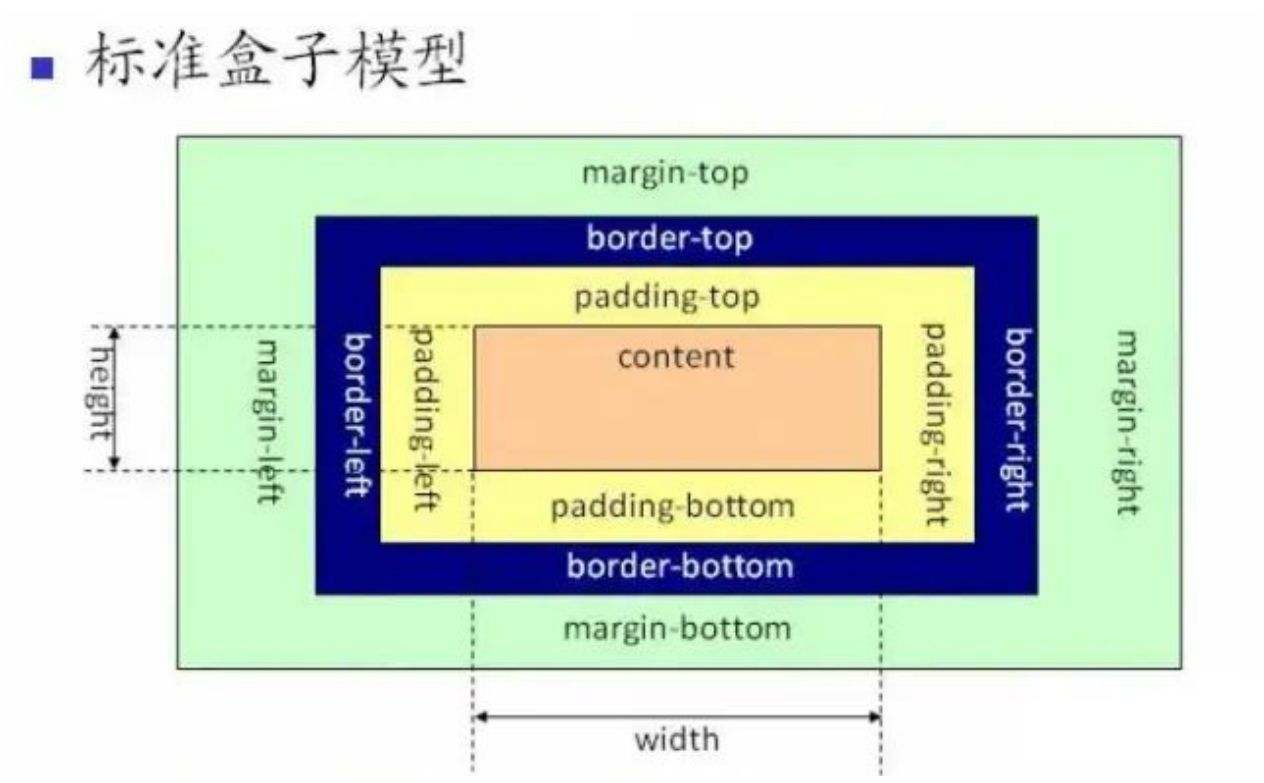
你需掌握的CSS知识都在这了（长文建议收藏，文末有福利）


爱前端不爱恋爱
微信公众号：web前端学习圈，关注领取全套50G前端系统教程
关注她

1.CSS盒模型，在不同浏览器的差异

css 标准盒子模型

css盒子模型 又称为框模型（Box Model），包含了元素内容（content）、内边距（padding）、边框（border）、外边距（margin）几个要素。如下图：



