专题八: CSS工作原理和性能优化

> 大纲目录：
> - CSS布局模型：
>   - 流动模型（Flow）
>   - 浮动模型（Float）
>   - 层模型（Layer）
> - BFC / IFC：
>   - BFC的原理和功能
>   - IFC的原理和功能
>   - 理解font-size、line-height、vertical-align
> - CSS渲染和解析原理
> - CSS性能优化：
>   - 提升CSS渲染性能的方面汇总
>   - absolute/display隐藏与回流等性能
>   - base64:URL背景图片与web页面性能优化

CSS布局模型

流动模型(Flow)

> 含义：

- 流动(Flow）是默认的网页布局模式。也就是说网页在默认状态下的 HTML 网页元素都是根据流动模型来分布网页内容的。

特征:

- 块状元素都会在所处的包含元素内自上而下按顺序垂直延伸分布，因为在默认状态下，块状元素的宽度都为100%。实际上，块状元素都会以行的形式占据位置。
- 在流动模型下，内联元素都会在所处的包含元素内从左到右水平分布显示（内联元素可不像块状元素这么霸道独占一行）

浮动模型(Float)

- 通过css的float属性可以将元素设置为浮动元素。元素浮动之后不再占据原来的位置，它们会尽可能的往包裹它们的父元素的左边框或右边框靠，并会在它们元素所处位置的下面产生浮动流，影响下面的元素定位。
- 浮动元素并没有完全脱离文档流，它只是从包裹它的盒子中浮动起来并尽可能远的往左侧或者右侧进行移动。
- 浮动设计的初衷是为了实现文字在图片周围的环绕效果。

层模型(Layer)

含义：

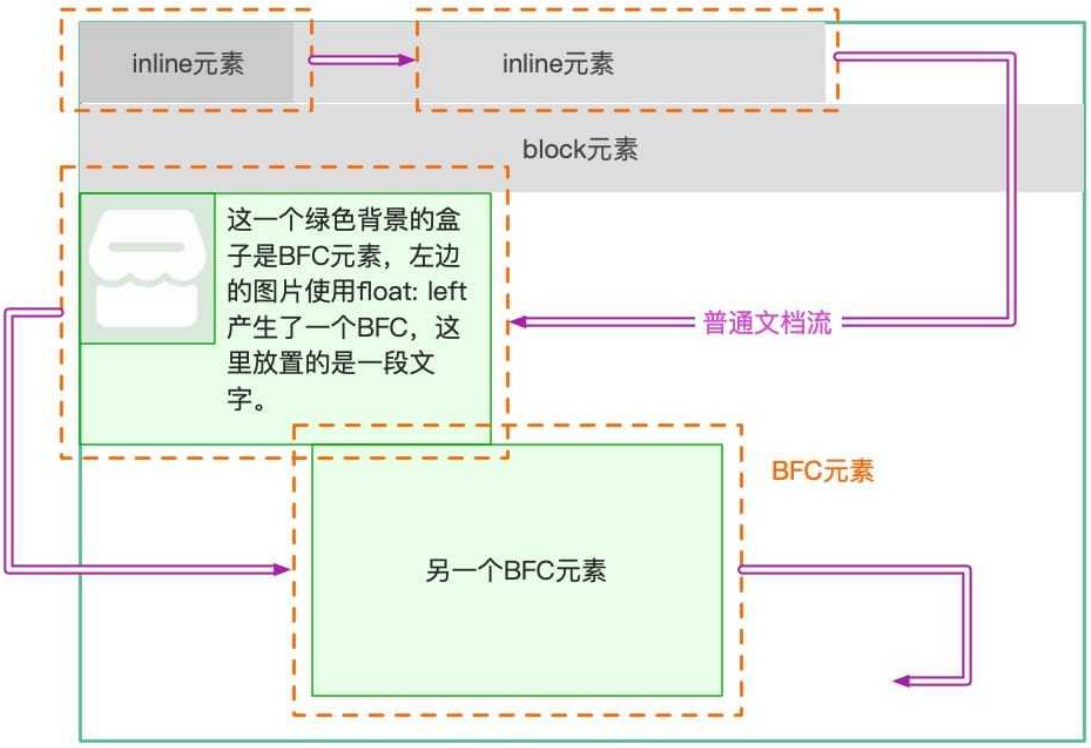
- 层布局模型就像是图像软件PhotoShop中非常流行的图层编辑功能一样，每个图层能够精确定位操作，但在网页设计领域，由于网页大小的活动性，层布局没能受到热捧。但是在网页上局部使用层布局还是有其方便之处的

