0. 在页面中添加样式的方式有 234, 1 不是在页面中添加样式
1.导入样式：在 .css 文件中使用@import url("...")来引入另一个 css 样式表
2.外部样式：在 html 页面中的 head 中使用 link 标签引入，如<link rel="stylesheet" type="text/css" href="a.css" />
3.内部样式(嵌入式)：在 HTML 页面中的 style 标签中使用样式 ，<style type="text/css">...</style>
4.内联样式：与 html 标签的内部使用 style 属性设置的样式，直接与当前 html 标签相关联，如

css 没有单行注释//

1. CSS3 新特性：
https://segmentfault.com/a/1190000010780991 敲代码后总结
gradient 线性渐变
transform 旋转 transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);//旋转,缩放,定位,
transition 当元素从一种样式变换为另一种样式时为元素添加效果的属性
倾斜
增加了更多的 CSS 选择器
多背景
rgba
在 CSS3 中唯一引入的伪元素是::selection.
媒体查询,
多栏布局
要运用 css3 动画，需要运用@keyframes 规则和 animation 属性
border-top-right-radius:2em;向某元素的右上角添加圆角边框：

一. box-shadow(阴影效果)
使用:
box-shadow: 20px 10px 0 #000;
-moz-box-shadow: 20px 10px 0 #000;
-webkit-box-shadow: 20px 10px 0 #000;
支持:
FF3.5, Safari 4, Chrome 3
二. border-colors(为边框设置多种颜色)
使用:
border: 10px solid #000;
-moz-border-bottom-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-top-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-left-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;
-moz-border-right-colors: #555 #666 #777 #888 #999 #aaa #bbb #ccc;
说明:
颜色值数量不固定，且 FF 的私有写法不支持缩写: -moz-border-colors: #333 #444 #555;
支持:
FF3+
三. boder-image(图片边框)
使用:
-moz-border-image: url(exam.png) 20 20 20 20 repeat;
-webkit-border-image: url(exam.png) 20 20 20 20 repeat;
说明:
(1). 20 20 20 20 ---> 边框的宽度，分别对应 top, right, bottom, left 边框，改变宽度可以实现不同的效果;
(2). 边框图片效果(目前仅实现了两种):

repeat --- 边框图片会平铺，类似于背景重复;
stretch --- 边框图片会以拉伸的方式来铺满整个边框;
(3). 必须将元素的边框厚度设置为非 0 非 auto 值.
支持:
FF 3.5, Safari 4, Chrome 3
四. text-shadow(文本阴影)
使用:
text-shadow: [<颜色><水平偏移><纵向偏移><模糊半径>] || [<水平偏移><纵向偏移><模糊半径><颜色>];
说明:
(1) <颜色>和<模糊半径>是可选的，当<颜色>未指定时，将使用文本颜色；当<模糊半径>未指定时，半径值为 0;
(2) shadow 可以是逗号分隔的列表，如:
text-shadow: 2px 2px 2px #ccc, 3px 3px 3px #ddd;
(3) 阴影效果会按照 shadow list 中指定的顺序应用到元素上;
(4) 这些阴影效果有可能相互重叠，但不会叠加文本本身;
(5) 阴影可能会跑到容器的边界之外，但不会影响容器的大小.
支持:
FF 3.5, Opera 10, Safari 4, Chrome 3
五. text-overflow(文本截断)
使用:
text-overflow: inherit | ellipsis | clip ;
-o-text-overflow: inherit | ellipsis | clip;
说明:
(1) 还有一个属性 ellipsis-word，但各浏览器均不支持.
支持:
IE6+, Safari4, Chrome3, Opera10
六. word-wrap(自动换行)
使用:
word-wrap: normal | break-word;
支持:
IE6+, FF 3.5, Safari 4, Chrome 3
七. border-radius(圆角边框)
使用:
-moz-border-radius: 5px;
-webkit-border-radius: 5px;
支持:
FF 3+, Safari 4, Chrome 3

八. opacity(不透明度)
使用:
opacity: 0.5;
filter: alpha(opacity=50); /* for IE6, 7 */
-ms-filter(opacity=50); /* for IE8 */
支持:
all
九. box-sizing(控制盒模型的组成模式)
使用:
box-sizing: content-box | border-box; // for opera
-moz-box-sizing: content-box | border-box;
-webkit-box-sizing: content-box | border-box;

说明:
1. content-box:
使用此值时，盒模型的组成模式是，元素宽度 = content + padding + border;
2. border-box:
使用此值时，盒模型的组成模式是，元素宽度 = content(即使设置了 padding 和 border，元素的宽度也不会变).
支持:
FF3+, Opera 10, Safari 4, Chrome 3

transition 当元素从一种样式变换为另一种样式时为元素添加效果的属性
ps：
在 Blink 和 WebKit 的浏览器中，某个元素具有 3D 或透视变换（perspective transform）的 CSS 属性，会让浏览器创建单独的图层。
我们平常会使用 left 和 top 属性来修改元素的位置，但 left 和 top 会触发重布局，取而代之的更好方法是使用 translate，这个不会触发重布局。
~~移动端要想动画性能流畅，应该使用 3D 硬件加速，因此最好给页面中的元素尽量添加 translate3d 或者 translateZ(0) 来触发 3D 硬件加速。~~滥用硬件加速会导致严重性能问题，因为它增加了内存使用，并且它会导致移动端电池寿命减少。
解决浏览器渲染的性能问题时，首要目标就是要避免层的重绘和重排。

A：box-shadow: [<颜色><水平偏移><纵向偏移><模糊半径>] || [<水平偏移><纵向偏移><模糊半径><颜色>];
说明:
(1) <颜色>和<模糊半径>是可选的，当<颜色>未指定时，将使用文本颜色；当<模糊半径>未指定时，半径值为 0;
(2) shadow 可以是逗号分隔的列表，如:
box-shadow: 2px 2px 2px #ccc, 3px 3px 3px #ddd;
(3) 阴影效果会按照 shadow list 中指定的顺序应用到元素上;
(4) 这些阴影效果有可能相互重叠，但不会叠加文本本身;
(5) 阴影可能会跑到容器的边界之外，但不会影响容器的大小.

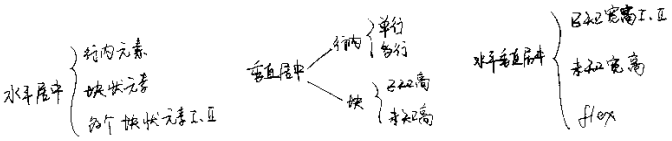
B: text-shadow 阴影的参数格式和 box-shadow 相同;

C:border-radius: r; Box 的四个角可以由边框半径来设置弯曲度，其定义方式和 border 属性类似;

D:rgba(0-255,0-255,0-255,0-1)
前三个数值是 RGB 颜色的值，最后一个数值指代的是元素的透明度 (0 表示透明, 1 表示不透明).

@keyframes 规则用于创建动画。在 @keyframes 中规定某项 CSS 样式，就能创建由当前样式逐渐改为新样式的动画效果。

2. css 水平垂直居中



垂直居中对齐的标签定义是:<vertical-align:center>
text-left 用于左对齐
text-center 水平居中对齐
text-uppercase 可以将字母全部大写

水平居中：行内元素解决方案
只需要把行内元素包裹在一个属性 display 为 block 的父层元素中，并且把父层元素添加如下属性即可：

```
.parent {
    text-align:center;
}
```

水平居中：块状元素解决方案

```
.item {
    /* 这里可以设置顶端外边距 */
    margin: 10px auto;
}
```

水平居中：多个块状元素解决方案
将元素的 display 属性设置为 inline-block，并且把父元素的 text-align 属性设置为 center 即可:

```
.parent {
    text-align:center;
}
```

水平居中：多个块状元素解决方案 (使用 flexbox 布局实现)
使用 flexbox 布局，只需要把待处理的块状元素的父元素添加属性 display:flex 及 justify-content:center 即可:

```
.parent {
    display:flex;
    justify-content:center;
}
```

垂直居中：单行的行内元素解决方案

```
.parent {
    background: #222;
    height: 200px;
}
/* 以下代码中，将 a 元素的 height 和 line-height 设置的和父元素一样高度即可实现垂直居中 */
a {
    height: 200px;
    line-height:200px;
    color: #FFF;
}
```

