**说明:本部分面试题均为HTML5及CSS3相对较难的面试题,比较偏经验类型,各位学员好好理解.**

**什么叫优雅降级和渐进增强？**

渐进增强:progressive enhancement,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:graceful degradation,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。

区别：

优雅降级:是从复杂的现状开始，并试图减少用户体验的供给；

渐进增强:是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来的环境的需要.

**如果需要手动编写css关键帧动画，你认为最小时间间隔是多少？为什么？**

多数显示器默认频率是60hz，所以为1/60\*1000=16.7ms

**请描述一下sessionStorage和localStorage的区别？**

sessionStorage用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储。

而localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。  
**web storage和cookie的区别** Web Storage的概念和cookie相似，区别是它是为了更大容量存储设计的。Cookie的大小是受限的，并且每次你请求一个新的页面的时候Cookie都会被发送过去，这样无形中浪费了带宽，另外cookie还需要指定作用域，不可以跨域调用。  
 除此之外，Web Storage拥有setItem,getItem,removeItem,clear等方法，不像cookie需要前端开发者自己封装setCookie，getCookie。但是Cookie也是不可以或缺的：Cookie的作用是与服务器进行交互，作为HTTP规范的一部分而存在 ，而Web Storage仅仅是为了在本地“存储”数据而生。

**什么是CSS Hack?**

一般来说是针对不同的浏览器写不同的CSS,就是 CSS Hack。

IE浏览器Hack一般又分为三种，条件Hack、属性级Hack、选择符Hack

**CSS实现垂直水平居中有几种方式?**

**未知宽高:**

**用inline-block和vertical-align来实现居中;**

<!DOCTYPE **html**>  
<**html lang="en"**>  
 <**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
 <**style**>  
 **#container**{  
 **text-align**: **center**;  
 **height**: 400**px**;  
 **background**: **#4dc71f**;  
 }  
 **#container**:**before**{  
 **content**: **""**;  
 **height**: 100%;  
 **display**: **inline-block**;  
 **vertical-align**: **middle**;  
 **margin-right**: -0.25**em**;  
 }  
 **#center-div**{  
 **display**: **inline-block**;  
 **vertical-align**: **middle**;  
 **background**: **#2aabd2**;  
   
 }  
 </**style**>  
 </**head**>  
 <**body**>  
 <**div id="container"**>  
 <**div id="center-div"**>  
 xxx  
 </**div**>  
 </**div**>  
 </**body**>  
</**html**>

**使用table-cell，inline-block实现水平垂直居中;**

<!DOCTYPE **html**>  
<**html lang="en"**>  
 <**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
 <**style**>  
 **#container**{  
 **display**: **table-cell**;  
 **text-align**: **center**;  
 **vertical-align**: **middle**;  
 **height**: 300**px**;  
 **background**: **#ccc**;  
 }  
 **#center-div**{  
 **display**: **inline-block**;  
 }  
 </**style**>  
 </**head**>  
 <**body**>  
 <**div id="container"**>  
 <**div id="center-div"**>  
 xxx  
 </**div**>  
 </**div**>  
 </**body**>  
</**html**>

**使用css3中的transform来时实现水平垂直居中**

<!DOCTYPE **html**>  
<**html lang="en"**>  
 <**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
 <**style**>  
 **#container**{  
 **position**: **relative**;  
 **height**: 200**px**;  
 **background**: **#333**;  
 }  
 **#center-div**{  
 **position**: **absolute**;  
 **top**:50%;  
 **left**: 50%;  
 **transform**: **translate**(-50%,-50%);  
 }  
 </**style**>  
 </**head**>  
 <**body**>  
 <**div id="container"**>  
 <**div id="center-div"**>  
 xxx  
 </**div**>  
 </**div**>  
 </**body**>  
</**html**>

**已知宽高:**

**用相对绝对定位和负边距实现上下左右居中**

<!DOCTYPE **html**>  
<**html lang="en"**>  
 <**head**>  
 <**meta charset="utf-8"**>  
 <**title**>test</**title**>  
 <**style**>  
 .**div2**{  
 **height**: 600**px**;  
 **width**: 600**px**;  
 **position**: **relative**;  
 **border**: 2**px solid #000**;  
 }  
 .**img2**{  
 **height**: 200**px**;  
 **width**: 200**px**;  
 **position**: **relative**;  
 **top**: 50%;  
 **left**: 50%;  
 **margin**: -100**px** 0 0 -100**px**;  
 }  
 </**style**>  
 </**head**>  
 <**body**>  
 <**div class="div2"**>  
 <**img class="img2" src="images/hongbao.png"**>  
 </**div**>  
 </**body**>  
</**html**>