层模型有三种形式：

- 相对定位（position: relative）
- 绝对定位（position: absolute）
- 固定定位（position: fixed）

BFC的原理和功能

其实BFC是上面三种布局方式中的普通流，BFC会产生一个独立的容器，该容器内部的元素不会在布局上影响到外部的元素，在外部的普通流看来它和其他普通流元素无差别，文档最终会按照上面说的普通流计算布局。



https://developer.mozilla.org/zh-CN/docs/Web/Guide/CSS/Block_formatting_context

BFC的含义

- MDN的定义：块格式化上下文（Block Formatting Context，BFC） 是Web页面的可视化CSS渲染的一部分，是块盒子的布局过程发生的区域，也是浮动元素与其他元素交互的区域。
- BFC(block formatting context)块级格式化上下文，它是页面中的一块渲染区域，并且有一套属于自己的渲染规则，它决定了元素如何对齐内容进行布局，以及与其他元素的关系和相互作用。 当涉及到可视化布局的时候，BFC提供了一个环境，HTML元素在这个环境中按照一定规则进行布局；
- 具有BFC特性的元素可以看做是隔离了的独立容器，容器里面的元素不会在布局上影响到外面的元素，并且BFC具有普通容器所没有的的一些特性。
- BFC是一个独立的布局环境，BFC内部的元素布局与外部互不影响。

BFC的布局规则

- 内部的盒子会在垂直方向，一个个地放置；
- 盒子垂直方向的距离由margin决定，属于同一个BFC的两个相邻Box的上下margin会发生重叠；
- 每个元素的左边，与包含的盒子的左边相接触，即使存在浮动也是如此；
- BFC的区域不会与float box重叠；
- BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素，反之也如此；
- 计算BFC的高度时，浮动元素也参与计算；

触发BFC的条件

只要元素满足下面任一条件即可触发BFC特性：

- html根元素或其他包含它的元素；

- float的属性不为none；
- overflow为auto、scroll、hidden；
- display为inline-block，table-cell，table-caption中的任何一个；
- position为absolute或fixed；

元素或属性	属性值
根元素	
float	left、right
position	absolute、fixed
overflow	auto、scroll、hidden
display	inline-block、table-cell

知乎 @九九欧

BFC的用处

案例一：让浮动内容和周围的内容等高

- overflow:auto 创建一个会包含这个浮动的 BFC，通常的做法是设置父元素 overflow: auto 或者设置其他的非默认的 overflow: visible 的值。overflow: auto 或者设置其他的非默认的 overflow: visible 的值。
- 使用display: flow-root; 一个新的 display 属性的值，它可以创建无副作用的 BFC。在父级块中使用 display: flow-root 可以创建新的 BFC。

案例二： 外边距折叠

- 原因: Box垂直方向的距离由margin决定。属于同一个BFC的两个相邻Box的margin会发生重叠
- 解决方法: 给上box或者下box任意一个包裹新的box并开启BFC
- 原理: BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素，反之也如此；

案例三： 清除浮动

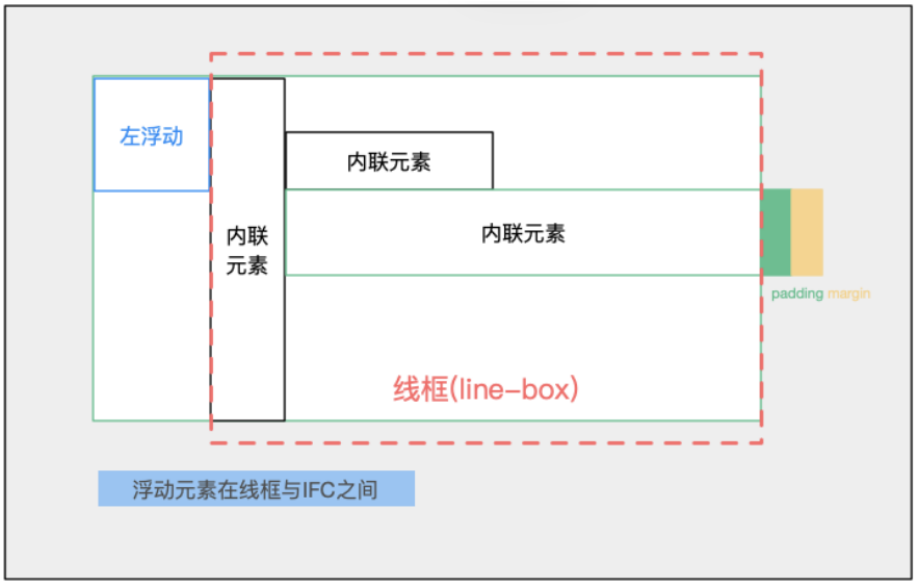
- 解决方法: 给父元素开启BFC
- 原理: 计算BFC的高度时，浮动子元素也参与计算