垂直居中：多行的行内元素解决方案
组合使用 display:table-cell 和 vertical-align:middle 属性来定义需要居中的元素的父容器元素生成效果，如下：

```
.parent {
    background: #222;
    width: 300px;
    height: 300px;
    /* 以下属性垂直居中 */
    display: table-cell;
    vertical-align:middle;
}
```

垂直居中：已知高度的块状元素解决方案

```
.item{
    top: 50%;
    margin-top: -50px;  /* margin-top 值为自身高度的一半 */
    position: absolute;
    padding:0;
}
```

垂直居中：未知高度的块状元素解决方案

```
.item{
    top: 50%;
    position: absolute;
    transform: translateY(-50%);  /* 使用 css3 的 transform 来实现 */
}
```

水平垂直居中：已知高度和宽度的元素解决方案 1

这是一种不常见的居中方法，可自适应，比方案 2 更智能，如下：

```
.item{
    position: absolute;
    margin:auto;
    left:0;
    top:0;
    right:0;
    bottom:0;
}
```

水平垂直居中：已知高度和宽度的元素解决方案 2

```
.item{
    position: absolute;
    top: 50%;
    left: 50%;
    margin-top: -75px;  /* 设置 margin-left / margin-top 为自身高度的一半 */
    margin-left: -75px;
}
```

水平垂直居中：未知高度和宽度元素解决方案

```
.item{
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);  /* 使用 css3 的 transform 来实现 */
}
```

水平垂直居中：使用 flex 布局实现

```
.parent{
    display: flex;
    justify-content:center;
    align-items: center;
    /* 注意这里需要设置高度来查看垂直居中效果 */
    background: #AAA;
```

```
    height: 300px;
}
```

对于不定宽的浮动元素我们也有一个常用的技巧解决它的水平居中问题。如下：

HTML 代码：

```
<div class="box">
    <p>我是浮动的</p>
    <p>我也是居中的</p>
</div>
```

CSS 代码：

```
.box{
    float:left;
    position:relative;
    left:50%;
}
p{
    float:left;
    position:relative;
    right:50%;
}
```

这样就解决了浮动元素水平居中了；

父元素和子元素同时左浮动，然后父元素相对左移动 50%，再然后子元素相对右移动 50%，或者子元素相对左移动-50%也就可以了。

3. 什么是无样式内容闪烁?如何避免?

FOUC（无样式内容闪烁）？你如何来避免 FOUC？

FOUC - Flash Of Unstyled Content 文档样式闪烁 <style type="text/css" media="all">@import "../fouc.css";</style> 而引用 CSS 文件的@import 就是造成这个问题的罪魁祸首。IE 会先加载整个 HTML 文档的 DOM，然后再去导入外部的 CSS 文件，因此，在页面 DOM 加载完成到 CSS 导入完成中间会有一段时间页面上的内容是没有样式的，这段时间的长短跟网速，电脑速度都有关系。 解决方法简单的出奇，只要在<head>之间加入一个<link>或者<script>元素就可以了。

what?

如果使用 import 方法对 CSS 进行导入,会导致某些页面在 Windows 下的 Internet Explorer 出现一些奇怪的现象:以无样式显示页面内容的瞬间闪烁,这种现象称之为文档样式短暂失效(Flash of Unstyled Content),简称为 FOUC。

why?

使用 import 导入样式表

将样式表放在页面底部

有几个样式表,放在页面不同位置

原因即:当样式表晚于结构性 html 加载，当加载到此样式表时，页面将停止之前的渲染。此样式表被下载和解析后，将重新渲染页面，也就出现了短暂的花屏现象。(js 背记资料中页面加载和渲染中也讲到)

how?

使用 LINK 标签将样式表放在文档 HEAD 中。

5.浮动、清除浮动

浮动可以理解为让某个 div 元素脱离标准流，漂浮在标准流之上，和标准流不是一个层次。假如某个 div 元素 A 是浮动的，如果 A 元素上一个元素也是浮动的，那么 A 元素会跟随在上一个元素的后边(如果一行放不下这两个元素，那么 A 元素会被挤到下一行)；如果 A 元素上一个元素是标准流中的元素，那么 A 的相对垂直位置不会改变，

也就是说 A 的顶部总是和上一个元素的底部对齐。

浮动引起的问题：

1）父元素的高度无法被撑开，影响与父元素同级的元素

2）与浮动元素同级的非浮动元素（内联元素）会跟随其后

3）若非第一个元素浮动，则该元素之前的元素也需要浮动，否则会影响页面显示的结构

清除浮动是为了消除浮动元素高度塌陷的问题，当一个内层元素是浮动的时候，如果没有关闭浮动时，其父元素也就不会再包含这个浮动的内层元素，因为此时浮动元素已经脱离了文档流。也就是为什么外层不能被撑开了！

现象：

分析 HTML 代码结构：

```
<div class="outer">
    <div class="div1">1</div>
    <div class="div2">2</div>
    <div class="div3">3</div>
</div>
```

分析 CSS 代码样式：

```
.outer{border: 1px solid #ccc;background: #fc9;color: #fff; margin: 50px auto;padding: 50px;}
.div1{width: 80px;height: 80px;background: red;float: left;}
.div2{width: 80px;height: 80px;background: blue;float: left;}
.div3{width: 80px;height: 80px;background: sienna;float: left;}
```

这里我没有给最外层的 DIV.outer 设置高度，但是我们知道如果它里面的元素不浮动的话，那么这个外层的高是会自动被撑开的。但是当内层元素浮动后，就出现了一下影响：

（1）：背景不能显示 （2）：边框不能撑开 （3）：margin 设置值不能正确显示

清除 css 浮动：

方法一：添加新的元素 、应用 clear：both；

HTML：

```
<div class="outer">
    <div class="div1">1</div>
    <div class="div2">2</div>
    <div class="div3">3</div>
    <div class="clear"></div>
</div>
```

CSS：

```
.clear{clear:both; height: 0; line-height: 0; font-size: 0}
```

方法二：父级 div 定义 overflow: auto（注意：是父级 div 也就是这里的 div.outer）

HTML：

```
<div class="outer over-flow"> //这里添加了一个 class
    <div class="div1">1</div>
    <div class="div2">2</div>
    <div class="div3">3</div>
    <!--<div class="clear"></div>-->
</div>
```

CSS：

```
.over-flow{
    overflow: auto; zoom: 1; //zoom: 1; 是在处理兼容性问题
}
```

原理：使用 overflow 属性来清除浮动有一点需要注意，overflow 属性共有三个属性值：hidden,auto,visible。我们可

以使用 hiddent 和 auto 值来清除浮动，但切记不能使用 visible 值，如果使用这个值将无法达到清除浮动效果，其他两个值都可以，其区据说在于一个对 seo 比较友好，另个 hidden 对 seo 不是太友好，其他区别我就说不上了，也不浪费时间。

方法三： 据说是最高大上的方法 :after 方法：（注意：作用于浮动元素的父亲）
先说原理：这种方法清除浮动是现在网上最拉风的一种清除浮动，他就是利用:after 和:before 来在元素内部插入两个元素块，从面达到清除浮动的效果。其实现原理类似于 clear:both 方法，只是区别在于:clear 在 html 插入一个 div.clear 标签，而 outer 利用其伪类 clear:after 在元素内部增加一个类似于 div.clear 的效果。下面来看看其具体的使用方法：

```
 .outer {zoom:1;}       /*==for IE6/7 Maxthon2==*/
 .outer :after {clear:both;content:'.';display:block;width: 0;height: 0;visibility:hidden;}/*==for FF/chrome/opera/IE8==*/
```

其中 clear:both;指清除所有浮动；content: '.';display:block;对于 FF/chrome/opera/IE8 不能缺少，其中 content（）可以取值也可以为空。visibility:hidden;的作用是允许浏览器渲染它，但是不显示出来，这样才能实现清楚浮动。
Tips：
下一标签直接清浮动兄弟标签浮动时，在下一标签的属性中直接写入清除 clear:both;
这样就可以清除以上标签的浮动而不用加入空标签来清除浮动。

```
<div style="width:400px;height:200px;">
  <span style="float:left;width:auto;height:100%;">
          <i style="position:absolute;float:left;width:100px;height:50px;">hello</i>
  </span>
</div>
```

问题：span 标签的 width 和 height 分别为多少？
首先 span 不是块级元素，是不支持宽高的，但是 style 中有了个 float :left ;就使得 span 变成了块级元素支持宽高，height:100%;即为，200，宽度由内容撑开。
但是内容中的 i 是绝对定位，脱离了文档流，所以不占父级空间，所以 span 的 width=0

