

前端周会分享

尚妆 ShowJoy.com
我的美丽 我的态度

PHPEExcel

Mac 安装java 7遇到的问题与解决

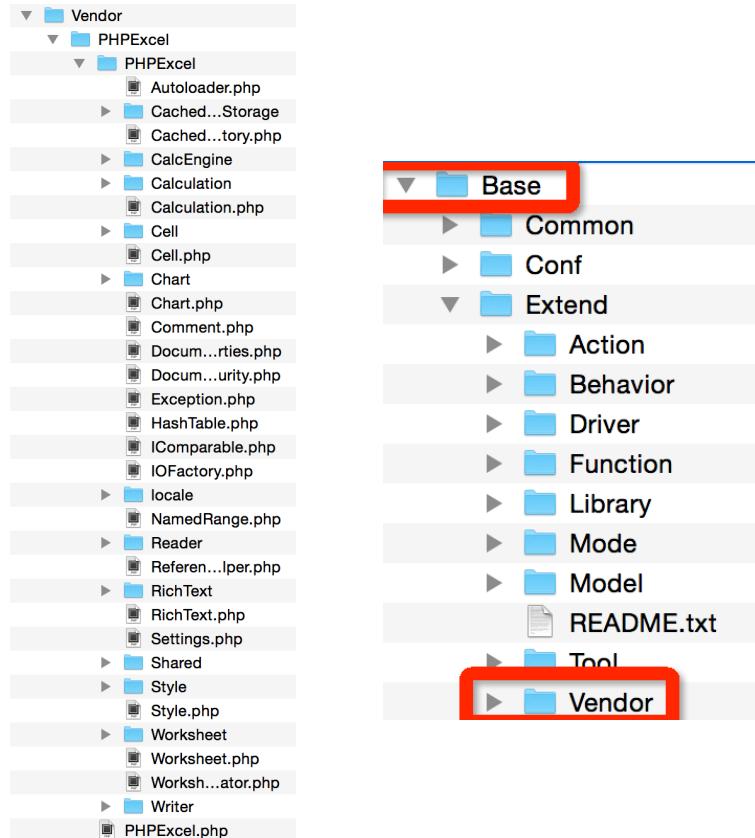
详解CSS3动画

CSS3动画 vs js动画 (简介layer树)

2015年8月14日

PHPEexcel - OpenXML - Read, Write and Create Excel documents in PHP

Ecma Office Open XML (“Open XML”）是针对字处理文档、演示文稿和电子表格的国际化开放标准，可免费供多个应用程序在多个平台上实现。Microsoft Office (2007、2003、XP、2000) 、OpenOffice Novell Edition、开源项目 Gnumeric、Neo-Office 2.1 和 PalmOS (Dataviz) 已经支持 Open XML。




```
//excel代码开始:  
Vendor("PHPEexcel.PHPEexcel");  
$objPHPEexcel = new PHPEexcel();  
$objWriter = new PHPEexcel_Writer_Excel2007($objPHPEexcel);  
$Year = explode("-", $consoleTime)[0];  
$Month = explode("-", $consoleTime)[1];  
$excelTitle = $Year.'年'.$Month.'月请假导出';  
$objPHPEexcel->setActiveSheetIndex(0);  
$objPHPEexcel->getActiveSheet()->setTitle($excelTitle);  
$objPHPEexcel->getActiveSheet()->setCellValue('A1', $excelTitle);  
$objPHPEexcel->getActiveSheet()->getStyle('A1')->getFont()->setSize(20);  
$objPHPEexcel->getActiveSheet()->getStyle('A1')->getFont()->setBold(true);  
$objPHPEexcel->getActiveSheet()->mergeCells('A1:A02');
```

简单方便的操作方式，可以容易的创建需要的Excel文件，或者通过类库中的方法来对Excel文件进行操作。

Mac 安装java 7遇到的问题

```
elvisdeMacBook-Pro:~ elvis$ java -version
java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)
```

java - version
查看java的版本

在安装java 7时遇到的不合理的问题:
mac x Yosemide
(10.10) 下安装
jdk 1.7 (jdk 1.8) 要求Mac OS X10.7.3
或更高版本解决