案例四： 自适应的两列布局(左图右文)

- 解决方法: 给父元素开启BFC
- 原理: BFC的区域不会与float box重叠；

IFC的原理和功能

存在块级格式化上下文BFC，则对应存在内联格式化上下文IFC、网格格式化上下文GFC、自适应格式化上下文FFC，这些都可以统称为格式化上下文；



IFC的含义:

- IFC(inline Formatting Context）叫做“内联格式化上下”
- 内部的元素从包含块的顶部开始，从左至右(默认)排列成一行形成的一个矩形盒子叫做line box；

IFC的布局规则:

- 盒子是水平一个接一个的排列，水平的margin，内边距，边框是可以有的。
- 垂直方向的对齐，可能是底部对齐，顶部对齐，也可能是基线对齐（这个是默认的）
- 行框中的内联盒子的高度小于行框的高度时，内联盒子的垂直方向的对齐方式取决于vertical-align属性。
- 当一个行框水平不能容纳内联盒子时，他们将会在垂直方向上产生多个行框，他们上下一个挨着一个，但是不会重叠。
- 一般来说，行框的左边界紧挨着包含容器的左边界，行框的右边界紧挨着包含容器的右边界，（是两个边都紧挨着）。然而，浮动盒子可能存在于包含边框边界和行框边界之间。
- 多个内联盒子的宽度小于包含他们的行框时，他们在水平方向的分布取决于text-align属性。

IFC的作用:

- 水平居中：当一个块要在环境中水平居中时候，设置其为inline-block则会在外层产生IFC，通过text-align:center则可以使其水平居中。
- 垂直居中：创建一个IFC，用其中一个元素撑开父元素的高度，然后设置其vertical-align:middle,其他行内元素则可以在此父元素下垂直居中。

理解font-size、line-height、vertical-align

https://www.zhangxinxu.com/wordpress/2015/08/css-deep-understand-vertical-align-and-line-height/
https://www.cnblogs.com/10yearsmanong/p/13084706.html

图片下面默认的留白:

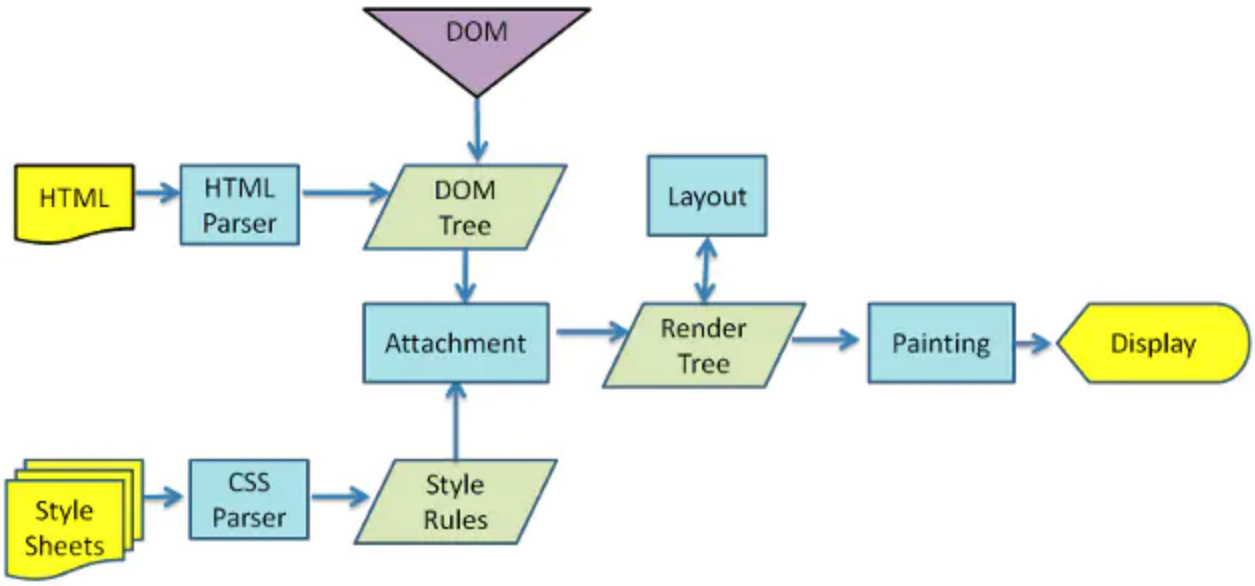
- 让vertical-align失效-图片设置display或者浮动、绝对定位
- 使用其他vertical-align值(bottom/middle/top)
- 直接修改line-height值
- line-height为相对单位，font-size间接控制