6. 解释 CSS Sprites，以及你要如何使用?
答: CSS Sprites 其实就是把网页中一些小图标图片整合到一张图片文件中，再利用 CSS 的“background-position”进行图片定位只显示某一部分图标。其实我在做 100du 首页的时候就用过这个概念。为的是减少对图片的 http 请求次数以提高性能。
雪碧图的生成和使用方法有：gulp 和 webpack 的生成和实现待编码后总结。
雪碧图实现 1： CSS Gaga
雪碧图实现 2：photoShop
雪碧图实现 3：gulp
雪碧图实现 4：webpack
CSS Sprites 在国内很多人叫 css 精灵，是一种网页图片应用处理方式。它允许你将一个页面涉及到的所有零星图片都包含到一张大图中去，这样一来，当访问该页面时，载入的图片就不会像以前那样一幅一幅地慢慢显示出来了。
利用 CSS 的“background-image”，“background- repeat”，“background-position”的组合进行背景定位，background-position 可以用数字精确的定位出背景图片的位置。
利用 CSS Sprites 能很好地减少网页的 http 请求，从而大大的提高页面的性能，这也是 CSS Sprites 最大的优点，也是其被广泛传播和应用的主要原因；
CSS Sprites 能减少图片的字节，曾经比较过多次 3 张图片合并成 1 张图片的字节总是小于这 3 张图片的字节总和。所以 C 错误
解决了网页设计师在图片命名上的困扰，只需对一张集合的图片上命名就可以了，不需要对每一个小元素进行命名，从而提高了网页的制作效率。
更换风格方便，只需要在一张或少张图片上修改图片的颜色或样式，整个网页的风格就可以改变。维护起来更加方

便。

7.功能限制的浏览器- Media Query- CSS hack-渐进增强-优雅降级

为低版本 IE,、手机浏览器、不同的设备指定特定的样式

<meta name="viewport" content="width=device-width, initial-scale=1">表示支持响应式设计

7.1 思路：

渐进增强: 针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

优雅降级: 一开始就构建完整的功能，然后再针对低版本浏览器进行兼容

7.2 技术:

7.2.1Media Query：

CSS3 的特性 https://segmentfault.com/a/1190000008701062

如何优化网页的打印样式?

针对打印机的样式: @media print{...}

参考:如何优化网页的打印样式？ 移动端 H5 知识普及 - CSS3 媒体查询

你用过媒体查询，或针对移动端的布局/CSS 吗？

通过媒体查询可以为不同大小和尺寸的媒体定义不同的 css，适合相应的设备显示；即响应式布局

@media screen and (min-width: 400px) and (max-width: 700px) { … }

@media handheld and (min-width: 20em), screen and (min-width: 20em) { … }

参考自: CSS3 媒体查询 使用 CSS 媒体查询创建响应式网站 《响应式 Web 设计实践》学习笔记

下列 media queries 的定义方式中，哪一个是适配 iphone6s 的？（）

正确答案: A 你的答案: C (错误)

. @media(min-device-width:375px) and(max-device-width:667px) and(-webkit-min-device-pixel-ratio:2){}

. @media(min-device-width:414px) and(max-device-width:736px) and(-webkit-min-device-pixel-ratio:3){}

@media only screen and (min-device-width:320px) and (max-device-width:480px) and (-webkit-device-pixel-ratio:2){}

@media only screen and (min-device-width:320px) and (max-device-width:568px) and (-webkit-min-device-pixel-ratio:2){}

不要纠结选择 A 或者选择 B 了 主要是在-webkit-min-device-pixel-ratio:2 这个属性

想知道的去看看这边文章 http://www.gbtags.com/gb/share/5307.htm（解释是对的，但是这上面的设备像素比率有错）

iPhone6s 是 2 而 plus 是 3 设备像素比率看这个 https://bjango.com/articles/min-device-pixel-ratio/

7.2.2CSS hack：

CSS hack 的目的就是使你的 CSS 代码兼容不同的浏览器。当然，我们也可以反过来利用 CSS hack 为不同版本的浏览器定制编写不同的 CSS 效果。一般情况下，我们尽量避免使用 CSS hack，但是有些情况为了顾及用户体验实现向下兼容，不得已才使用 hack。

CSS Hack 表现形式：类内属性前缀法、选择器前缀法、条件注释法(if IE)、选择器结合 JavaScript 的 Hack

http://blog.csdn.net/kongjiea/article/details/42641177

ps：bootstrap 的不同屏幕适配

bootstrap3.x 使用了四种栅格选项来形成栅格系统， 其实区别只有一条就是适合不同尺寸的屏幕设备 。姑且以前缀命名这四种栅格选项，他们分别是 col-xs、col-sm、col-md、col-lg,

mobile – xs (<768px)

tablet – sm ([768~992)px)

desktop – md [992~1200)px)

large desktop – lg (>=1200px)

8. 高效 CSS

1）reset。参照下题“描述下 “reset” CSS 文件的作用和使用它的好处”的答案。
2）规范命名。尤其对于没有语义化的 html 标签，例如 div，所赋予的 class 值要特别注意。
3）抽取可重用的部件，注意层叠样式表的“优先级”。

9. CSS 预处理

书写 CSS 时难以解决的问题：

语法不够强大，比如无法嵌套书写导致模块化开发中需要书写很多重复的选择器；

没有变量和合理的样式复用机制，使得逻辑上相关的属性值必须以字面量的形式重复输出，导致难以维护。

所以这就决定了 CSS 预处理器的主要目标：提供 CSS 缺失的样式层复用机制、减少冗余代码，提高样式代码的可维护性。这不是锦上添花，而恰恰是雪中送炭。

目前最主流的 CSS 预处理器是 LESS、SASS 和 Stylus

http://efe.baidu.com/blog/revisiting-css-preprocessors/

http://www.cnblogs.com/haoyijing/p/5793788.html

优点:

结构清晰，便于扩展

可以方便屏幕浏览器私有语法差异

可以轻松实现多重继承

完全兼容 css 代码

10. 非标准的字体如何实现

1)少量文字的话使用图片替代

2)web : fonts 在线字库

fonts 在线字库，如 Google Webfonts、Typekit 等等；Typekit：网站会给出对应此 kit 的一段 js 代码，将其嵌入到页面之间即可；http://www.chinaz.com/free/2012/0815/269267.shtml

3)@font-face

CSS3

11. CSS 选择器

种类

1.id 选择器（ # myid）

2.类选择器（.myclassname）

3.标签选择器（div, h1, p）

4.相邻选择器（h1 + p）

5.子选择器（ul > li）

6.后代选择器（li a）

7.通配符选择器（ * ）

8.属性选择器（a[rel = "external"]）

9.伪类选择器（a: hover, li:nth-child）

10. 并选择器 span, .item

http://www.w3school.com.cn/cssref/css_selectors.asp

p:first-child 选择属于父元素(不确定、不止一个)的第一个<p>子元素。

p:first-child i 选择同上的 p 元素中的所有 i 元素

优先级(级别数组从高到低)为:

!important >内联样式 > id 选择器 > 类选择器 = 属性选择器 = 伪类选择器 > 标签选择器 = 伪元素选择器
还要根据这四个级别出现的次数累加得到最后的优先级，优先级数字越大越优先

基于以下 HTML 结构和 CSS 样式，文本 Dijkstra 的颜色应该是？

```
<ul class="authors" id="favorite">
  <li><span>Martin Fowler</span></li>
  <li id="related"><span class="highlight">Dijkstra</span></li>
</ul>
ul#related span {
  color: red;
}
#favorite .highlight {
  color: orange;
}
.highlight {
 color: black;
}
```

标签的权重是 1，类的权重是 10 ，id 的权重是 100
ul#related span 的权重为 1+100+1=102
#favorite .highlight 的权重为 100+10=110
highlight 的权重为 10
所以是 orange。

11.1 浏览器判断元素是否匹配某个选择器的过程

从后往前判断。

浏览器先产生一个元素集合，这个集合往往由最后一个部分的索引产生（如果没有索引就是所有元素的集合）。然后向上匹配，如果不符合上一个部分，就把元素从集合中删除，直到真个选择器都匹配完，还在集合中的元素就匹配这个选择器了。

举个例子，有选择器：

body.ready #wrapper > .lol233

先把所有 class 中有 lol233 的元素拿出来组成一个集合，然后上一层，对每一个集合中的元素，如果元素的 parent id 不为 #wrapper 则把元素从集合中删去。 再向上，从这个元素的父元素开始向上找，没有找到一个 tagName 为 body 且 class 中有 ready 的元素，就把原来的元素从集合中删去。