**利用绝对定位来实现居中**

<!DOCTYPE **html**>  
<**html lang="en"**>  
 <**head**>  
 <**meta charset="UTF-8"**>  
 <**title**>Title</**title**>  
 <**style**>  
 **#container**{  
 **text-align**: **center**;  
 **height**: 400**px**;  
 **background**: **#4dc71f**;  
 **position**: **relative**;  
 }  
 **#center-div**{  
 **position**: **absolute**;  
 **margin**: **auto**;  
 **top**: 0;  
 **right**: 0;  
 **left**:0;  
 **bottom**: 0;  
 **width**: 200**px**;  
 **height**: 200**px**;  
 **background**: **#2b669a**;  
 }  
 </**style**>  
 </**head**>  
 <**body**>  
 <**div id="container"**>  
 <**div id="center-div"**>  
 xxx  
 </**div**>  
 </**div**>  
 </**body**>  
</**html**>

**什么是reflow？什么会触发reflow？如何减少reflow？**

什么是reflow:

浏览器为了重新渲染部分或整个页面，重新计算页面元素位置和几何结构（geometries）的进程叫做 reflow。reflow 是一种浏览器中的用户拦截（user-blocking）操作.

如何触发reflow:

1. 调整浏览器窗口的大小;
2. 用 javascript 计算样式（[computed styles](http://www.w3.org/TR/1998/REC-CSS2-19980512/cascade.html#computed-value)）;
3. 在 DOM 中创建删除元素;
4. 改变元素的 class ;
5. CSS3 动画（animation）和过渡（transition）;
6. 使用 offsetWidth 和 offsetHeight：这一点很特别，你读一个 DOM 的 offsetWidth和 offsetHeight 属性同样会触发一下 Reflow，因为这两个属性需要依赖一些元素去计算。
7. 用户交互：用户可以通过 hover 一下 a 链接，在 input 里面输入文字，拖动浏览器的大小，改变字体大小，更换样式表或者字体等都会触发 reflow。

如何减少reflow:

1.减少不必要的 DOM 层级（DOM depth）。改变 DOM 树中的一级会导致所有层级的改变，上至根部，下至被改变节点的子节点。这导致大量时间耗费在执行 reflow 上面。

2.尽量减少 CSS 规则，去除未用到的 CSS。

3.如果做复杂的表现变化，如动画，让它脱离文档流。用绝对定位或 fixed 定位来完成。

4.避免不必要的复杂的 CSS 选择器，尤其是后代选择器（descendant selectors），因为为了匹配选择器将耗费更多的 CPU。

### 5. 慎改 class:在一个 DOM 树中，尽可能改那些没有特别多子元素 DOM 的 class，子元素少的可以改，多的不推荐。

**有关标准盒子模型和IE盒子模型,两者的总宽度是如何计算的?**

标准盒子模型:总宽度=左右外边距+左右border+左右padding+width

IE盒子模型:总宽度=左右外边距+width

**Css样式优先级顺序**

!important > style(内联) > Id(权重100) > class(权重10) > 标签（权重1） 同类别的样式，后面的会覆盖前面的。  
百度视频部门一道面试题是这样的：