CSS渲染和解析原理

浏览器渲染原理

1. 浏览器在接收到服务器返回的html页面后，
2. 浏览器开始构建DOM树DOM TREE，遇到CSS样式会构建CSS规则树CSS RULE TREE，
3. 遇到javascript会通过DOM API和CSSDOM API来操作DOM Tree和CSS Rule Tree，解析完成后，
4. 浏览器引擎会通过DOM Tree 和CSS Rule Tree 来构造 Rendering Tree（渲染树），
5. 最后，渲染树构建完成后就是 "布局" 处理，也就是确定每个节点在屏幕上的确切显示位置
6. 下个步骤（渲染之后），开始 "绘制"，遍历渲染树，并用UI后端层，将每一个节点绘制出来。

整个流程如下图所示



CSS渲染规则

CSS的渲染规则，是从上到下，从右到左渲染的。

```
1 .main h4 a { font-size: 14px; }
```

渲染过程是这样的：首先先找到所有的 a，沿着 a 的父元素查找h4，然后再沿着 h4，查找.main。中途找到了符合匹配规则的节点就加入结果集。如果找到根元素的 html 都没有匹配，则这条路径不再遍历。下一个 a 开始重复这个查找匹配，直至没有a继续查找。
浏览器的这种查找规则是为了尽早过滤掉一些无关的样式规则和元素。

CSS性能优化

CSS常见性能优化方法汇总

- 尽量避免类似.a.b{} .list a{}以及其他一些复杂选择器，以提高整站整体CSS渲染。
- 避免某些expression表达式，避免IE6的AlphaImageLoader png透明滤镜，可以试试使用fireworks生成png8 alpha透明（目前photoshop只有png8 索引透明）。
- 适当定高。
- 图片设定不响应重绘的尺寸，如果你的<img>不设定尺寸、同时外部容器没有定死高宽，则图片在首次载入时候，占据空间会从0到完全出现，左右上下都可能位移，发生大规模的重绘。可以使用width/height控制，或者在CSS中设置。
- <textarea>或者使用<script type="text/html">存储动态载入HTML或模板HTML，降低首屏加载的渲染时间。
- 具有复杂动画的元素绝对定位-脱离文档流，避免强烈的回流。现代浏览器可以渐进使用CSS3 transition实现动画效果，比改变像素值来的高性能。
- 不使用iframe，据说开销最大的DOM元素。
- 不要使用类选择器和ID选择器修饰元素标签，例如：p#id1 {color:red;}
- 保持简单，不要使用嵌套过多过于复杂的选择器，避免深层次,嵌套层级不要超过三级；
- 通配符和属性选择器效率最低，需要匹配的元素最多，尽量避免使用。
- 通常将浏览器前缀置于前面，将标准样式属性置于最后，例如：-moz-border-radius: 5px;border-radius: 5px;
- 减少CSS文件体积：移除空的CSS规则、值为0不需要单位,复合属性使用缩写、属性值为浮动小数0.*,可以省略小数点之前的0、不给h1-h6元素定义过多的样式；
- 把 Stylesheets放在HTML页面头部，不要使用 @import；
- 减少使用昂贵的属性,如box-shadow/border-radius/filter/透明度/:nth-child等；
- 优化重排与重绘:减少重排、避免不必要的重绘；
- 小图标的处理方案：cssSprite、字体图标font-face、base64编码；

absolute/display隐藏与回流等性能

> https://www.zhangxinxu.com/wordpress/2013/01/absolute-display-visibility-reflow/

base64:URL背景图片与web页面性能优化

> https://www.zhangxinxu.com/wordpress/2012/04/base64-url-image-%e5%9b%be%e7%89%87-%e9%a1%b5%e9%9d%a2%e6%80%a7%e8%83%bd%e4%bc%98%e5%8c%96/
>
> 优点：
> - 减少了HTTP请求
> - 某些文件可以避免跨域的问题
> - 没有图片更新要重新上传，还要清理缓存的问题