### 1. 几种布局 1、静态布局（Static Layout）

即传统Web设计，对于PC设计一个Layout，在屏幕宽高有调整时，使用横向和竖向的滚动条来查阅被遮掩部分；   
意思就是不管浏览器尺寸具体是多少。

**2、弹性布局**

弹性布局是CSS3引入的强大的布局方式，用来替代以前Web开发人员使用的一些复杂而易错hacks方法（如使用float进行类似流式布局）。

其中flex-flow是flex-direction和flex-wrap属性的简写方式，语法如下：

flex-flow: <flex-direction> || <flex-wrap>

flex-direction: row（初始值） | row-reverse | column | column-reverse

flex-wrap: nowrap（初始值） | wrap | wrap-reverse

flex-direction定义了弹性项目在弹性容器中的放置方向，默认是row，即行内方向（一般而言是由左往右，但注意这个和书写模式有关）。

flex-wrap定义是否需要拆行以使得弹性项目能被容器包含。\*-reverse代表相反的方向。

两者结合起来即flex-flow属性就确定了弹性容器在main axis和cross axis两个方向上的显示方式.

**3、自适应布局（Adaptive Layout）**

自适应布局（Adaptive）的特点是分别为不同的屏幕分辨率定义布局。布局切换时页面元素发生改变，但在每个布局中，页面元素不随窗口大小的调整发生变化。   
你可以把自适应布局看作是静态布局的一个系列。   
就是说你看到的页面，里面元素的位置会变化而大小不会变化；

**4、流式布局（Liquid Layout）**

流式布局（Liquid）的特点（也叫”Fluid”) 是页面元素的宽度按照屏幕进行适配调整，主要的问题是如果屏幕尺度跨度太大，那么在相对其原始设计而言过小或过大的屏幕上不能正常显示   
。   
你看到的页面，元素的大小会变化而位置不会变化——这就导致如果屏幕太大或者太小都会导致元素无法正常显示。

1. 响应式布局  
   <meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no">
2. 利用媒体查询

***方法1：***

*/\*当你的屏幕大于1224px时采用这个样式 一行8列\*/*

**@media all and** (**min-width**: 1224**px**) {

**#div1**,**#div2**,**#div3**,**#div4**,**#div5**,**#div6**,**#div7**,**#div8**{

**float**: **left**;

**width**: 12.5%;

}

}

*/\*当屏幕宽度大于968px的时候采用该样式 2行4列*

*大小范围一定要明确指定不然样式可能会被覆盖\*/*

**@media all and** (**min-width**: 968**px**) **and** (**max-width**: 1224**px**){

**#div1**,**#div2**,**#div3**,**#div4**,**#div5**,**#div6**,**#div7**,**#div8**{

**float**: **right**;

**width**: 25%;

}

}

*/\*当屏幕宽度大于320px的时候采用该样式 4行2列\*/*

**@media all and** (**min-width**: 380**px**) **and** (**max-width**: 968**px**){

**#div1**,**#div2**,**#div3**,**#div4**,**#div5**,**#div6**,**#div7**,**#div8**{

**float**: **right**;

**width**: 50%;

}

}

**@media all and** (**max-width**: 380**px**){

**#div1**,**#div2**,**#div3**,**#div4**,**#div5**,**#div6**,**#div7**,**#div8**{

**float**: **left**;

**width**: 100%;

}

}

**方法2：**

<**style**>

*/\*注意引入外部样式后不能再在style里面添加其他样式了，对比script标签对比记忆\*/*

**@import "../../../css/css0.css"**;

**@import "../../../css/css1.css" all and** (**min-width**: 1024**px**);

**@import "../../../css/css2.css" all and** (**min-width**: 768**px**) **and** (**max-width**: 1024**px**);

**@import "../../../css/css3.css" all and** (**min-width**: 320**px**) **and** (**max-width**: 768**px**);

</**style**>

方法3：

<**link rel="stylesheet" href="../../../css/css0.css"**>

<**link rel="stylesheet" href="../../../css/css1.css" media="all and (min-width:1024px)"**>

<**link rel="stylesheet" href="../../../css/css2.css" media="all and (min-width:768px) and (max-width:1024px)"**>

<**link rel="stylesheet" href="../../../css/css3.css" media="all and (max-width:768px)"**>

媒体查询常用属性

<http://caibaojian.com/css3/properties/media-queries/index.htm>

window.devicePixelRatio是设备上物理像素和设备独立像素(device-independent pixels (dips))的比例。  
公式表示就是：window.devicePixelRatio = 物理像素 / dips

Example: BootStrap

* REM  
  rem（font size of the root element）是指相对于根元素的字体大小的单位。简单的说它就是一个相对单位。看到rem大家一定会想起em单位，em（font size of the element）是指相对于父元素的字体大小的单位。它们之间其实很相似，只不过一个计算的规则是依赖根元素一个是依赖父元素计算。  
    
  **使用方法：**  
  1.html中默认的 font-size:16px; 也就是 1rem = 16px   
  2.以设计宽度为750px为例，那么定义 font-size:50px; 那么 1rem = 50px，比例为：750/50=15   
  3.定义页面中的尺寸，就是: 宽度/50 rem（比例为15）。例如：设计稿中dom（20px）的尺寸为：20px/50=0.4rem   
  4.针对不同屏幕尺寸，按同比例设置font-size（比例仍为15）。相当于1rem 变成对应大小（font-size变小多少，1rem就同比例变小多少）。   
    
    
    
    
    
  **Example: Flexible.js**动态改写标签
* 给<html>元素添加data-dpr属性，并且动态改写data-dpr的值
* 给<html>元素添加font-size属性，并且动态改写font-size的值