至此这个选择器匹配结束，所有还在集合中的元素满足。

大体就是这样，不过浏览器还会有一些奇怪的优化。

为什么从后往前匹配因为效率和文档流的解析方向。效率不必说，找元素的父亲和之前的兄弟比遍历所哟儿子快而且方便。关于文档流的解析方向，是因为现在的 CSS，一个元素只要确定了这个元素在文档流之前出现过的所有元素，就能确定他的匹配情况。应用在即使 html 没有载入完成，浏览器也能根据已经载入的这一部分信息完全确定出现过的元素的属性。

为什么是用集合主要也还是效率。基于 CSS Rule 数量远远小于元素数量的假设和索引的运用，遍历每一条 CSS Rule 通过集合筛选，比遍历每一个元素再遍历每一条 Rule 匹配要快得多。

如果需要匹配包含文本的元素，用下面哪种方法来实现？
正确答案: B 你的答案: B (正确)
text()
contains()
input()
attr(name)
解答：
text()是 jQuery 中的方法，可以设置或返回被选元素的文本内容
：contains 选择器，选取包含指定字符串的元素，字符串也可以是文本
:input()选择器，选取表单元素
attr(name,value)属性操作，设置或返回被选元素的属性和属性值

给定下面的 HTML 代码：

```
<div id=”wrapper”>
    <div class=”wText”>…</div>…<!—more wText items here -->
    <div class=”wImg”>…</div>…<!—more wImg items here -->
    <div class=”wVideo”>…</div>…<!—more wVideo items here -->
</div>
```

怎么能够取得 ”wrapper” 中全部项的集合？

$(‘#wrapper’).children(); //（只沿着 DOM 树向下遍历单一层级）查询直接的子元素。而不管子元素的子元素。

$(‘#wrapper’).html(); //返回或设置被选元素的内容 (inner HTML)；返回的是 dom 结构。而不是集合

$(‘#wrapper’).contents();获得匹配元素集合中每个元素的子节点，包括文本和注释节点。

$(‘#wrapper’).find(“all”); //获得当前元素集合中每个元素的后代，通过选择器、jQuery 对象或元素来筛选。并没有 all 这个元素

12. 盒模型、以及如何在 CSS 中告诉浏览器使用不同的盒模型来渲染你的布局。

CSS 样式，边距： 10px 20px 40px 30px ；哪一个是底边距？上左下右

盒模型 :文档中的每个元素被描绘为矩形盒子，渲染引擎的目的就是判定大小，属性——比如它的颜色、背景、边框方面——及这些盒子的位置。

在 CSS 中，这些矩形盒子用标准盒模型来描述。这个模型描述了一个元素所占用的空间。每一个盒子有四条边界：外边距边界 margin edge，边框边界 border edge，内边距边界 padding edge 和内容边界 content edge。

内容区域是真正包含元素内容的区域，位于内容边界的内部，它的大小为内容宽度或 content-box 宽及内容高度或 content-box 高。如果 box-sizing 为默认值，width、min-width、max-width、height、min-height 和 max-height 控制内容大小。

内边距区域 padding area 用内容可能的边框之间的空白区域扩展内容区域。通常有背景——颜色或图片（不透明图片盖住背景颜色）。

边框区域扩展了内边距区域。它位于边框边界内部，大小为 border-box 宽和 border-box 高。

外边距区域 margin area 用空白区域扩展边框区域，以分开相邻的元素。它的大小为 margin-box 的高宽。

在外边距合并的情况下，由于盒之间共享外边距，外边距不容易弄清楚。

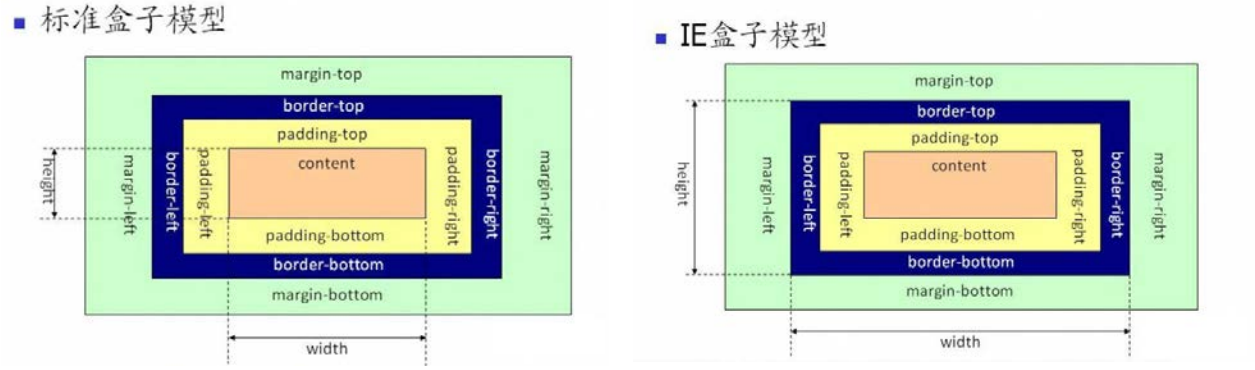
对于非替换的行内元素来说，尽管内容周围存在内边距与边框，但其占用空间（行高）由 line-height 属性决定。

盒子模型分为两类：W3C 标准盒子模型和 IE 盒子模型 （微软确实不喜欢服从他家的标准）

这两者的关键差别就在于：

W3C 盒子模型——属性高（height）和属性宽（width）这两个值不包含 填充（padding）和边框（border）

IE 盒子模型——属性高（height）和属性宽（width）这两个值包含 填充（padding）和边框（border）



标准 w3c 盒子模型的范围包括 margin、border、padding、content，并且 content 部分不包含其他部分

ie 盒子模型的范围也包括 margin、border、padding、content，但它的 content 部分包含了 border 和 pading。

例：一个盒子的 margin 为 20px，border 为 1px，padding 为 10px，content 的宽为 200px、高为 50px;

假如用标准 w3c 盒子模型解释，
那么这个盒子需要占据的位置为：宽 20*2+1*2+10*2+200=262px、高 20*2+1*2*10*2+50=112px，
盒子的实际大小(border、padding、content)为：宽 1*2+10*2+200=222px、高 1*2+10*2+50=72px；
假如用 ie 盒子模型，
那么这个盒子需要占据的位置为：宽 20*2+200=240px、高 20*2+50=70px，
盒子的实际大小(border、padding、content)为：宽 200px、高 50px。

解决兼容型为题最简洁和值得推荐的方式是：将页面设为“标准模式”(在网页的顶部加上 doctype 声明)，添加对应的 dtd 标识；或者使用 hack 或者在外面套上一层 wrapper

* { box-sizing: border-box; }统一 IE 和非 IE 浏览器之间的差异。
对于 IE 浏览器，当我们设置 box-sizing: content-box; 时，浏览器对盒模型的解释遵从我们之前认识到的 W3C 标准，当它定义 width 和 height 时，它的宽度不包括 border 和 padding；
对于非 IE 浏览器，当我们设置 box-sizing: border-box; 时，浏览器对盒模型的解释与 IE6 之前的版本相同，当它定义 width 和 height 时，border 和 padding 则是被包含在宽高之内的。内容的宽和高可以通过定义的“width”和“height”减去相应方向的“padding”和“border”的宽度得到。内容的宽和高必须保证不能为负，必要时将自动增大该元素 border box 的尺寸以使其内容的宽或高最小为 0。

box-sizing。盒模型默认的值是 content-box，新增的值是 padding-box 和 border-box，几种盒模型计算元素宽高的区别如下：
content-box（默认）
布局所占宽度 Width：
Width = width + padding-left + padding-right + border-left + border-right
布局所占高度 Height:
Height = height + padding-top + padding-bottom + border-top + border-bottom
padding-box
布局所占宽度 Width：
Width = width(包含 padding-left + padding-right) + border-top + border-bottom
布局所占高度 Height:
Height = height(包含 padding-top + padding-bottom) + border-top + border-bottom
border-box
布局所占宽度 Width：
Width = width(包含 padding-left + padding-right + border-left + border-right)
布局所占高度 Height:
Height = height(包含 padding-top + padding-bottom + border-top + border-bottom)