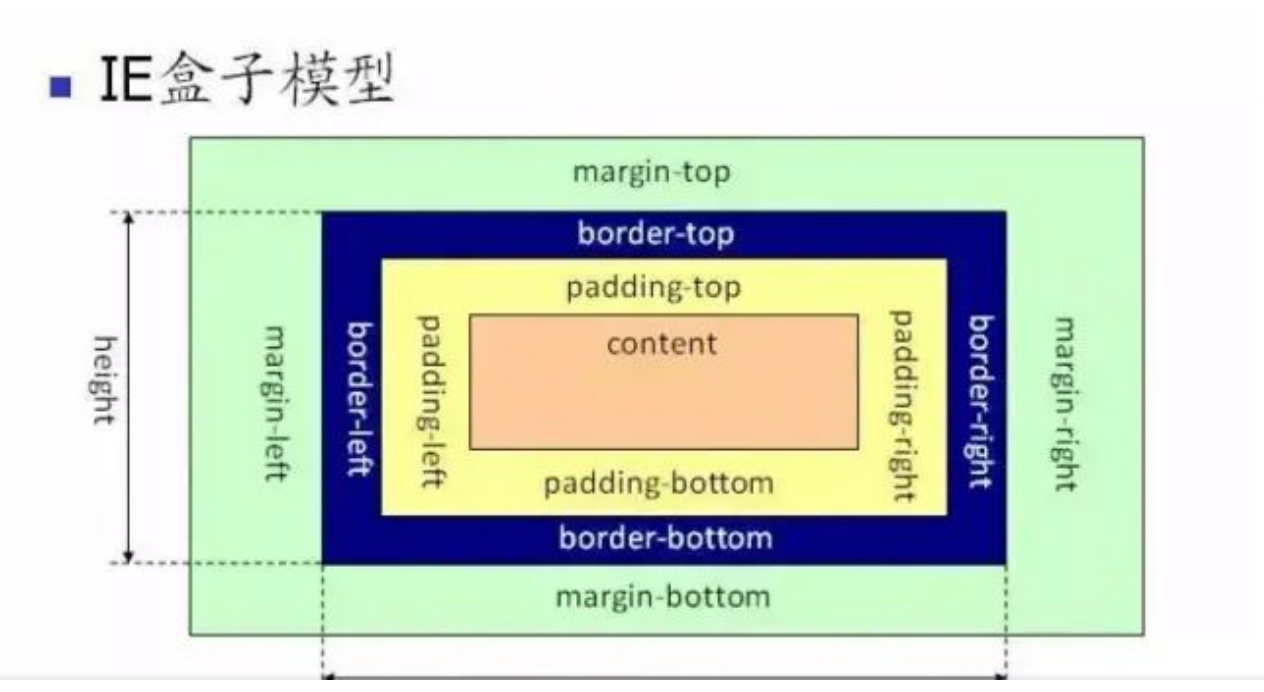
图中的内层是content依次是padding border margin。通常我们设置背景时就是内容、内边距、边框这三部分，如果border设置颜色的时候会显示boder颜色当boder颜色是透明时会显示background-color的颜色。而该元素的子元素的是从content开始的。而外边距是透明的，不会遮挡其他元素。

元素框的总宽度=width+padding-left+padding-right+border-left+border-right+margin-left+margin-right;

元素框的总高度=height+padding-top+padding-bottom+border-top+border-bottom+margin-top+margin-bottom;

IE盒子模型

IE盒子模型如下图：



元素框的总宽度=width（padding-left+padding-right+border-left+border-right);

元素框的总高度=height（padding-top+padding-bottom+border-top+border-bottom）;

两个模型宽度和高度的计算（是不一样的）

w3c中的盒子模型的宽:包括margin+border+padding+width;
width:margin*2+border*2+padding*2+width;
height:margin*2+border*2+padding*2+height;

iE中的盒子模型的width:包括border+padding+width;

上面的两个宽度相加的属性是一样的。因此我们应该选择标准盒子模型，在网页的顶部加上 DOCTYPE 声明。

2.CSS所有选择器及其优先级、使用场景，哪些可以继承，如何运用at规则

css选择器种类有：

通用选择器：*

id选择器：#header{}

class选择器：.header{}

元素选择器：div{}

子选择器：ul > li{}

后代选择器：div p{}

伪类选择器：:hover、::selection、.action、:first-child、:last-child、:first-of-type、:last-of-type、:nth-of-type(n)、:nth-of-last-type(n)等,例如a:hover{}

伪元素选择器: :after、:before等,例如：li:after

属性选择器: input[type="text"]

组合选择器：E,F/E F（后代选择器）/E>F（子元素选择器）/E+F（直接相邻元素选择器----匹配之后的相邻同级元素）/E~F（普通相邻元素选择器----匹配之后的同级元素）