Mac 安装java 7遇到的问题



获取pkg
安装包
及其地
址。

```
elvisdeMacBook-Pro:~ elvis$ pkgutil
```

File Commands:

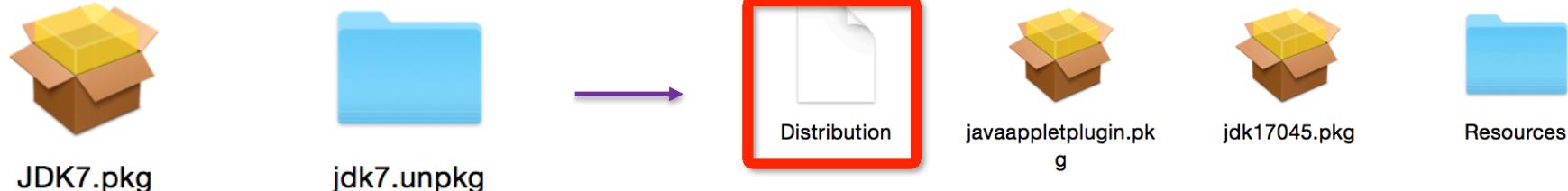
--expand PKG DIR
--flatten DIR PKG
--bom PATH
--payload-files PATH

Expand the flat package PKG to DIR
Flatten the files at DIR as PKG
Extract any Bom files from the pkg at PATH into /tmp
List the paths archived within the (m)pkg at PATH

解压安装包：
pkgutil命令
的用法

Mac 安装java 7遇到的问题

```
elvisdeMacBook-Pro:~ elvis$ pkgutil --expand /Users/elvis/Documents/java/JDK 7 Update 45.  
pkg /Users/elvis/Documents/java/jdk7.unpkg_
```



```
function pm_install_check() {  
    if(!(checkForMacOSX('10.7.3') == true)) {  
        my.result.title = 'OS X Lion required';  
        my.result.message = 'This Installer is supported only on OS X 10.7.3 or Later.';  
        my.result.type = 'Fatal';  
        return false;  
    }  
    return true;  
}
```

```
function pm_install_check() {  
    return true;  
}
```

```
elvisdeMacBook-Pro:~ elvis$ pkgutil --flatten /Users/elvis/Documents/java/jdk7.unpkg/ /U  
sers/elvis/Documents/java/jdk7-1.pkg _
```

CSS3动画介绍

transform (变形)

transition (转换)

animation (动画)

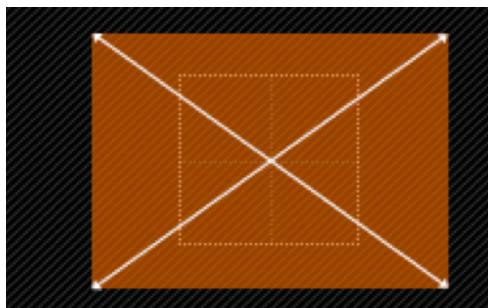
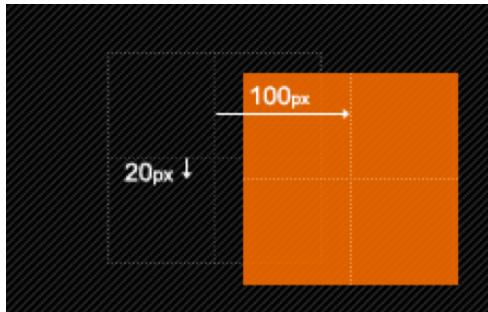
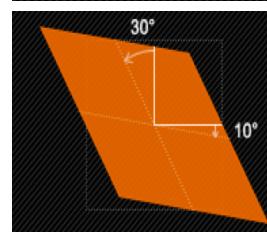
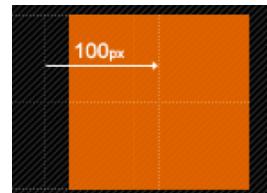
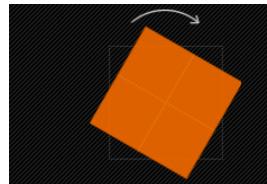
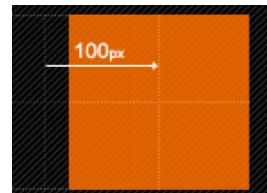
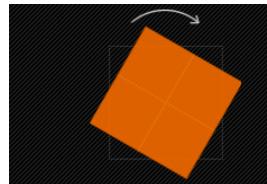
Rotate (旋转)