<style>

.red{color:red;}

.blue{color:blue;}

</style>

<p class="blue red"></p>

<!-- 此时显示蓝色，样式的显示跟class里面的先后顺序无关，都是类选择器，后面的会覆盖前面的，因此蓝色覆盖红色的 -->

**现在有并排的三个Div框，如何实现三个div都自适应**

<!DOCTYPE **html**>  
<**html lang="zh-CN"**>  
 <**head**>  
 <**meta charset="utf-8"**/>  
 <**title**>等高布局</**title**>  
 <**style**>  
 **html** {  
 **font-size**: 10**px**;;  
 }  
 **body** {  
 **font-size**: 1.4**rem**;  
 }  
 .**box** {  
 **background-color**: **rgba**(200,200,200,0.7);  
 **margin**: 0 1**rem**;  
 **width**: 33.33%;  
 **padding**: 1**rem**;  
 }  
 .**box**:**nth-child**(2) {  
 **height**: 5**rem**;  
 **background-color**: **rgba**(200,210,230,0.7);  
 }  
 .**accordant** {  
 **display**: **table-row**;  
 }  
 .**table** {  
 **width**: 100%;  
 **display**: **table**;  
 }  
 .**table** .**accordant** {  
 **display**: **table-row**;  
 }  
 .**table** .**accordant** .**box** {  
 **display**: **table-cell**;  
 **vertical-align**: **middle**;  
 **text-align**: **center**;  
 }  
 </**style**>  
 </**head**>  
 <**body**>  
 <**div class="table"**>  
 <**div class="accordant"**>  
 <**div class="box"**>  
 <**p**>这是一个 Figure</**p**>  
 </**div**>  
 <**div class="box"**>  
 <**p**>这是一个 Figure</**p**>  
 </**div**>  
 <**div class="box"**>  
 <**p**>这是一个 Figure</**p**>  
 </**div**>  
 </**div**>  
 </**div**>  
 </**body**>  
</**html**>

**css全称叫做css层叠样式表，那么我想问一下，层叠样式表中的层叠，到底是什么意思呢？**

第一种回答：

层叠指的是样式的优先级，当产生冲突时以优先级高的为准。  
 1. 开发者样式>读者样式>浏览器样式（除非使用!important标记 ）  
 2. id选择符>（伪）类选择符>元素选择符  
 3. 权重相同时取后面定义的样式

第二种回答：

层叠就是浏览器对多个样式来源进行叠加，最终确定结果的过程.css之所以有“层叠”的概念，是因为有多个样式来源。其中css样式来源有5个，分别是内联样式（<a style="">）,内部样式(<style></style>),  
外部样式（写在css文件中的样式），浏览器用户自定义样式，浏览器默认样式；按照其来源优先级为内联样式>内部样式>外部样式>浏览器用户自定义样式>浏览器默认样式.按照选择器优先级为id >class>元素选择器如果有important,important优先级最高。

**自适应布局及响应式布局概念及区别**

概念：

自适应布局：指能使网页自动适应不同屏幕分辨率以显示在不同分辨率大小的终端设备上的网页设计方式及技术。

响应式布局：一个[网站](http://www.fgkj.cc/)能够兼容多个终端——而不是为每个终端做一个特定的版本。

区别：

自适应布局是为了让同一张网页自动适应不同尺寸大小、不同分辨率的屏幕，根据屏幕的宽度，自动调节网页内容的大小，但是无论怎样，他们的主体内容和布局是不会变化的；而响应式其实是自适应布局的一个升华，响应式布局在自适应布局能做到的功能基础之上还可以根据屏幕的大小自动的调整页面的展现方式，以及布局。总结的来说，自适应布局及响应式布局都是为了实现页面能够自动适应不同尺寸的屏幕、不同分辨率的屏幕已达到最佳显示效果的布局方式，区别在于自适应布局不会改变网页的主体内容和布局，而响应式布局会根据屏幕的大小自动调整页面的展现方式，以及布局。