层次选择器：p~ul 选择前面有p元素的每个ul元素

css选择器优先级：

- 选择器优先级由高到低分别为：
  !important > 作为style属性写在元素标签上的内联样式 >id选择器>类选择器>伪类选择器>属性选择器>标签选择器> 通配符选择器（* 应少用）>浏览器自定义;
- 当比较多个相同级别的CSS选择器优先级时，它们定义的位置将决定一切。下面从位置上将CSS优先级由高到低分为六级：
  1、位于<head/>标签里的<style/>中所定义的CSS拥有最高级的优先权。
  2、第二级的优先属性由位于 <style/>标签中的 @import 引入样式表所定义。
  3、第三级的优先属性由<link/>标签所引入的样式表定义。
  4、第四级的优先属性由<link/>标签所引入的样式表内的 @import 导入样式表定义。
  5、第五级优先的样式有用户设定。
  6、最低级的优先权由浏览器默认。

使用场景：

- class使用场景：需要某些特定样式的标签则放在同一个class中，需要此样式的标签可再添加此

CSS哪些属性可以继承?
css继承特性主要是指文本方面的继承(比如字体、颜色、字体大小等)，盒模型相关的属性基本没有继承特性。

不可继承的：
display、margin、border、padding、background、height、min-height、max-height、width、min-width、max-width、overflow、position、top、bottom、left、right、z-index、float、clear、 table-layout、vertical-align、page-break-after、page-bread-before和unicode-bidi。

所有元素可继承的：
visibility和cursor

终极块级元素可继承的：
text-indent和text-align

内联元素可继承的：
letter-spacing、word-spacing、white-space、line-height、color、font、font-family、font-size、font-style、font-variant、font-weight、text-decoration、text-transform、direction

列表元素可继承的：
list-style、list-style-type、list-style-position、list-style-image

常用at规则及使用示例：

- @charset
- @import
- @namespace
- @document
- @font-face
- @keyframes
- @media
- @page
- @supports

```
/*定义字符集*/
@charset "utf-8"
/*导入css文件*/
@import "base.css"
/*自定义字体*/
@font-face {}
/*声明CSS3 animation动画关键帧*/
@keyframes fadeIn {}
/*媒体查询*/
@media{}
```

3.CSS伪类和伪元素有哪些，它们的区别和实际应用

伪类的例子有：

:hover

:active

:first-child

:visited

:first-line

:first-letter

:after

:before

伪类和伪元素的根本区别在于：

它们是否创造了新的元素(抽象)。从我们模仿其意义的角度来看，如果需要添加新元素加以标识的，就是伪元素，反之，如果只需要在既有元素上添加类别的，就是伪类。

伪元素在一个选择器里只能出现一次，并且只能出现在末尾;

伪类则是像真正的类一样发挥着类的作用，没有数量上的限制，只要不是相互排斥的伪类，也可以同时使用在相同的元素上。

实际使用：

伪类用一个冒号表示 :first-child
伪元素则使用两个冒号表示 ::first-line

4.CSS几种定位的规则、定位参照物、对文档流的影响，如何选择最好的定位方式，雪碧图实现原理

1）、static定位(普通流定位) ---------- 默认定位

2）、float定位(浮动定位) 例：float:left;

有两个取值：left(左浮动)和right(右浮动)。浮动元素会在没有浮动元素的上方，效果上看是遮挡住了没有浮动的元素，有float样式规则的元素是脱离文档流的，它的父元素的高度并不能有它撑开。

3）、relative定位(相对定位) position:relative;

相对本元素的左上角进行定位，top,left,bottom,right都可以有值。虽然经过定位后，位置可能会移动，但是本元素并没有脱离文档流，还占有原来的页面空间。可以设置z-index。使本元素相对于文档流中的元素，或者脱离文档流但是z-index的值比本元素的值要小的元素更加靠近用户的视线。

相对定位最大的作用是为了实现某个元素相对于本元素的左上角绝对定位，本元素需要设置position为relative。

4）、absolute定位(绝对定位) position:absolute;

相对于祖代中有relative(相对定位)并且离本元素层级关系上是最近的元素的左上角进行定位，如果在祖代元素中没有有relative定位的，就默认相对于body进行定位。

绝对定位是脱离文档流的，与浮动定位是一样的效果，会压在非定位元素的上方。可以设置z-index属性。

雪碧图实现原理：

CSS雪碧的基本原理是把你的网站上用到的一些图片整合到一张单独的图片中，从而减少你的网站的HTTP请求数量。该图片使用CSS background和background-position属性渲染，这也就意味着你的标签变得更加复杂了，图片是在CSS中定义，而非<img>标签。

