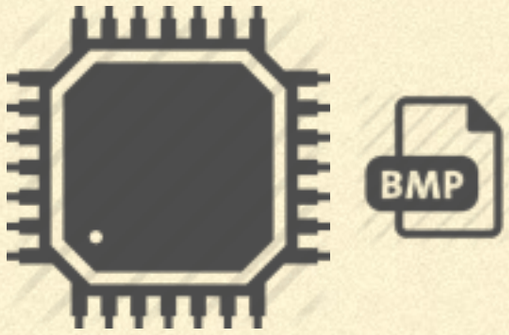
PAINT

Summary | Bottom-Up | Call Tree | Event Log

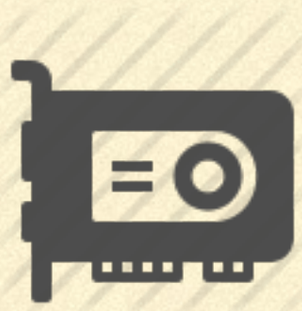
Filter | All | Loading | Scripting | Rendering | Painting

Start Time	Self Time	Total Time	Activity
140.9 ms	39.1 ms	39.4 ms	▶ Parse HTML
184.7 ms	0.1 ms	0.1 ms	Recalculate Style
184.7 ms	0.0 ms	0.0 ms	Layout
184.8 ms	0.1 ms	0.1 ms	Update Layer Tree
184.9 ms	0.0 ms	0.0 ms	Paint
185.2 ms	0.5 ms	0.5 ms	Composite Layers

COMPOSITE

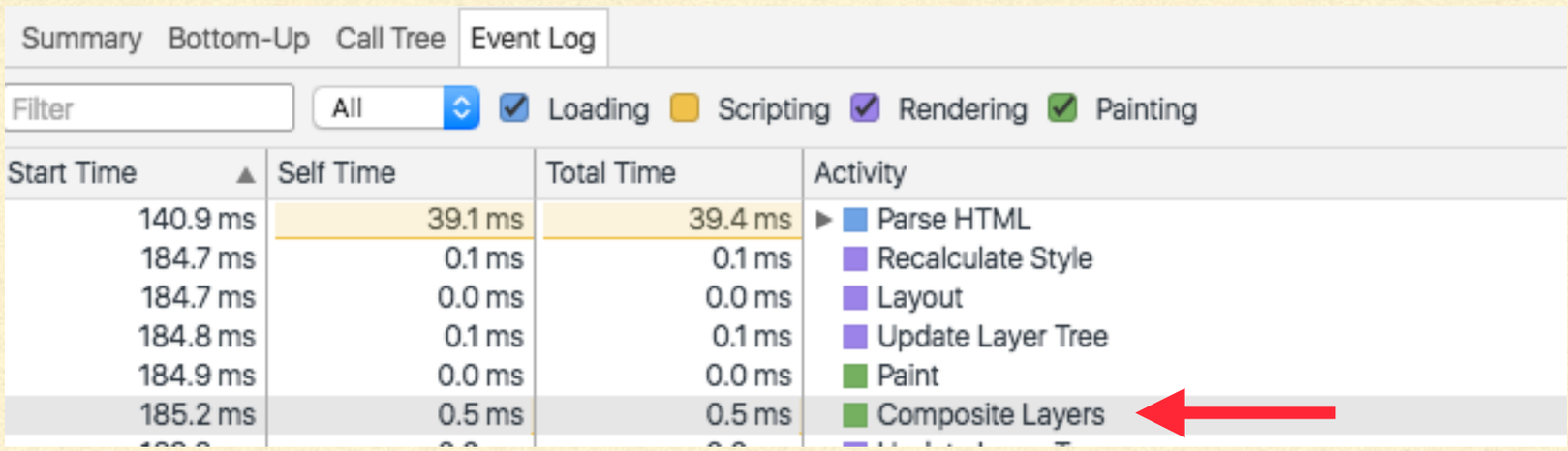


CPU



GPU

传输到GPU，渲染到屏幕



渲染三阶段

- Layout，Paint，Composite
- 修改不同CSS属性会触发不同阶段
- 触发的阶段越前，渲染的代价越高

影响LAYOUT的样式

写

width	height
padding	margin
display	border-width
border	top
position	font-size
float	text-align
overflow-y	font-weight
overflow	left
font-family	line-height
vertical-align	right
clear	white-space
bottom	min-height