**inline-block元素会有4px左右的空隙，请问是什么原因导致的，该如何解决**

4px的空隙是因为我们写代码时候的换行符所致。

解决办法：在inline-block的父元素中设置样式letter-spacing: -4px以恢复正常的显示。

**inline-block的元素默认的对齐方式是什么？有什么特殊情况？**

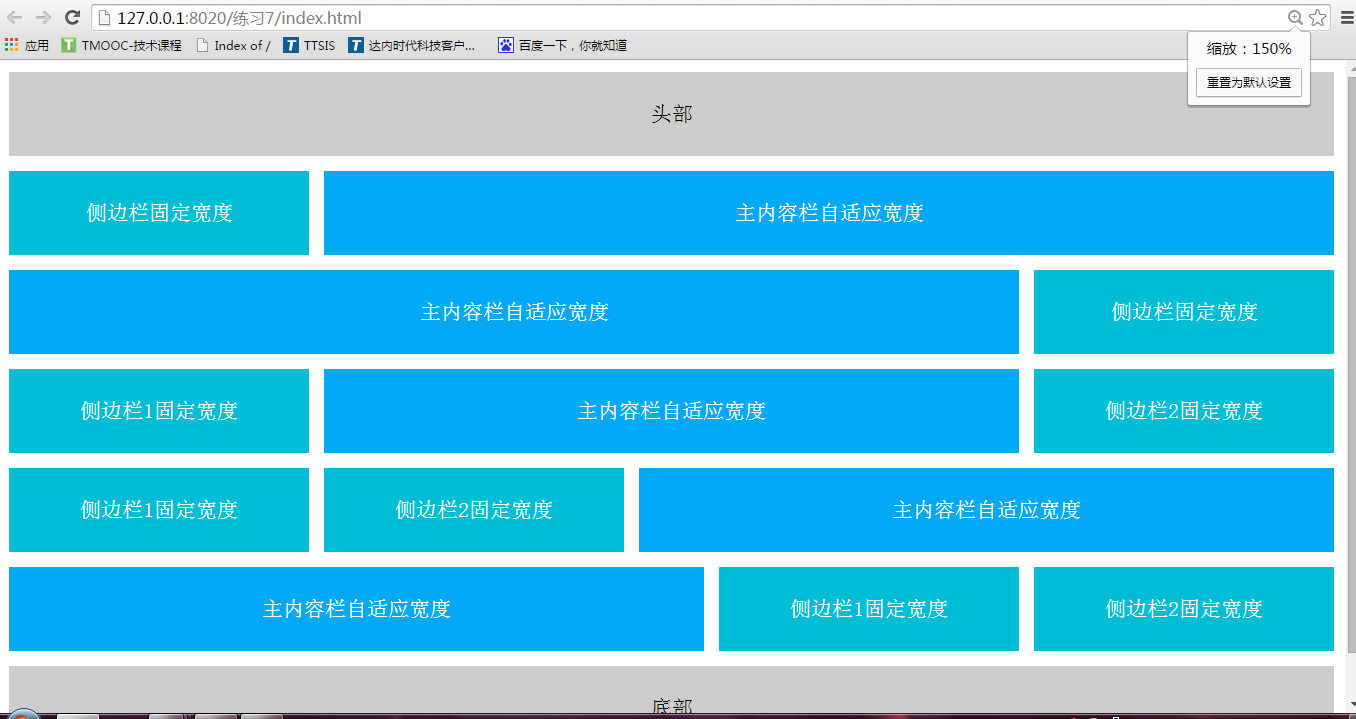
默认是基线对齐，且分为有文字和无文字两种情况：

1）无文字：默认已容器的margin-bottom下边缘为基线进行基线对齐。

2）有文字：默认以最后一行文字的下边缘为基线进行基线对齐。

**经典布局方式——圣杯布局。**

如图：110%缩放比例下效果

150%缩放比例下效果

无论百分比如何缩放，自适应的元素总能自适应，定宽的元素宽度总是不变，这就是圣杯布局。最早提出圣杯布局的是Kevin Cornell，在2006年提出的一个布局模型概念，在国内，最早是由阿里的UED团队推广。在中国也有叫法为“双飞翼布局”。圣杯布局有以下几点技术要求：1、三列布局，中间宽度自适应，两边定宽； 2、中间栏要在浏览器中优先展示渲染； 3、允许任意列的高度最高； 4、要求只用一个额外的DIV标签； 5、要求用最简单的CSS、最少的HACK语句。

圣杯布局示例代码如下:

<!DOCTYPE **html**>  
<**html**>  
 <**head**>  
 <**meta charset="utf-8"**>  
 <**meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"**>  
 <**title**>圣杯布局</**title**>  
  
 <**style type="text/css"**>  
 **body** {**background-color**: **#ffffff**; **font-size**:14**px**;}  
 **#hd**, **#ft** {**padding**:20**px** 3**px**; **background-color**: **#cccccc**; **text-align**: **center**;}  
 .**bd-lft**, .**bd-rgt**, .**bd-3-lr**, .**bd-3-ll**, .**bd-3-rr** {**margin**:10**px** 0; **min-width**:400**px**;}  
 .**main** {**background-color**: **#03a9f4**; **color**:**#ffffff**;}  
 .**aside**, .**aside-1**, .**aside-2** {**background-color**: **#00bcd4**; **color**:**#ffffff**;}  
 **p** {**margin**:0; **padding**:20**px**; **text-align**: **center**;}  
  