Skew (扭曲)

Scale (缩放)

Translate (移动)

Matrix (矩阵变形)



```
transform : none | <transform-function> [ <transform-function> ] *
```

当我们使用多种变换的效果的时候采用的是空格来进行间隔。

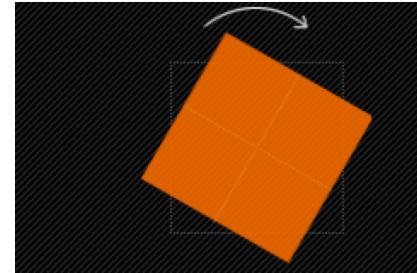
Transform -- 变换

一、旋转 (rotate)

rotate(<angle>)

<angle>为正的时候就是顺时针旋转，如果是负数的时候就是逆时针旋转。

transform:rotate(30deg)



二、移动 (translate)

translate(x,y) translateX(x) translateY(Y)

这种变换的方式和我们采用js进行top和left进行移动的方式是差不多的，在介绍完动画的后面将会进行js动画与css动画上面的一些对比。

Transform -- 变换

三、缩放 (scale) → x,y指定了对应方向的放大倍数

scale(x,y)

scaleX(x)

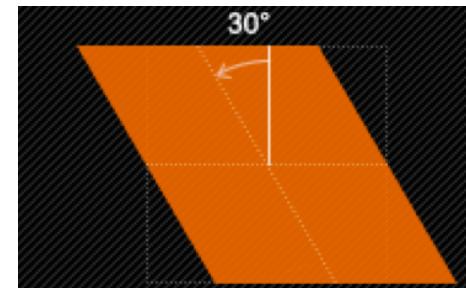
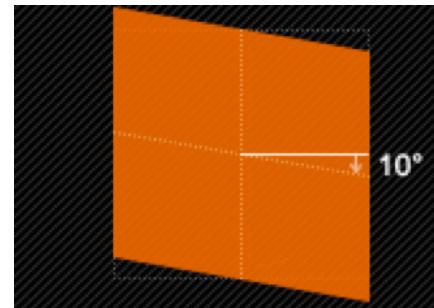
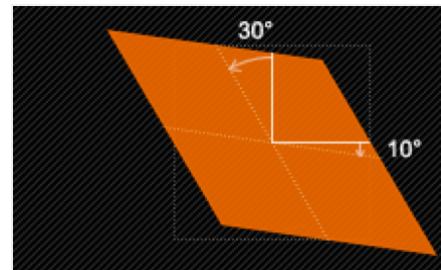
scaleY(Y)

三、扭曲 (skew)

transform:skew(30deg,10deg)

transform:skewX(30deg)

transform:skewY (10deg)