读

clientHeight	clientLeft
clientTop	clientWidth
getClientRects()	getBoundingClientRect()
innerText	outerText
offsetHeight	offsetLeft
offsetParent	offsetTop
offsetWidth	focus()
scrollByLines()	scrollByPages()
scrollIntoView()	scrollIntoViewIfNeeded()
scrollHeight	scrollWidth
scrollLeft	scrollTop

影响PAINT的样式

color	border-style
visibility	background
text-decoration	background-image
background-position	background-repeat
outline-color	outline
outline-style	border-radius
outline-width	box-shadow
background-size	

COMPOSITE

- GPU也是有限度的，不要滥用GPU资源生成不必要的Layer
- 留意意外生成的Layer

CSS属性对动画的影响

CSS Property	% of pages		layout	paint	composite
opacity	40.20%				
webkit-transform	12.37%				
visibility	5.76%				
left	5.76%				
background-color	4.33%				
width	4.17%				
color	3.17%				
box-shadow	2.28%				
top	2.27%				
webkit-box-shadow	1.86%				
height	1.57%				

GPU硬件加速

- animation, transform 以及 transition 不会自动开启GPU加速
- 启动GPU开关：最显著特征是元素的3D变换
- transform: translateZ(0)， transform: translate3d(0, 0, 0)

屏幕刷新频率

- 30FPS+感觉流畅，60FPS更舒服完美
- 不同国家屏幕刷新频率规格不同，有的国家的为60FPS, 部分国家，如UK，为50FPS
- 大部分都是60FPS
- 在1 / 60FPS, 约等于16.7ms内，我们要把这一帧准备好。

绘制时机

setTimeout(callback, 1/60);