  
 */\* 左侧栏固定宽度，右侧自适应 \*/* .**bd-lft** {  
 **zoom**:1;  
 **overflow**:**hidden**;  
 **padding-left**:210**px**;  
 }  
 .**bd-lft** .**aside** {  
 **float**:**left**;  
 **width**:200**px**;  
 **margin-left**:-100%; */\*= -100%\*/* **position**:**relative**;  
 **left**:-210**px**; */\* = -parantNode.paddingLeft \*/* **\_left**: 0; */\*IE6 hack\*/* }  
 .**bd-lft** .**main** {  
 **float**:**left**;  
 **width**:100%;  
 }  
  
  
 */\* 右侧栏固定宽度，左侧自适应 \*/* .**bd-rgt** {  
 **zoom**:1;  
 **overflow**:**hidden**;  
 **padding-right**:210**px**;  
 }  
 .**bd-rgt** .**aside** {  
 **float**:**left**;  
 **width**:200**px**;  
 **margin-left**:-200**px**; */\* = -this.width \*/* **position**:**relative**;  
 **right**:-210**px**; */\* = -parantNode.paddingRight \*/* }  
 .**bd-rgt** .**main** {  
 **float**:**left**;  
 **width**:100%;  
 }  
  
  
 */\* 左中右 三栏自适应 \*/* .**bd-3-lr** {  
 **zoom**:1;  
 **overflow**:**hidden**;  
 **padding-left**:210**px**;  
 **padding-right**:210**px**;  
 }  
 .**bd-3-lr** .**main** {  
 **float**:**left**;  
 **width**:100%;  
 }  
 .**bd-3-lr** .**aside-1** {  
 **float**: **left**;  
 **width**:200**px**;  
 **margin-left**: -100%;  
   
 **position**:**relative**;  
 **left**: -210**px**;  
 **\_left**: 210**px**; */\*IE6 hack\*/* }  
 .**bd-3-lr** .**aside-2** {  
 **float**: **left**;  
 **width**:200**px**;  
 **margin-left**: -200**px**;  
   
 **position**:**relative**;  
 **right**: -210**px**;  
 }  
  
 */\* 都在左边，右侧自适应 \*/* .**bd-3-ll** {  
 **zoom**:1;  
 **overflow**:**hidden**;  
 **padding-left**:420**px**;  
 }  
 .**bd-3-ll** .**main** {  
 **float**:**left**;  
 **width**:100%;  
 }  
 .**bd-3-ll** .**aside-1** {  
 **float**: **left**;  
 **width**:200**px**;  
 **margin-left**: -100%;  
   
 **position**:**relative**;  
 **left**: -420**px**;  
 **\_left**: 0**px**; */\*IE6 hack\*/* }  
 .**bd-3-ll** .**aside-2** {  
 **float**: **left**;  
 **width**:200**px**;  
 **margin-left**: -100%;  
   
 **position**:**relative**;  
 **left**: -210**px**;  
 **\_left**: 210**px**; */\*IE6 hack\*/* }  
  
 */\* 都在右边，左侧自适应 \*/* .**bd-3-rr** {  
 **zoom**:1;  
 **overflow**:**hidden**;  
 **padding-right**:420**px**;  
 }  
 .**bd-3-rr** .**main** {  
 **float**:**left**;  
 **width**:100%;  
 }  
 .**bd-3-rr** .**aside-1** {  
 **float**: **left**;  
 **width**:200**px**;  
 **margin-left**: -200**px**;  
   
 **position**:**relative**;  
 **right**: -210**px**;  
 }  
 .**bd-3-rr** .**aside-2** {  
 **float**: **left**;  
 **width**:200**px**;  
 **margin-left**: -200**px**;  
   
 **position**:**relative**;  
 **right**: -420**px**;  
 }  
 </**style**>  
  
 </**head**>  
 <**body**>  
  
 <**div id="hd"**>头部</**div**>  
   
 <**div class="bd-lft"**>  
 <**div class="main"**>  
 <**p**>主内容栏自适应宽度</**p**>  
 </**div**>  
   
 <**div class="aside"**>  
 <**p**>侧边栏固定宽度</**p**>  
 </**div**>  
 </**div**>  
   
 <**div class="bd-rgt"**>  
 <**div class="main"**>  
 <**p**>主内容栏自适应宽度</**p**>  
 </**div**>  
   