Transform-origin

进行transform动作之前可以改变元素的基点位置,默认基点就是其中心位置

1、top left | left top 等价于 0 0 | 0% 0%

2、top | top center | center top 等价于 50% 0

3、right top | top right 等价于 100% 0

4、left | left center | center left 等价于 0 50% | 0% 50%

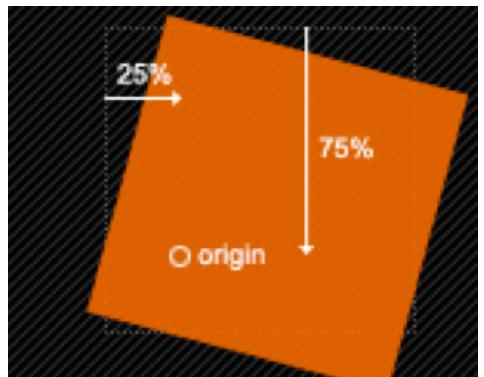
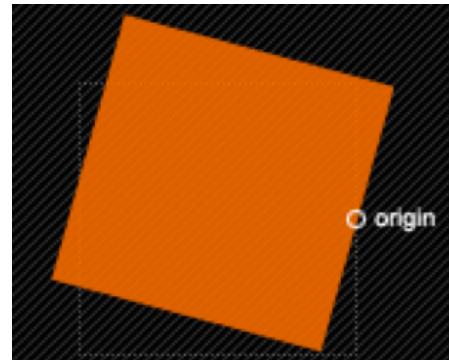
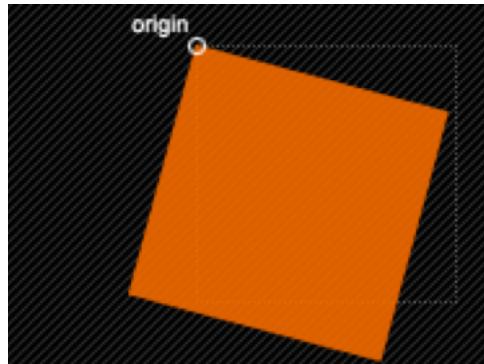
5、center | center center 等价于 50% 50% (默认值)

6、right | right center | center right 等价于 100% 50%

7、bottom left | left bottom 等价于 0 100% | 0% 100%

8、bottom | bottom center | center bottom 等价于 50% 100%

9、bottom right | right bottom 等价于 100% 100%



```
//Mozilla内核浏览器: firefox 3.5+
-moz-transform-origin: x y;
//Webkit内核浏览器: Safari and Chrome
-webkit-transform-origin: x y;
//Opera
-o-transform-origin: x y ;
//IE9
-ms-transform-origin: x y;
//W3C标准
transform-origin: x y ;
```

transition

css的transition允许css的属性值在一定的时间区间内平滑地过渡。这种效果可以在鼠标单击、获得焦点、被点击或对元素任何改变中触发，并圆滑地以动画效果改变CSS的属性值。

```
transition : [<'transition-property'> || <'transition-duration'> || <'transition-timing-function'> || <'transition-delay'> [, [<'transition-property'> || <'transition-duration'> || <'transition-timing-function'> || <'transition-delay'>]]*
```

transition-property

transition-duration

transition-timing-function

transition-delay

transition

一、 **transition-property** 当元素其中一个属性改变时执行**transition**效果

属性值: **none all indent,...**

1、color: 通过红、绿、蓝和透明度组件变换 (每个数值处理) 如 : background-color, border-color, color, outline-color 等css属性 ;

2、length: 真实的数字 如 : word-spacing, width, vertical-align, top, right, bottom, left, padding, outline-width, margin, min-width, min-height, max-width, max-height, line-height, height, border-width, border-spacing, background-position 等属性 ;

3、percentage: 真实的数字 如 : word-spacing, width, vertical-align, top, right, bottom, left, min-width, min-height, max-width, max-height, line-height, height, background-position 等属性 ;

...

并非所有的属性改变都会触发此条性质，需要注意！例子： **ghost button**

<http://www.w3.org/TR/css3-transitions/#properties-from-css->

transition

二、 **transition-duration** 用来指定元素转换过程的持续时间

可以采用s（秒）或者ms（毫秒）这两种方式来定义持续的时间。

三、 **transition-timing-function** 根据时间的推进去改变属性值的变换速率

1、 ease : (逐渐变慢) 默认值，ease函数等同于贝塞尔曲线(0.25, 0.1, 0.25, 1.0).

2、 linear : (匀速) , linear 函数等同于贝塞尔曲线(0.0, 0.0, 1.0, 1.0).