5.写出尽可能多的水平垂直居中的方案并对比它们的优缺点

如果不是，则先将其父元素设置为块级元素，再给父元素设置 **text-align: center**

块级元素水平居中(定宽度)：

1）需要谁居中，给其设置 margin: 0 auto; （作用：使盒子自己居中）

2) 首先设置父元素为相对定位，再设置子元素为绝对定位，设置子元素的**left:50%**，即让子元素的左上角水平居中；

设置绝对子元素的 **margin-left: -元素宽度的一半px;** 或者设置**transform: translateX(-50%);**

块级元素水平居中(不宽度)：

1) 默认子元素的宽度和父元素一样，这时需要设置子元素为display: inline-block; 或 display: inline;即将其转换成行内块级/行内元素，给父元素设置 text-align: center;

2) 首先设置父元素为相对定位，再设置子元素为绝对定位，设置子元素的**left:50%**，即让子元素的左上角水平居中；

利用css3新增属性**transform: translateX(-50%);**

使用flexbox布局实现水平居中（宽度定不定都可以）：

使用flexbox布局，只需要给待处理的块状元素的父元素添加属性 display: flex; justify-content: center;

单行的行内元素垂直居中:

只需要设置单行行内元素的"行高等于盒子的高"即可；

多行的行内元素垂直居中:

使用给父元素设置**display:table-cell;和vertical-align: middle;**属即可；

块级元素垂直居中方法一：使用定位

首先设置父元素为相对定位，再设置子元素为绝对定位，设置子元素的**top: 50%**，即让子元素的左上角垂直居中；

定高度：设置绝对子元素的 **margin-top: -元素高度的一半px;** 或者设置**transform: translateY(-50%);**

不定高度：利用css3新增属性**transform: translateY(-50%);**

块级元素垂直居中方法二：使用flexbox布局实现（高度定不定都可以）

使用flexbox布局，只需要给待处理的块状元素的父元素添加属性 **display: flex; align-items: center;**

水平垂直居中-已知高度和宽度的元素：

方法一：

设置父元素为相对定位，给子元素设置绝对定位，**top: 0; right: 0; bottom: 0; left: 0; margin: auto;**

方法二：

设置父元素为相对定位，给子元素设置绝对定位，**left: 50%; top: 50%; margin-left: --元素宽度的一半px; margin-top: --元素高度的一半px;**

水平垂直居中-未知高度和宽度的元素：

方案二：使用flex布局实现

设置父元素为flex定位，**justify-content: center; align-items: center;**

6.BFC的布局规则，实现原理，可以解决的问题

BFC直译为**块级格式化上下文**，它是一个独立的渲染区域，只有Block-level box参与，它规定了**内部的Block-level Box**如何布局，并且与外部毫不相干。
注意：可以把BFC理解为一个大的盒子，其内部是由Block-level box组成的

BFC布局规则：

1. 内部的Box会在垂直方向，一个接一个地放置。
2. 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
3. 每个元素的margin box的左边， 与包含块border box的左边相接触(对于从左往右的格式化，否则相反)。即使存在浮动也是如此。
4. BFC的区域不会与float box重叠。
5. 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。
6. 计算BFC的高度时，浮动元素也参与计算

BFC的作用及原理：

1. 自适应两栏布局

2. 清除内部浮动

3. 防止垂直 margin 重叠

BFC 内部的元素和外部的元素绝对不会互相影响，因此， 当 BFC 外部存在浮动时，它不应该影响 BFC 内部Box的布局， BFC 会通过变窄，而不与浮动有重叠。同样的，当 BFC 内部有浮动时，为了不影响外部元素的布局， BFC 计算高度时会包括浮动的高度。避免margin重叠也是这样的一个道理。

7.CSS函数有哪些?