13. 伪类
清除浮动的最佳办法
http://www.cnblogs.com/ghost-xyx/p/3763669.html

CSS3 新增伪类举例：
p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。 p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。 p:only-of-type 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。 p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。p:nth-child(2) 选择属于其父元素的第二个子元素的每个 <p> 元素。 :enabled :disabled 控制表单控件的禁用状态。 :checked 单选框或复选框被选中。

在 class 和 id 中中是否都可以定义:hover 伪类？
伪类选择符 E:hover 的定义和用法:

设置元素在其鼠标悬停时的样式。
E 元素可以通过其他选择器进行选择，比如使用类选择符、id 选择符、类型选择符等等。
特别说明:IE6 并非不支持此选择符，但能够支持 a 元素的:hover ，也就是只支持通过类型选择符选择的 a 元素的:hover 。
总之，id 和 class 都可以

14. reset.css

reset.css 能够重置浏览器的默认属性。不同的浏览器具有不同的样式，重置能够使其统一。比如说 ie 浏览器和 FF 浏览器下 button 显示不同，通过 reset 能够统一样式，显示相同的效果。但是很多 reset 是没必要的，多写了会增加浏览器在渲染页面的负担。

比如说，

1）我们不应该对行内元素设置无效的属性，对 span 设置 width 和 height，margin 都不会生效的。

2）对于 absolute 和 fixed 定位的固定尺寸（设置了 width 和 height 属性），如果设置了 top 和 left 属性，那么 bottom 和 right，margin 和 float 就没有作用。

3）后面设置的属性将会覆盖前面重复设置的属性。

期待能够指出它的负面影响，或者提到它的一个更好的替换者"normalize"　normalize.css 是一个可以定制的 css 文件，它让不同的浏览器在渲染元素时形式更统一。

15. block, inline 和 inline-block 的区别

15.1 起新行
block 元素会独占一行，多个 block 元素会各自新起一行. 默认情况下, block 元素宽度自动填满其父元素宽度
inline 元素不会独占一行，多个相邻的行内元素会排列在同一行里，直到一行排列不下，才会新换一行，其宽度随元素的内容而变化
15.2 设置宽高
block 元素可以设置 width, height 属性. 块级元素即使设置了宽度，仍然独占一行
inline 元素设置 width, height 无效，但是可以设置 line-height
15.3 内外边距
block 元素可以设置 margin 和 padding 属性
inline 元素的 margin 和 padding 属性,水平方向的 padding-left, padding-right, margin-left, margin-right 都会产生边距效果. 但是竖直方向的 margin/padding-top/bottom 不会产生边距效果
15.4 包含
block 可以包含 inline 和 block 元素,而 inline 元只能包含 inline 元素
而 display: inline-block, 则是将对象呈现为 inline 对象，但是对象的内容作为 block 对象呈现. 之后的内联对象会被排列到一行内. 比如我们可以给一个 link(a 元素)inline-block 的属性，使其既有 block 的高宽特性又有 inline 的同行特性

常见的空元素：br hr img input link meta　base area command embed keygen param source track wbr....

常用的块状元素有：

、<p>、<h1>...<h6>、<ol>、<ul>、<dl>、<table>、<address>、<blockquote> 、<form>

常用的内联元素有：

<a>、、
、<i>、、、<label>、<q>、<var>、<cite>、<code>

常用的内联块状元素有：

<img>、<input>

1.常见的块级元素(自动换行，可设置高宽)有：
div, h1-h6, p, pre, ul, li, ol, form, table, label, dl, dt, dd, hr
2.常见的行内元素（无法自动换行，无法设置宽高）有：

a,img,span,i,em,sub,sup,input, strong select
3.常见的行块级元素(拥有内在尺寸，可设置高宽，不会自动换行)有：行内块元素也是行内元素
选择 button,select,输入 input,textarea, img 等

form

form 表单中 input 元素的 readonly 与 disabled 属性
设置 readonly = true，页面上无法修改内容，但是可以通过 JavaScript 修改,内容会被提交
设置 disabled = true,页面上无法修改内容，也可以通过 JavaScript 修改，但不会被提交

当表单中包含文件上传控件时，需要将 enctype 设置为（）
正确答案: B　你的答案: D (错误)
application/x-www-form-urlencoded
multipart/form-data
text/explain
file-data
解析：
enctype 属性规定在发送到服务器之前应该如何对表单数据进行编码。属性值：
application/x-www-form-urlencoded 在发送前编码所有字符（默认）
multipart/form-data 不对字符编码。 在使用包含文件上传控件的表单时，必须使用该值。
text/plain 空格转换为 "+" 加号，但不对特殊字符编码。

Table

在使用 table 表现数据时，有时候表现出来的会比自己实际设置的宽度要宽，为此需要设置下面哪些属性值
单元格边距(表格填充)(cellpadding) -- 代表单元格外面的一个距离,用于隔开单元格与单元格空间单；
元格间距(表格间距)(cellspacing) -- 代表表格边框与单元格补白的距离,也是单元格补白之间的距离。

没有 cellpadding:

First	Row
Second	Row

带有 cellpadding:

First	Row
Second	Row

没有 cellspacing:

First	Row
Second	Row

带有 cellspacing:

First	Row
Second	Row

16. css 动画和 js 动画

CSS3 的动画
优点：
1.在性能上会稍微好一些，浏览器会对 CSS3 的动画做一些优化（比如专门新建一个图层用来跑动画）
2.代码相对简单
缺点：
1.在动画控制上不够灵活
2.兼容性不好
3.部分动画功能无法实现（如滚动动画，视差滚动等）

JavaScript 的动画
优点：
1.控制能力很强，可以单帧的控制、变换

2.兼容性好，写得好完全可以兼容 IE6，且功能强大。
缺点：
计算没有 css 快，另外经常需要依赖其他的库。

结论
所以，不复杂的动画完全可以用 css 实现，复杂一些的，或者需要交互的时候，用 js 会靠谱一些~

17. 有哪些隐藏内容的方法(同时还要保证屏幕阅读器可用)
display:none 或者 visibility:hidden，overflow:hidden。
visibility:hidden;所占据的空间位置仍然存在，仅为视觉上的完全透明；
display:none;不为被隐藏的对象保留其物理空间；
display：none 指的是元素完全不陈列出来，不占据空间，涉及到了 DOM 结构，故产生 reflow(回流)与 repaint(重绘)
visibility：hidden 指的是元素不可见但存在，保留空间，不影响结构，故只产生 repaint(重绘)

overflow:hidden(对块级有效，行内元素需要 display:inline-block;)

引发回流：
1.调整窗口大小（Resizing the window）
2.改变字体（Changing the font）
3.增加或者移除样式表（Adding or removing a stylesheet）
4.内容变化，比如用户在 input 框中输入文字（Content changes, such as a user typing text in an input box）
5.激活 CSS 伪类，比如 :hover (IE 中为兄弟结点伪类的激活)（Activation of CSS pseudo classes such as :hover (in IE the activation of the pseudo class of a sibling)）
6.操作 class 属性（Manipulating the class attribute）
7.脚本操作 DOM（A script manipulating the DOM）
8.计算 offsetWidth 和 offsetHeight 属性（Calculating offsetWidth and offsetHeight）
9.设置 style 属性的值 （Setting a property of the style attribute）
应对方案：
1.如果想设定元素的样式，通过改变元素的 class 名 (尽可能在 DOM 树的最末端)（Change classes on the element you wish to style (as low in the dom tree as possible)）
2.避免设置多项内联样式（Avoid setting multiple inline styles）
3.应用元素的动画，使用 position 属性的 fixed 值或 absolute 值（Apply animations to elements that are position fixed or absolute）
4.权衡平滑和速度（Trade smoothness for speed）
5.避免使用 table 布局（Avoid tables for layout）
6.避免使用 CSS 的 JavaScript 表达式 (仅 IE 浏览器)（Avoid JavaScript expressions in the CSS (IE only)）

1.display:none;的缺陷
搜索引擎可能认为被隐藏的文字属于垃圾信息而被忽略
屏幕阅读器（是为视觉上有障碍的人设计的读取屏幕内容的程序）会忽略被隐藏的文字。
2.visibility: hidden ;的缺陷
隐藏的内容会占据他所应该占据物理空间
3.overflow:hidden;一个比较合理的方法
例：.texthidden { display:block; overflow: hidden; width: 0; height: 0; }
将宽度和高度设定为 0，然后超过部分隐藏，就会弥补上述一、二方法中的缺陷，也达到了隐藏内容的目的
参数是 scroll 时候，必会出现滚动条。
参数是 auto 时候，子元素内容大于父元素时出现滚动条。

参数是 visible 时候，溢出的内容出现在父元素之外。
参数是 hidden 时候，溢出隐藏。

18. img 设置属性 title 和 alt 的区别?