渲染时机

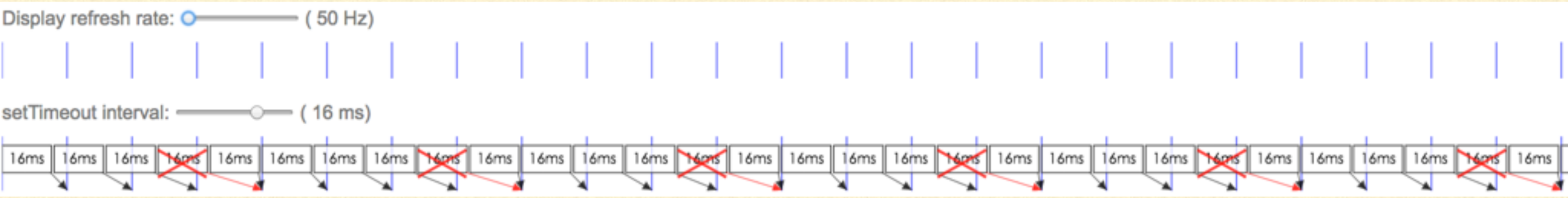
- requestAnimationFrame

- 目标： 16ms-

SETTIMEOUT不够精确

- 依靠浏览器内置时钟更新频率. eg. IE8及以前更新间隔为15.6, setTimeout 16.7, 它需要两个15.6ms才触发。超过14.5ms

- main thread队列



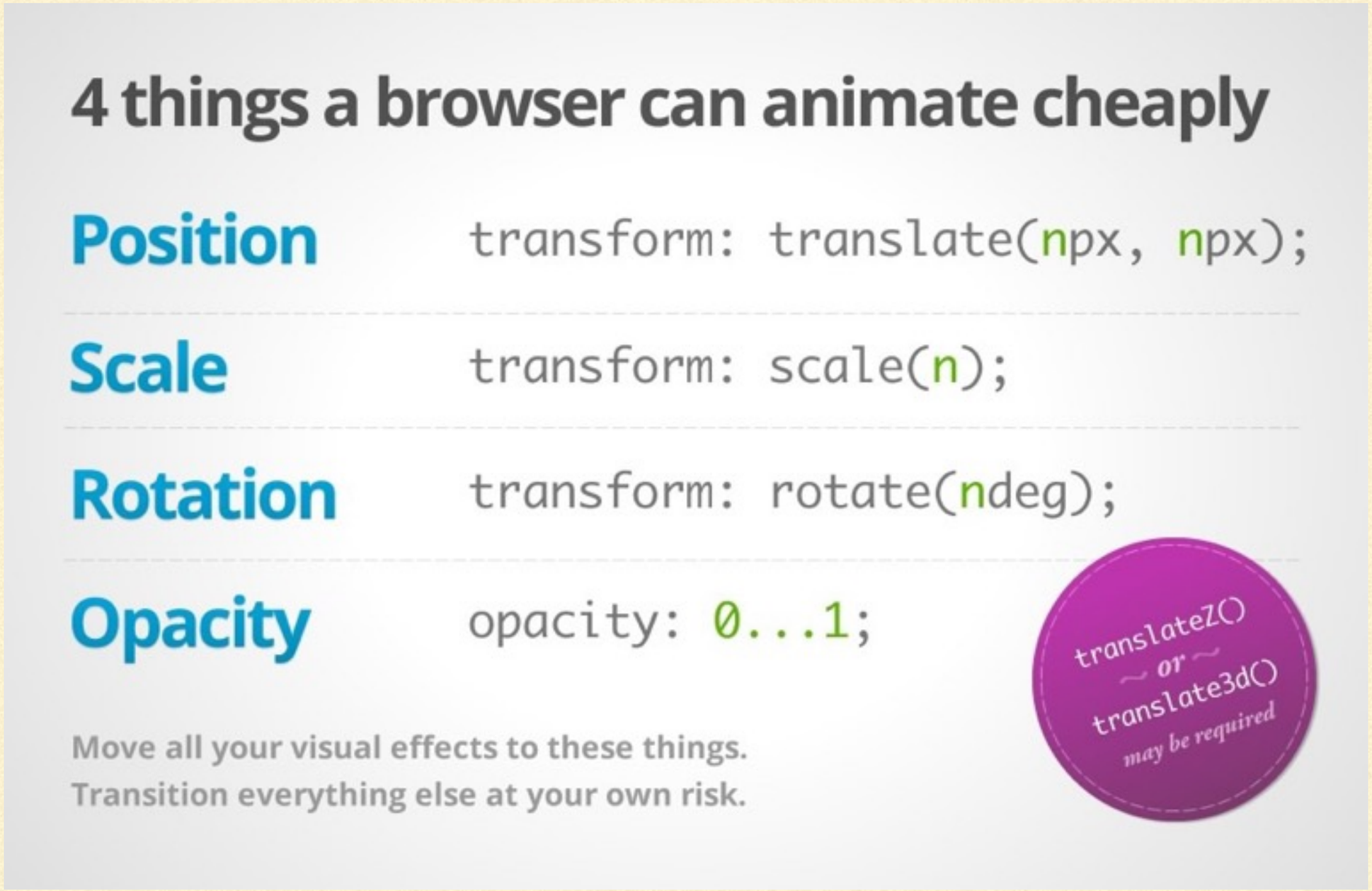
REQUESTANIMATIONFRAME

- 定义绘制每一帧前的工作。 requestAnimationFrame(callback)
- 自动调节频率。callback工作太多无法在一帧内完成，会自动降低为30FPS, 虽然频率降低但比丢帧好。

性能优化

- 是否导致layout：将动画元素absolute化以避免影响文档树，造成大面积重新计算layout
- 是否启用硬件加速
- 是否是有高消耗的属性（css shadow、background-attachment: fixed等）
- repaint的面积：缩小动画面积

四种属性的动画，消耗成本较低



用户输入事件

- touchmove
- mousemove
- scroll

参考资料

- http://www.html5rocks.com/en/tutorials/speed/high-performance-animations/
- https://developers.google.com/web/fundamentals/performance/rendering/
- http://jankfree.org/

谢谢