根据w3cplus中可以划分为以下几类：

属性函数：attr()；

背景图片函数：linear-gradient()、radial-gradient()、conic-gradient()、repeating-linear-gradient()、repeating-radial-gradient()、repeating-conic-gradient()、image-set()、image()、url()、element()；

颜色函数：rgb()、rgba()、hsl()、hsla()、hwb()、color-mod()；

图形函数：circle()、ellipse()、inset()、polygon()、path()

滤镜函数：blur()、brightness()、contrast()、drop-shadow()、grayscale()、hue-rotate()、invert()、opacity()、saturate()、sepia()；

转换函数：matrix()、matrix3d()、perspective()、rotate()、rotate3d()、rotateX()、rotateY()、rotateZ()、scale()、scale3d()、scaleX()、scaleY()、scaleZ()、skew()、skewX()、skewY()、translate()、translateX()、translateY()、translateZ()、translate3d()；

数学函数：calc()、min()、max()、mixmax()、repeat()；

8.PostCSS、Sass、Less的异同，以及使用配置，至少掌握一种

● 编译环境不一样，Sass的安装需要Ruby环境，是在服务端处理的，而Less是需要引入less.js来处理Less代码输出css到浏览器，也可以在开发环节使用Less，然后编译成css文件，直接放到项目中；

● 变量符号不一样，Less是@，而Scss是$；

● 输出设置，Less没有输出设置，Sass提供4中输出选项：nested, compact, compressed 和 expanded；

● 处理条件语句，Sass支持条件语句，可以使用if{}else{},for{}循环等等。LESS的条件语句使用有些另类，他不是我们常见的关键词if和else if之类，而其实现方式是利用关键词“when”；

● 引用外部文件，文件名如果以下划线_开头的话，Sass会认为该文件是一个引用文件，不会将其编译为css文件，ess引用外部文件和css中的@import没什么差异；

● 工具库的不同，Sass有工具库Compass, 简单说，Sass和Compass的关系有点像Javascript和jQuery的关系,Compass在Sass的基础上，封装了一系列有用的模块和模板，补充强化了Sass的功能。Less有UI组件库Bootstrap,Bootstrap是web前端开发中一个比较有名的前端UI组件库，Bootstrap的样式文件部分源码就是采用Less语法编写。

● PostCSS介绍：

PostCSS 的主要功能只有两个：第一个就是前面提到的把 CSS 解析成 JavaScript 可以操作的 AST，第二个就是调用插件来处理 AST 并得到结果。因此，不能简单的把 PostCSS 归类成 CSS 预处理或后处理工具。PostCSS 所能执行的任务非常多，同时涵盖了传统意义上的预处理和后处理。

● PostCSS使用

PostCSS 一般不单独使用，而是与已有的构建工具进行集成。PostCSS 与主流的构建工具，如 Webpack、Grunt 和 Gulp 都可以进行集成。完成集成之后，选择满足功能需求的 PostCSS 插件并进行配置。现在经常用到的是基于PostCSS的Autoprefixer插件，使用方式可以在官网的插件库进行查询。下面是官网地址：

PostCSS官网地址

9.CSS模块化方案有哪些?

css的模块化方案可能和js的一样多，下面简单介绍几种主要的模块方案。

oocss

面对对象的规则，主要的原则是两种：分离结构和外观，分离容器和内容。

名词解释

分离结构和外观：增加可重复的设计单元，同时去推进产品和ui对这方面的思考，比如下面的css使用时对象模式的命名和模块化规则。

分离容器和内容：指的是样式的使用不以元素位置唯一匹配，在任何位置你都可以使用这个样式，如果你不适用这个样式，会保持默认的样式。

实例

smacss

sma和oocss有很多类似之处，但区分的地方有很多，主要是对样式的分类。分别是：基础、布局、模块、状态、主题

基础

可以适用于任何位置，我也称全局样式

布局

主要是用来实现不同的特色布局，提高布局的复用率，

模块

设计中的模块化，可重复使用的一个单元，一般是dom+css的耦合绑定。

状态

描述在特定状态下的布局或者模块的特殊化表现，这里我觉得要推荐下《css禅意花园》，在dom结构不变的情况下，可以通过css的皮肤化实现样式的改版。

主题

与状态相比更加定制的是，我们会针对有些特殊的模块，进行主题的设置，包括一系列的颜色、尺寸、交互等进行重度设计，参数化设计。

案例

```
// dom结构<div class="toogle toogle-simple">  <div class="toogle-control is-active">
```

与oocss相比，其实大部分设计思路是一样的，以一个类作为css的作用域（作用域就是两个限制，1 不符合场景时限制禁止使用 2 符合场景时要正确的使用），另外的区别就是针对皮肤和状态的不同书写规则。

bem

bem就是块、元素、修饰符的思维去写样式。它不涉及具体的css结构，只是建议你如何命名css.