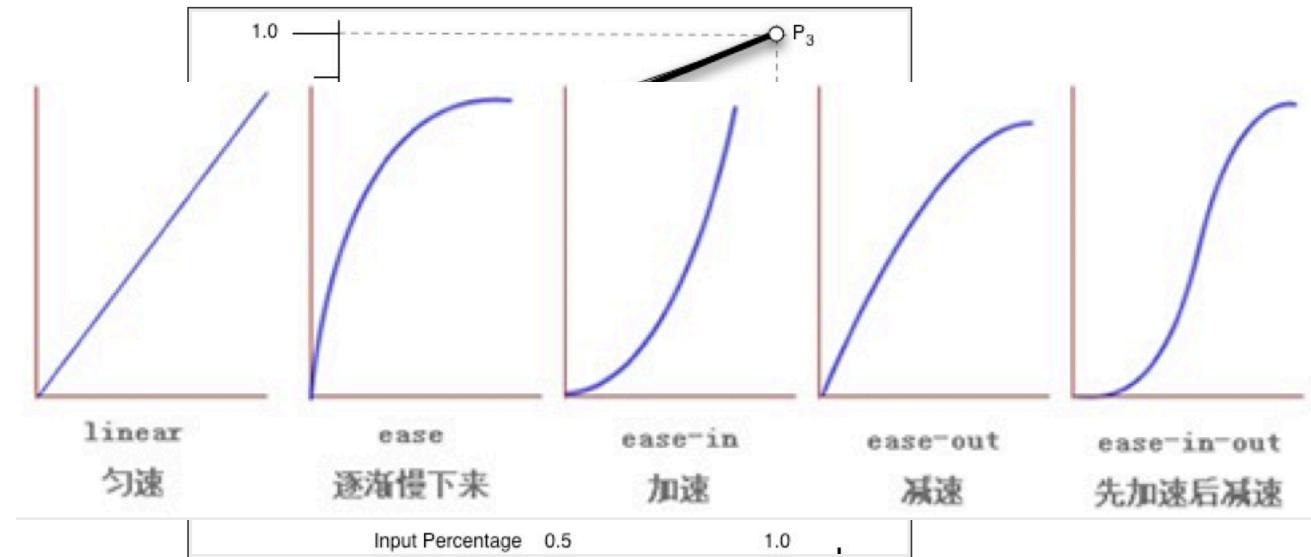
3、 ease-in : (加速) , ease-in 函数等同于贝塞尔曲线(0.42, 0, 1.0, 1.0).

4、 ease-out : (减速) , ease-out 函数等同于贝塞尔曲线(0, 0, 0.58, 1.0).

5、 ease-in-out : (加速然后减速) , ease-in-out 函数等同于贝塞尔曲线(0.42, 0, 0.58, 1.0)

6、cubic-bezier：（该值允许你去自定义一个时间曲线），特定的cubic-bezier曲线。 $(x1, y1, x2, y2)$ 四个值特定于曲线上点P1和点P2。所有值需在 $[0, 1]$ 区域内，否则无效。

其是cubic-bezier为通过贝塞尔曲线来计算“转换”过程中的属性值，如下曲线所示，通过改变P1($x1, y1$)和P2($x2, y2$)的坐标可以改变整个过程的Output Percentage。初始默认值为default.



Animation (兼容性)

@keyframes -- 关键帧

vs transition

```
@keyframes IDENT {  
0% { Properties:Properties value; }  
Percentage { Properties:Properties value; }  
100% { Properties:Properties value; }  
}
```

相比于**transition**更加精确的控制

transition需要一些事件来进行触发，而**animation**是不需要通过触发来完成的。

```
-webkit-animation-name: 'wobble'; /*动画属性名，也就是我们前面keyframes定义的动画名*/  
-webkit-animation-duration: 10s; /*动画持续时间*/  
-webkit-animation-timing-function: ease-in-out; /*动画频率，和transition-timing-function是一样的*/  
-webkit-animation-delay: 2s; /*动画延迟时间*/  
-webkit-animation-iteration-count: 10; /*定义循环次数，infinite为无限次*/  
-webkit-animation-direction: alternate; /*定义动画方式*/
```