 <**div class="aside"**>  
 <**p**>侧边栏固定宽度</**p**>  
 </**div**>  
 </**div**>  
   
 <**div class="bd-3-lr"**>  
 <**div class="main"**>  
 <**p**>主内容栏自适应宽度</**p**>  
 </**div**>  
   
 <**div class="aside-1"**>  
 <**p**>侧边栏1固定宽度</**p**>  
 </**div**>  
   
 <**div class="aside-2"**>  
 <**p**>侧边栏2固定宽度</**p**>  
 </**div**>  
 </**div**>  
   
 <**div class="bd-3-ll"**>  
 <**div class="main"**>  
 <**p**>主内容栏自适应宽度</**p**>  
 </**div**>  
   
 <**div class="aside-1"**>  
 <**p**>侧边栏1固定宽度</**p**>  
 </**div**>  
   
 <**div class="aside-2"**>  
 <**p**>侧边栏2固定宽度</**p**>  
 </**div**>  
 </**div**>  
   
 <**div class="bd-3-rr"**>  
 <**div class="main"**>  
 <**p**>主内容栏自适应宽度</**p**>  
 </**div**>  
   
 <**div class="aside-1"**>  
 <**p**>侧边栏1固定宽度</**p**>  
 </**div**>  
   
 <**div class="aside-2"**>  
 <**p**>侧边栏2固定宽度</**p**>  
 </**div**>  
 </**div**>  
   
 <**div id="ft"**>底部</**div**>  
  
 </**body**>  
</**html**>

圣杯布局原理非常简单，左右侧边栏定宽并浮动，中部内容区放最后不浮动、默认width：auto并设置相应外边距，让左右侧边栏浮动到上面。注意：子元素设置为浮动之后，父对象的高度就坍塌了，需要设置父对象后的元素清除浮动，这样父对象的高度才能被浮动子元素撑起来了。

**CSS3中的Media Query（媒介查询）是什么？**

通过不同的媒介类型和条件定义样式表规则。媒介查询让CSS可以更精确作用于不同的媒介类型和同一媒介的不同条件。媒介查询的大部分媒介特性都接受min和max用于表达”大于或等于”和”小于或等于”。如：width会有min-width和max-width媒介查询可以被用在CSS中的@media和@import规则上，也可以被用在HTML和XML中。通过这个标签属性，我们可以很方便的在不同的设备下实现丰富的界面，特别是移动设备，将会运用更加的广泛。

**media query能够获取哪些值？**

设备的宽和高device-width，device-height显示屏幕/触觉设备。

渲染窗口的宽和高width，height显示屏幕/触觉设备。

设备的手持方向，横向还是竖向orientation（portrait|lanscape）和打印机等。

画面比例aspect-ratio点阵打印机等。

设备比例device-aspect-ratio-点阵打印机等。

对象颜色或颜色列表color，color-index显示屏幕。

设备的分辨率resolution。

**4.移动端响应式布局中常见的主流js框架(除bootstrap之外的其他常见js框架)**

1. SkeIjs。轻量，仅包含了一个JS文件（大小有18kb），可以处理任何类型的复杂布局，简化了响应式布局的构建。

2.UI Kit。基于Less开发，是一款轻量级、模块化的前端框架，可快速构建强大的web前端界面,且提供了全面的HTML、CSS及JS组件，使用简单、定制方便。

3.GetWebplate. 一款优秀的前端框架，可使开发者专注于站点或应用的开发，它提供了稳健的布局引擎、全局按钮元素和可定制性的表格，同时还包括了jQuery插件和大量工具。

4.Cardinal、一个适用于移动项目的CSS框架，包含很多有用的默认样式、矢量字体、可重用的模块以及一个简单的响应式模块系统。

5.Schema。使用Less构建的功能强大但很轻量的前端UI框架，提供了丰富的CSS合集，可作为项目的前端基础。

6.Kickoff。可以用来创建可扩展性和响应式的站点，对移动设备比较友好。

7.Pure。来自雅虎的一套CSS模块，可以将其作为基础用在Web项目中，非常小巧。

8.Base。一款基于语义的、轻量和可扩展的框架，可用来帮你快速创建一个响应式站点。

9.Spark。一款轻量、以移动为先的web响应式设计框架，简单易用，可定制性强。

10.Flexbox grid。一个基于flex显示属性的网格系统。

网上能搜索到的主流响应式布局js框架很多，感兴趣可自行做了解。