案例

```
// dom结构<div class="toogle toogle--simple">  <div class="toogle_control toogs="toogl
```

解释

- 块级：所属组件的名称
- 元素：元素在组件里的名称
- 修饰符：任何与元素修饰相关的类

style-components

使用JS编译原生的CSS文件，使其具备模块化的能力，点击CSS Modules进入github主页。

这些模块化方案都是各有优缺点，如命名约定：命名复杂、缺乏扩展、 CSS Modules当然也有一些缺点(你得先学会它再去谈优劣)。在众多解决方案中，没有绝对的优劣。还是要结合自己的场景来决定。

10.CSS如何配置按需加载

- 使用require.js按需加载CSS

```
//模块test.js
define(['css!../css/test.css'], function() { //先加载依赖样式
    var test = {};
    return test;
});

//配置
require.config({
    map: { //map告诉RequireJS在任何模块之前，都先载入这个模块
        '*': {
            css: 'lib/css'
        }
    },
    paths: {
        test: 'lib/test',
    }
});

//调用
require(['test'])
```

webpack配置CSS的按需加载

这里以ant desgin css 为例：

```
{
  test: /\.(js|mjs|jsx|ts|tsx)$/,
  include: paths.appSrc,
  loader: require.resolve('babel-loader'),
  options: {
    customize: require.resolve(
      'babel-preset-react-app/webpack-overrides'
    ),
    plugins: [
      ["import",{libraryName: "antd", style: 'css'}],   //只需加一行，手动划重点antd按需
      [
        require.resolve('babel-plugin-named-asset-import'),
        {
          loaderMap: {
            svg: {
              ReactComponent: '@svgr/webpack?-svgo,+ref![path]',
            },
          },
        },
      ],
    ],
    cacheDirectory: true,
    cacheCompression: isEnvProduction,
    compact: isEnvProduction,
  },
}
```

默认情况下，CSS 被视为阻塞渲染的资源，这意味着浏览器将不会渲染任何已处理的内容，直至 CSSOM 构建完毕。请务必精简您的 CSS，并利用媒体类型和查询来解除对渲染的阻塞。

我们可以通过 CSS“媒体类型”和“媒体查询”来解决这类用例：

```
<link href="style.css" rel="stylesheet">
<link href="print.css" rel="stylesheet" media="print">
<link href="other.css" rel="stylesheet" media="(min-width: 40em)">
```

媒体查询由媒体类型以及零个或多个检查特定媒体特征状况的表达式组成。

例如，上面的第一个样式表声明未提供任何媒体类型或查询，因此它适用于所有情况，也就是说，它始终会阻塞渲染。第二个样式表则不然，它只在打印内容时适用---或许您想重新安排布局、更改字体等等，因此在网页首次加载时，该样式表不需要阻塞渲染。最后，最后一个样式表声明提供由浏览器执行的“媒体查询”：符合条件时，浏览器将阻塞渲染，直至样式表下载并处理完毕。

12.熟练使用CSS(3)实现常见动画，如渐变、移动、旋转、缩放等等

我把一些常用的CSS动画效果代码上传到github了，有需要的同学可以点击下载，CSS常用动画；

另外还有一些CSS动画库，比如：animate.css、magic.css、Hover.css、

13.CSS浏览器兼容性写法

1. 浏览器CSS样式初始化

由于每个浏览器的css默认样式不尽相同，所以最简单有效的方式就是对其进行初始化，相信很多朋友都写过这样的代码，在所有CSS开始前，先把marin和padding都设为0，以防不同浏览器的显示效果不一样。

```
*{
 margin: 0;
 padding: 0;
}
```

关于浏览器CSS样式初始化，经验不丰富的话，可能也不知道该初始化什么，这里给大家推荐一个库，Normalize.css，github star数量接近3.4万，选取展示其中几个样式设置，如下：

```
html {
 line-height: 1.15; /* Correct the line height in all browsers */
 -webkit-text-size-adjust: 100%; /* Prevent adjustments of font size after orientation
}

body {
 margin: 0;
}

a {
 background-color: transparent; /* Remove the gray background on active links in IE 10
}

img {
 border-style: none; /* Remove the border on images inside links in IE 10. */
}
```