Normal: 按照正常向前的顺序进行播放； **alternate:** 偶数次向前，奇数次反向

CSS3动画 VS js动画

Chromium

Chromium 是 Google 的[chrome](#)浏览器背后的引擎，其目的是为了创建一个安全、稳定和快速的通用浏览器。



使用Chromium开源代码（基于webkit内核）的浏览器有[360极速浏览器](#)、[枫树浏览器](#)、[太阳花浏览器](#)、[世界之窗极速版](#)、[傲游浏览器](#)和[UC浏览器](#)电脑版等。[搜狗高速浏览器](#)和[qq浏览器](#)官网未提及Chromium，只是说采用webkit内核，经网友测试这两款浏览器极有可能也是使用的Chromium，只是官方不承认而已。

渲染进程： main thread 和 compositor thread

- ◆ 如果CSS动画只是改变transforms和opacity，这时整个CSS动画得以在compositor thread完成
- ◆ 如果采用js，这时则会在main thread执行，然后触发compositor进行下一步操作

此时如果main thread繁忙，采用CSS动画可以在compositor中进行，则会更流畅

CSS3动画 VS js动画

在Chromium基础上的浏览器中：

- ◆ JS在执行一些昂贵的任务
- ◆ 同时CSS动画不触发layout或paint



CSS动画更流畅

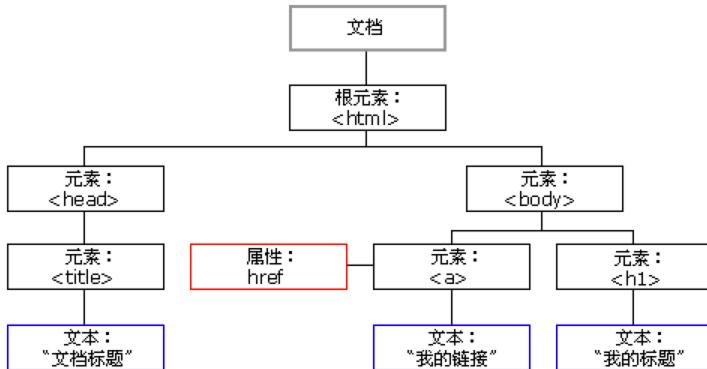
在CSS动画或JS动画触发了paint或layout时，需要main thread进行Layer树的重计算，这时CSS动画或JS动画都会阻塞后续操作。