Alt 是 img 的特有属性，或与<input type="image">配合使用，规定图像的替代文本. 如果无法显示图像，浏览器将显示替代文本. 用于图片无法加载显示、读屏器阅读图片，可提高图片可 访问性，搜索引擎会重点分析。最长可包含 1024 个字符，
Title 为元素提供附加的提示信息，用于鼠标滑到元素上的时候显示。其值可以比 alt 属性值设置的更长，但是有些浏览器会截断过长的文字.

19. BFC

BFC 就是“块级格式化上下文”的意思，创建了 BFC 的元素就是一个独立的盒子，不过只有 Block-level box 可以参与创建 BFC，它规定了内部的 Block-level Box 如何布局，并且与这个独立盒子里的布局不受外部影响，当然它也不会影响到外面的元素。

BFC 的生成:
根元素
float 的值不为 none
overflow 的值不为 visible
display 的值为 inline-block, table-cell, table-caption
position 的值为 absolute 或 fixed

BFC 的约束规则:
内部的 Box 会在垂直方向上一个接一个的放置
垂直方向上的距离由 margin 决定。(完整的说法是：属于同一个 BFC 的两个相邻 Box 的 margin 会发生重叠，与方向无关。)
每个元素的左外边距与包含块的左边界相接触（从左向右），即使浮动元素也是如此。(这说明 BFC 中子元素不会超出他的包含块，而 position 为 absolute 的元素可以超出他的包含块边界)
BFC 的区域不会与 float 的元素区域重叠
计算 BFC 的高度时，浮动子元素也参与计算
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元素，反之亦然

BFC 在布局中的应用:
不和浮动元素重叠:
如果一个浮动元素后面跟着一个非浮动元素，就会产生覆盖
防止 margin 重叠:
同一个 BFC 中的两个相邻 Box 才会发生重叠与方向无关，不过由于上文提到的第一条限制，我们甚少看到水平方向的 margin 重叠。这在 IE 中是个例外，IE 可以设置 write-mode

解决浮动相关问题:
父元素: overflow:hidden　　IE: zoom:1(hasLayout)
根本原因在于创建 BFC 的元素，子浮动元素也会参与其高度计算，即不会产生高度塌陷问题。实际上只要让父元素生成 BFC 即可，并不只有这两种方式。
多栏布局
比如左右两栏宽度固定，中间栏自适应 则中间栏设置 overflow:hidden 生成 BFC
IE 中类似概念: hasLayout
参考: 我对 BFC 的理解　CSS BFC 和 IE Haslayout 介绍

BFC，块级格式化上下文，一个创建了新的 BFC 的盒子是独立布局的，盒子里面的子元素的样式不会影响到外面的

元素。在同一个 BFC 中的两个毗邻的块级盒在垂直方向（和布局方向有关系）的 margin 会发生折叠。

20.圣杯双飞翼

http://www.cnblogs.com/imwtr/p/4441741.html#top

21.position 的值， relative 和 absolute 分别是相对于谁进行定位的？

absolute :生成绝对定位的元素， 相对于最近一级的 定位不是 static 的父元素来进行定位。
fixed （老 IE 不支持）生成绝对定位的元素，通常相对于浏览器窗口或 frame 进行定位。
relative 生成相对定位的元素，相对于其在普通流中的位置进行定位。
static 默认值。没有定位，元素出现在正常的流中
sticky 生成粘性定位的元素，容器的位置根据正常文档流计算得出，并不脱离文档流
inherit 规定应该从父元素继承 position 属性的值

position:absolute 和 float 属性的异同
共同点：对内联元素设置 float 和 absolute 属性，可以让元素脱离文档流，并且可以设置其宽高。
不同点：float 仍会占据位置，absolute 会覆盖文档流中的其他元素。

在 css 的定位机制有三种，分别是 1：文档流，2：浮动（float），3 定位（position）
其中文档流的意义就是按照 HTML 里面的写法就是从上到下，从左到右的排版布局;
在 4 答案选项中的属性，float（浮动）和 position（定位）了
A：position: absolute;
生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位；都绝对定位了，肯定脱离了文档流。。
B:position: fixed;
生成绝对定位的元素，相对于浏览器窗口进行定位;相对于浏览器了，也和正常顺序排下来没什么关系。。
C:position: relative;
生成相对定位的元素，相对于其正常位置进行定位。生成相对定位，也就是说还在原本的上下左右之间，上下左右的元素都不变，so 这个没有能脱离文档流。就这个了
D:float: left;都浮动出去了，还上哪保持原位置去。

22.CSS 中 link 和@import 的区别是？

(1) link 属于 HTML 标签，而@import 是 CSS 提供的;
(2) 页面被加载的时，link 会同时被加载，而@import 被引用的 CSS 会等到引用它的 CSS 文件被加载完再加载；所以会出现一开始没有 css 样式，闪烁一下出现样式后的页面(网速慢的情况下)。
(3) import 只在 IE5 以上才能识别，而 link 是 HTML 标签，无兼容问题;
(4) link 方式的样式的权重 高于@import 的权重.
(5)当使用 javascript 控制 dom 去改变样式的时候，只能使用 link 标签，因为@import 不是 dom 可以控制的。
(6)link 除了能加载 css 外还能定义 RSS，定义 rel 连接属性，@import 只能加载 css

23.说说你对语义化的理解？

1，去掉或者丢失样式的时候能够让页面呈现出清晰的结构
2，有利于 SEO：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重；
3，方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；
4，便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循 W3C 标准的团队都遵循这个标准，可以减少差异化。

25.常见兼容性问题？

png24 位的图片在 iE6 浏览器上出现背景，解决方案是做成 PNG8.也可以引用一段脚本处理. 浏览器默认的 margin 和 padding 不同。解决方案是加一个全局的*{margin:0;padding:0;}来统一。 IE6 双边距 bug:块属性标签 float 后，又有横行的 margin 情况下，在 ie6 显示 margin 比设置的大。 浮动 ie 产生的双倍距离（IE6 双边距问题：在 IE6 下，如果对元素设置了浮动，同时又设置了 margin-left 或 margin-right，margin 值会加倍。） #box{ float:left; width:10px; margin:0 0 0 100px;} 这种情况之下 IE 会产生 20px 的距离，解决方案是在 float 的标签样式控制中加入 _display:inline; 将其转化为行内属性。(_这个符号只有 ie6 会识别) 渐进识别的方式，从总体中逐渐排除局部。 首先，巧妙的使用“\9”这一标记，将 IE 游览器从所有情况中分离出来。 接着，再次使用“+”将 IE8 和 IE7、IE6 分离开来，这样 IE8 已经独立识别。 css .bb{ background-color:#f1ee18;/*所有识别*/ .background-color:#00deff\9; /*IE6、7、8 识别*/ +background-color:#a200ff;/*IE6、7 识别*/ _background-color:#1e0bd1;/*IE6 识别*/ } 怪异模式问题：漏写 DTD 声明，Firefox 仍然会按照标准模式来解析网页，但在 IE 中会触发 怪异模式。为避免怪异模式给我们带来不必要的麻烦，最好养成书写 DTD 声明的好习惯。现在 可以使用[html5](http://www.w3.org/TR/html5/single-page.html)推荐的写法：`<doctype html>`
上下 margin 重合问题
ie 和 ff 都存在，相邻的两个 div 的 margin-left 和 margin-right 不会重合，但是 margin-top 和 margin-bottom 却会发生重合。 解决方法，养成良好的代码编写习惯，同时采用 margin-top 或者同时采用 margin-bottom。

26.上下 margin 重合问题

ie 和 ff 都存在，相邻的两个 div 的 margin-left 和 margin-right 不会重合，但是 margin-top 和 margin-bottom 却会发生重合。 解决方法，养成良好的代码编写习惯，同时采用 margin-top 或者同时采用 margin-bottom。

27. margin-top/margin-bottom 的百分数是相对最近父级块级元素的 width，padding-top/padding-bottom 的百分数也是相对最近父级块级元素的 width