通过CSS样式初始化，相信能解决不少常规的兼容性问题，接下来再看看浏览器的私有属性。

2. 浏览器私有属性

为什么会出现私有属性呢？这是因为制定HTML和CSS标准的组织W3C动作是很慢的。

通常，有W3C组织成员提出一个新属性，比如说圆角border-radius，大家都觉得好，但W3C制定标准，要走很复杂的程序，审查等。而浏览器商市场推广时间紧，如果一个属性已经够成熟了，就会在浏览器中加入支持。

但是为避免日后W3C公布标准时有所变更，会加入一个私有前缀，比如-webkit-border-radius，通过这种方式来提前支持新属性。等到日后W3C公布了标准，border-radius的标准写法确立之后，再让新版的浏览器支持border-radius这种写法。常用的前缀有：

- -moz代表firefox浏览器私有属性
- -ms代表IE浏览器私有属性
- -webkit代表chrome、safari私有属性
- -o代表opera私有属性

对于私有属性的顺序要注意，把标准写法放到最后，兼容性写法放到前面

```
-webkit-transform:rotate(-3deg); /*为Chrome/Safari*/
-moz-transform:rotate(-3deg); /*为Firefox*/
-ms-transform:rotate(-3deg); /*为IE*/
-o-transform:rotate(-3deg); /*为Opera*/
transform:rotate(-3deg);
```

每个CSS属性写这么一堆兼容性代码，无疑是对生命最大的浪费，后面我们会讲一下通过自动化插件来处理这块。

3. CSS hack

有时我们需要针对不同的浏览器或不同版本写特定的CSS样式，这种针对不同的浏览器/不同版本写相应的CSS code的过程，叫做CSS hack!

CSS hack的写法大致归纳为3种：条件hack、属性级hack、选择符级hack。

- 各游览器常用兼容标记一览表:

标记	IE6	IE7	IE8	FF	Opera	Sarari
[*+><]	√	√	X	X	X	X
_	√	X	X	X	X	X
\9	√	√	√	X	X	X
\0	X	X	√	X	√	X
@media screen and (-webkit-min-device-pixel-ratio:0) {.bb {}}	X	X	X	X	X	√
.bb , x:-moz-any-link, x:default	X	√	X	√(ff3.5及以下)	X	X
@-moz-document url-prefix()	X	X	X	√	X	X

(min-width: 0px){.bb {}}						
* +html .bb {}	X	√	X	X	X	X
游览器内核	Trident	Trident	Trident	Gecko	Presto	WebKit

（以上 .bb 可更换为其它样式名）

4.自动化插件

Autoprefixer是一款自动管理浏览器前缀的插件，它可以解析CSS文件并且添加浏览器前缀到CSS内容里，使用Can I Use（caniuse网站）的数据来决定哪些前缀是需要的。

把Autoprefixer添加到资源构建工具（例如Grunt）后，可以完全忘记有关CSS前缀的东西，只需按照最新的W3C规范来正常书写CSS即可。如果项目需要支持旧版浏览器，可修改browsers参数设置 。

```
//我们编写的代码
div {
 transform: rotate(30deg);
}

// 自动补全的代码，具体补全哪些由要兼容的浏览器版本决定，可以自行设置
div {
 -ms-transform: rotate(30deg);
 -webkit-transform: rotate(30deg);
 -o-transform: rotate(30deg);
 -moz-transform: rotate(30deg);
 transform: rotate(30deg);
}
```

目前webpack、gulp、grunt都有相应的插件，如果还没有使用，那就赶紧应用到我们的项目中吧，别再让CSS兼容性浪费你的时间！

14.掌握一套完整的响应式布局方案

比较常用的布局方式有float,position,display,table,flex,grid等。

全屏布局相关方案的兼容性、性能和自适应一览表：

实际项目使用中一般是根据具体场景去选择相应的布局方式。

附两张CSS知识图谱(建议收藏)：

注：部分内容来源于网络，仅供参考与学习，有遗漏或者错误的地方欢迎指出或者讨论！

> 原作者姓名： 深圳湾码农
> 原出处：掘金
> 原文链接：你需掌握的CSS知识都在这了（长文建议收藏，文末有福利） - 掘金

发布于 09-06

文章被以下专栏收录

推荐阅读

疯狂Html+CSS+JS 中JS总结

刘亦云

技术周刊 2020-03-02：CSS 4要来了吗

humph... 发表于阿里妈妈前...

CSS技巧(01

大漠

1 条评论

切换为时间排序

写下你的评论...

小黑 09-16

两张CSS知识图谱看不清

赞