- ◆ 是否导致layout
- ◆ repaint的面积
- ◆ 是否是有高消耗的属性（css shadow等）
- ◆ 是否启用硬件加速

大部分场景下更关注右面的这些问题：

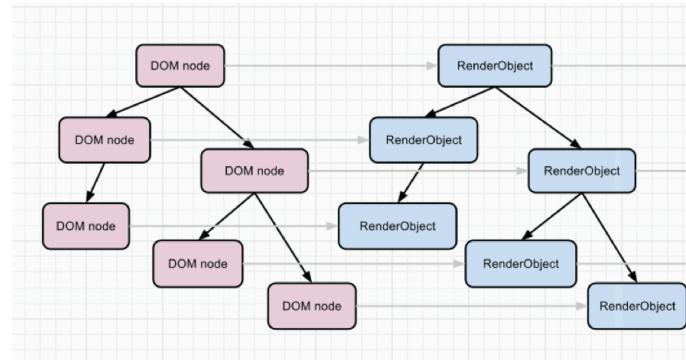
Layer树（涉及页面的加载的过程）

HTML → DOM树



经常采用js来对DOM树上的元素进行操作。

但是 DOM 树本身并不能直接用于排版和渲染，内核会生成另外一棵树： Render树。



**Render 树是
浏览器排版
引擎的主要
作业对象**

DOM + CSS → Render Tree

Render 树是衔接浏览器排版引擎和渲染引擎之间的桥梁，它是排版引擎的输出，渲染引擎的输入。

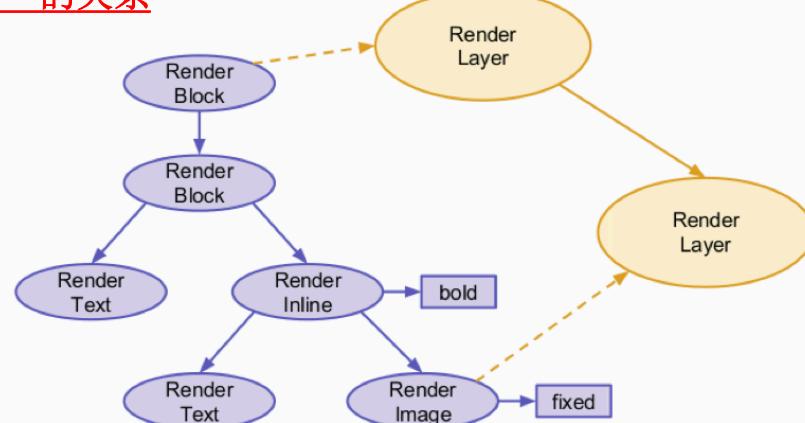
Layer树（涉及页面的加载的过程）

浏览器渲染引擎并不是直接使用 Render 树进行绘制

渲染引擎会为一些特定的 **RenderObject** 生成对应的 **RenderLayer**

LayerTree

多对一的关系



在软件渲染下，通常各个 **RenderLayer** 的内容都绘制在同一块后端存储上。在 **GPU** 硬件加速的下，某些 **RenderLayer** 可能有自己独立的后端存储，而后通过合成器来把这些不同的后端合成在一起，最终形成网页的可视化内容。

Layer树（涉及页面的加载的过程）

哪些情况下的**RenderObject**节点需要建立新的**RenderLayer**节点呢？

- DOM树的document节点对应的RenderView节点
- DOM树中的document 的子女节点，也就是HTML节点对应的RenderBlock节点
- 显式的CSS位置
- 有透明效果的对象
- 节点有溢出（overflow），alpha或者反射等效果的
- Canvas 2D和3D (WebGL)
- Video节点对应的RenderObject对象

Layer树决定了网页绘制的层次顺序，而从属于 RenderLayer 的 RenderObject 决定了这个 Layer 的内容，所有的 RenderLayer 和 RenderObject 一起就决定了网页在屏幕上最终呈现出来的内容。

CSS3动画 VS js动画

在Chromium基础上的浏览器中：

- ◆ JS在执行一些昂贵的任务
- ◆ 同时CSS动画不触发layout或paint



CSS动画更流畅

在CSS动画或JS动画触发了paint或layout时，需要main thread进行Layer树的重计算，这时CSS动画或JS动画都会阻塞后续操作。

- ◆ 是否导致layout
- ◆ repaint的面积
- ◆ 是否是有高消耗的属性（css shadow等）
- ◆ 是否启用硬件加速

大部分场景下更关注右面的这些问题：

CSS3动画 VS js动画

其他浏览器？（旧~）

Animated properties	JS-based Animation更快	CSS-based Animation更快
top, left, width, height	Windows Surface RT, iPhone 5s (iOS7), iPad 3 (iOS 6), iPad 3 (iOS7), Samsung Galaxy Tab 2, Chrome, Firefox, Safari, Opera, Kindle Fire HD, IE11	(none)
translate, scale	Windows Surface RT, iPhone 5s (iOS7), iPad 3 (iOS7), Samsung Galaxy Tab 2, Firefox, Opera, IE11	iPad 3 (iOS6), Safari, Chrome

CSS3动画的效率问题，还是一个有待支持与提高的一个技术。

CSS3动画 VS js动画

对比总结

- 功能涵盖面来说，js比CSS3要多许多，很多功能在精度上或者是更多方面的支持上js可以表现出更加强大的一面。
- 实现的难度来讲，CSS3的动画更加容易实现，在某些机器上面的性能CSS3还可以做到自然降级，而js则没有那么智能化的表现。
- CSS3的兼容性还是有待加强的。

感谢关注
THANKS

尚妆 ShowJoy.com
我的美丽 我的态度