宽度参照宽度这点毫无疑问，但高度参照的也是宽度这点，可能就和通常的思路相左，因为毕竟 height 应用百分比参照的，依然是容器的高度。
关于为什么标准要这么定义，有几种通常的解释，就一并(个人)分析一下。由于 padding 和 margin 类似，下文就只以 padding-top 为例。
第一种说法是，padding-top 如果以容器高度为参照，那么子元素应用 padding 值将会继续加高容器的高度，容器高度的变化又会反过来继续影响子元素的 padding-top，陷入一个无限循环。
对于不定高的容器来说情况确实如此，但对定高容器则并不成立，这点和 height 类似，这也是为什么 height 的容器也必须确定好高度。也就是说，如果 padding-top 要参照容器高度，就必须和 height 一样做两种处理。
第二种说法则更为可靠， 为了保持 padding(margin)四个值的一统一 。
padding(margin)的值无论引用何种计量，四个值都一直保持统一，难道单独给百分比开特例？结合第一条的多情况处理，如果标准定义 padding-top 参照容器高度的话，恐怕要列出至少 4 条以上的例外说明——而这对无论谁，都没有好处:)
事实上，垂直 padding 参照体是宽度这点也非常有用。虽然早期固化像素的设计中百分比值几乎不应用在 padding 或者 margin 上，但随着移动自 适应的布局的需要，百分比也逐渐稀疏平常。比如配合 background-sizing 保持背景的比例，配合 media query 调整对应的间距，不一而足。这些应用都基于一个事实： 无论垂直还是水平，百分比值始终参考宽度 。
而宽度，实际上，正是自适应和现代 web 设计的灵魂。

代码题：
https://github.com/hawx1993/Front-end-Interview-questions/blob/master/readme.html

28.IE 和其他浏览器

下面关于 IE、FF 下面 CSS 的解释区别描述正确的有？
正确答案: C D 你的答案: 空 (错误)
A. FireFox 的 div 的内嵌 div 可以把父级的高度撑大，而 IE6.0 不可以，要自己设置高度。
B. 当设置为三列布局时，FireFox0 的 float 宽度不能达到 100%，而 IE6.可以。当设置为两列布局时，两种浏览器都可以。

C. 火狐浏览器中，非 float 的 div 前面有同一父级的 float 的 div，此 div 若有背景图，要使用 clear：both，才能显示背景图，而 IE6.0 中不用使用 clear：both

D. 在[text-decoration:underline]的属性下，IE6.0 显示的下划线会比 FireFox 低一点。在 FireFox 中，部分笔画会在下划线的下面 1 个象素左右。

解析：

A:IE6.0 的 div 的内嵌 div 可以把父级的高度撑大，而 FireFox 不可以，要自己设置高度。

B: 当设置为三列布局时，IE6.0 的 float 宽度不能达到 100%，而 FireFox 可以。当设置为两列布局时，两种浏览器都可以。

29.样式继承

继承就是指子节点默认使用父节点的样式属性。

不可继承的属性太多了不要背，记住可以继承的属性有哪些就行了。可以继承的属性很少，只有**颜色，文字**，字体间距行高对齐方式，和**列表的样式**可以继承。

1.所有元素可继承：visibility 和 cursor。

2.内联元素可继承：letter-spacing、word-spacing、white-space、line-height、color、font、font-family、font-size、

3.font-style、font-variant、font-weight、text-decoration、text-transform、direction。

Ps： 注册，通过 style 调用时不能出现-

4.终端块状元素可继承：text-indent 和 text-align。

5.列表元素可继承：list-style、list-style-type、list-style-position、list-style-image。

30.

有一段 html 代码：

```
1  <div style=”color:red; text-color:blue;”><span  style=”color:green;text-color:black;”>Hello</span></div>,
```

那么”Hello”的字体颜色是（）

Green，不存在 text-color 属性

31.flex 布局

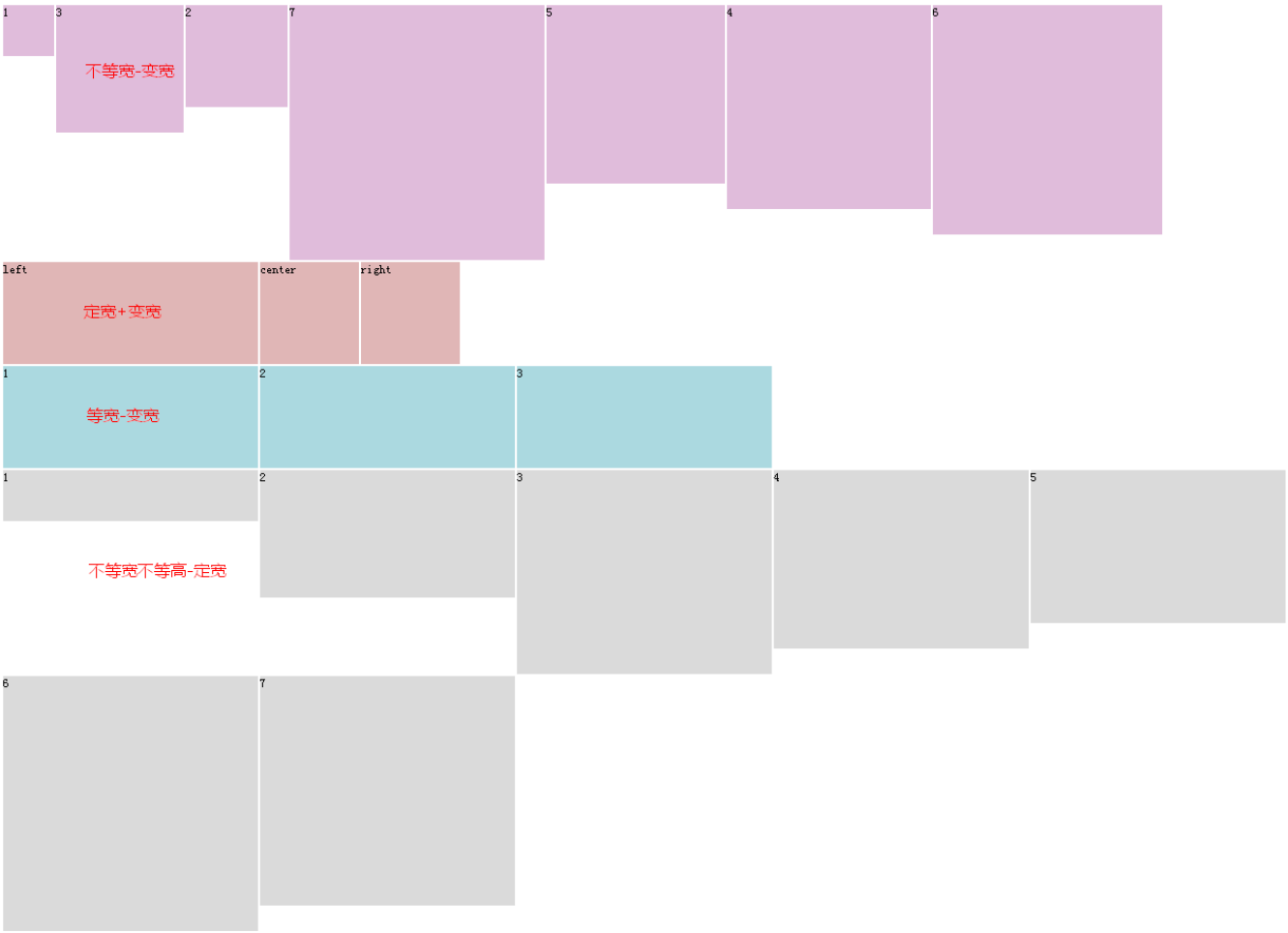
HTML

```
<h1>不等宽不等高（定宽） </h1>
<div class="box box1">
    <div class="flex-box" style="width:100px;height:100px;">1</div>
    <div class="flex-box" style="width:250px;height:250px;">3</div>
    <div class="flex-box" style="width:200px;height:200px;">2</div>
    <div class="flex-box" style="width:350px;height:350px;">5</div>
    <div class="flex-box" style="width:400px;height:400px;">4</div>
    <div class="flex-box" style="width:500px;height:500px;">7</div>
    <div class="flex-box" style="width:450px;height:450px;">6</div>
</div>
<h1>不 等宽 等高（ 定宽+变宽）</h1>
<div class="box box2">
    <div class="left flex-box">left</div>
    <div class="center flex-box">center</div>
    <div class="right flex-box">right</div>
</div>
<h1>等宽 等高（变宽） </h1>
<div class="box box3">
    <div class="flex-box">1</div>
    <div class="flex-box">2</div>
```

```
    <div class="flex-box">3</div>
</div>
<h1>等宽不等高 （定宽） </h1>
<div class="box box4">
    <div class="flex-box" style="height:100px;">1</div>
    <div class="flex-box" style="height:250px;">2</div>
    <div class="flex-box" style="height:400px;">3</div>
    <div class="flex-box" style="height:350px;">4</div>
    <div class="flex-box" style="height:300px;">5</div>
    <div class="flex-box" style="height:500px;">6</div>
    <div class="flex-box" style="height:450px;">7</div>
</div>
```

CSS

```
.box {
    display: -webkit-flex;
    display: flex;
    flex-direction: row;
    flex-wrap:nowrap;
    justify-content: flex-start;
    align-items: stretch;
    align-content: flex-start;
}
.flex-box{
    height:200px;
    width:500px;
    background-color:#ddd;
    border:1px solid #fff;
}
.box1 .flex-box{
    background-color:#E0BCDB;
}
.box2 .flex-box{
    background-color:#E0B6B6;
   width:800px;
}
.left{
    flex-shrink:0;//空间不足时不允许左侧缩小
}
.box3 .flex-box{
    background-color:#ABD9E0;
    width:900px;
}
.box4{
    flex-wrap: wrap;//空间不足换行
}
.box4 .flex-box{
    background-color:#dadada;
}
```



32.border

下述有关 border:none 以及 border:0 的区别，描述错误的是？

正确答案: C D　你的答案: A (错误)

border:none 表示边框样式无

border:0 表示边框宽度为 0

当定义了 border:none，即隐藏了边框的显示，实际就是边框宽度为 0

当定义边框时，仅设置边框宽度也可以达到显示的效果

解析

C:当定义 border:none 时，表示无边框样式，浏览器并不会对边框进行渲染，也就没有实际的宽度；

D:定义边框时，除了设置宽度外，还必须设置边框的样式才能显示出来。

33.换行

通过使用 word-break 属性，可以让浏览器实现在任意位置的换行。

值	描述
normal	使用浏览器默认的换行规则。
break-all	允许在单词内换行。
keep-all	只能在半角空格或连字符处换行。

white-space 属性设置如何处理元素内的空白。

值	描述
normal	默认。空白会被浏览器忽略。
pre	空白会被浏览器保留。其行为方式类似 HTML 中的 <pre> 标签。
nowrap	文本不会换行，文本会在在同一行上继续，直到遇到 标签为止。
pre-wrap	保留空白符序列，但是正常地进行换行。
pre-line	合并空白符序列，但是保留换行符。
inherit	规定应该从父元素继承 white-space 属性的值。

34. 去除 inline-block 元素间间距的 N 种方法

http://www.zhangxinxu.com/wordpress/2012/04/inline-block-space-remove-%E5%8E%BB%E9%99%A4%E9%97%B4%E8%B7%9D/

问题：

<input /> <input type="submit" />

间距就来了~~

二、方法之移除空格

元素间留白间距出现的原因就是标签段之间的空格，因此，去掉 HTML 中的空格，自然间距就木有了。考虑到代码可读性，显然连成一行的写法是不可取的，我们可以：

```
<div class="space">
    <a href="##">
    惆怅</a><a href="##">
    淡定</a><a href="##">
    热血</a>
</div>
```

或者是：

```
<div class="space">
    <a href="##">惆怅</a
    ><a href="##">淡定</a
    ><a href="##">热血</a>
</div>
```

或者是借助 HTML 注释：

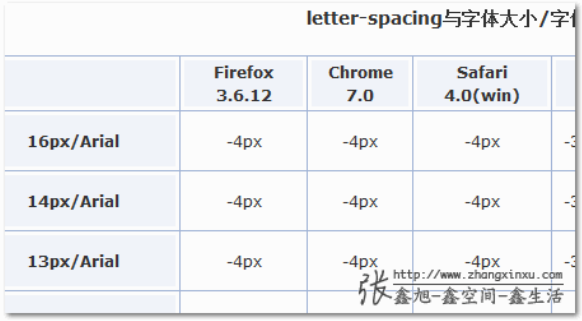
```
<div class="space">
    <a href="##">惆怅</a><!--
    --><a href="##">淡定</a><!--
    --><a href="##">热血</a>
</div>
```

等。

三、使用 margin 负值

```
.space a {
    display: inline-block;
    margin-right: -3px;
}
```

margin 负值的大小与上下文的字体和文字大小相关，其中，间距对应大小值可以参见我之前“基于 display:inline-block 的列表布局”一文 part 6 的统计表格：



letter-spacing与字体大小/字

	Firefox 3.6.12	Chrome 7.0	Safari 4.0(win)	
16px/Arial	-4px	-4px	-4px	-3
14px/Arial	-4px	-4px	-4px	-3
13px/Arial	-4px	-4px	-4px	-3

例如，对于 12 像素大小的上下文，Arial 字体的 margin 负值为-3 像素，Tahoma 和 Verdana 就是-4 像素，而 Geneva 为-6 像素。

由于外部环境的不确定性，以及最后一个元素多出的父 margin 值等问题，这个方法不适合大规模使用。

四、让闭合标签吃胶囊

如下处理：

```
<div class="space">
    <a href="##">惆怅
    <a href="##">淡定
    <a href="##">热血</a>
</div>
```

注意，为了向下兼容 IE6/IE7 等喝蒙牛长大的浏览器，最后一个列表的标签的结束（闭合）标签不能丢。

在 HTML5 中，我们直接：

```
<div class="space">
    <a href="##">惆怅
    <a href="##">淡定
    <a href="##">热血
</div>
```

好吧，虽然感觉上有点怪怪的，但是，这是 OK 的。

五、使用 font-size:0

类似下面的代码：

```
.space {
    font-size: 0;
}
.space a {
    font-size: 12px;
}
```

这个方法，基本上可以解决大部分浏览器下 inline-block 元素之间的间距(IE7 等浏览器有时候会有 1 像素的间距)。

不过有个浏览器，就是 Chrome，其默认有最小字体大小限制，因为，考虑到兼容性，我们还需要添加：

类似下面的代码：

```
.space {
    font-size: 0;
    -webkit-text-size-adjust:none;
}
```

补充：根据小杜在评论中中的说法，目前 Chrome 浏览器已经取消了最小字体限制。因此，上面的-webkit-text-size-

adjust:none;代码估计时日不多了。

六、使用 letter-spacing

类似下面的代码：

```
.space {
    letter-spacing: -3px;
}
.space a {
    letter-spacing: 0;
}
```

根据我去年的测试，该方法可以搞定基本上所有浏览器，包括吃“东鞋”、“西毒(胶囊)”、“南地(沟油)”、“北钙(三鹿)”的 IE6/IE7 浏览器，不过 Opera 浏览器下有蛋疼的问题：最小间距 1 像素，然后，letter-spacing 再小就还原了。

七、使用 word-spacing

类似下面代码：

```
.space {
    word-spacing: -6px;
}
.space a {
    word-spacing: 0;
}
```

一个是字符间距(letter-spacing)一个是单词间距(word-spacing)，大同小异。据我测试，word-spacing 的负值只要大到一定程度，其兼容性上的差异就可以被忽略。因为，貌似，word-spacing 即使负值很大，也不会发生重叠。

如果您使用 Chrome 浏览器，可能看到的是间距依旧存在。确实是有该问题，原因我是不清楚，不过我知道，可以添加 display: table;或 display:inline-table;让 Chrome 浏览器也变得乖巧。

```
.space {
    display: inline-table;
    word-spacing: -6px;
}
```

八、其他成品方法

下面展示的是 YUI 3 CSS Grids 使用 letter-spacing 和 word-spacing 去除格栅单元见间隔方法（注意，其针对的是 block 水平的元素，因此对 IE8-浏览器做了 hack 处理）：

```
.yui3-g {
    letter-spacing: -0.31em; /* webkit */
    *letter-spacing: normal; /* IE < 8  重置  */
    word-spacing: -0.43em; /* IE < 8 && gecko */
}

.yui3-u {
    display: inline-block;
    zoom: 1; *display: inline; /* IE < 8:  伪造  inline-block */
    letter-spacing: normal;
    word-spacing: normal;
    vertical-align: top;
}
```

以下是一个名叫 RayM 的人提供的方法：

```
li {
    display:inline-block;
    background: orange;
    padding:10px;
    word-spacing:0;
```

```
        }
ul {
        width:100%;
        display:table;    /* 调教 webkit*/
        word-spacing:-1em;
}
.nav li { *display:inline;}
```