响应式页面开发

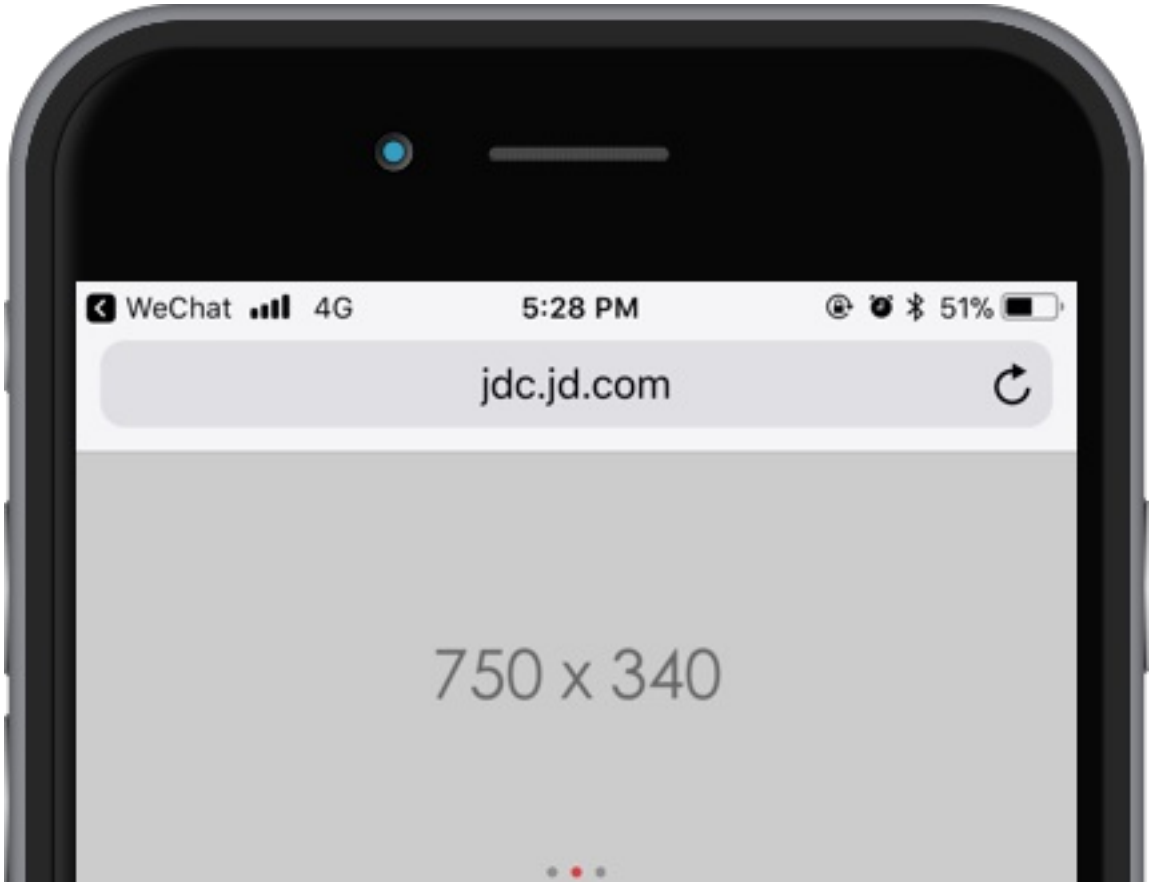
响应式页面开发的能力可以定义为：

> 利用一套代码实现页面的布局和排版以适配不同分辨率的设备。

响应式页面开发要求我们解决两大问题：

- 为不同特性（如横屏还是竖屏等）的浏览器视窗使用不同的样式代码
- 让页面元素的尺寸能够依据浏览器视窗尺寸变化而平滑变化

本小节的学习目标是学会解决上述问题并能够开发这样一个经典的移动端响应式页面：



我们分 3 个步骤来实现这样一个响应式页面。

步骤 1 – 添加 viewport meta 标签

在页头 head 标签内添加 viewport meta 标签是实现响应式页面的第一步。

viewport meta 标签源于 Apple 公司，用来定义 iOS Safari 浏览器展示网页内容的可视范围及缩放比率。它虽然没有成为W3C标准，但是被其他绝大多数的移动端浏览器所支持（目前已知 IE Mobile 10 不支持）。W3C 尝试将 viewport meta 标签的功能进行标准化并通过 CSS 的 @viewport 规则来实现同样的功能，但这个标准目前还在草案中，兼容性也没有 viewport meta 标签好。

PageSpeed 准则

Google 网页性能分析工具 PageSpeed Insights (https://developers.google.com/speed/pagespeed/insights/) 的其中一条准则就是：

网页应在 head 标签内添加 viewport meta 标签，以便优化在移动设备上的展示效果，其推荐的设置为：

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

扩展阅读

- Mozilla《Using the viewport meta tag to control layout on mobile browsers》(https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag)
- Google《Configure the viewport》(https://developers.google.com/web/fundamentals/design-and-ux/responsive/#set-the-viewport)
- Mozilla《@viewport》(https://developer.mozilla.org/en-US/docs/Web/CSS/@viewport)

步骤 2 - 使用 Media Queries

Media Queries 是为指定特性的浏览器视窗应用指定样式的手段，可以看成是 CSS 样式的过滤器或拦截器，通常情况下它可以通过「@media 规则」结合「6 个查询参数」来拦截设备的浏览器特性（如显示类型、视窗高度、视窗宽度、横竖屏等），藉此可以为不同的特性应用不同的样式代码（相当于为不同的设备应用了不同的 CSS 样式）。

6 个参数

参数名称	参数描述）
min-width	当视窗宽度大于或等于指定值时，@media 规则下的样式将被应用
max-width	当视窗宽度小于或等于指定值时，@media 规则下的样式将被应用
min-height	当视窗高度大于或等于指定值时，@media 规则下的样式将被应用
max-height	当视窗高度小于或等于指定值时，@media 规则下的样式将被应用
orientation=portrait	当视窗高度大于或等于宽度时，@media 规则下的样式将被应用
orientation=landscape	当视窗宽度大于高度时，@media 规则下的样式将被应用

2 种用法

方法 1，使用 link 标签，根据指定特性引入特定的外部样式文件

```
<link rel="stylesheet" media="(max-width: 640px)" href="max-640px.css">
```

方法 2，直接在 style 标签或 样式文件内使用 @media 规则

```
@media (max-width: 640px) {
  /*当视窗宽度小于或等于 640px 时，这里的样式将生效*/
}
```

样式断点

Media Queries 所使用的查询参数的临界值又可称为「样式断点」。
在响应式页面开发过程中，对于「样式断点」我们需要掌握 2 个重要的技巧：

依据目标设备的分辨率，制定一套合适的样式断点，并为不同的断点定制必要的 CSS 样式。
移动端优先的页面，可使用 min-width 查询参数从小到大来定义断点。
如果我们页面的响应式设计要涵盖从手机到高清大屏幕，什么样的「样式断点」比较合理呢?

我们可以从业界一些热门可靠的 CSS 框架中寻找参考答案，例如 Bulma (https://bulma.io/)，其采用的「样式断点」有 5 个：

断点名称	断点描述）
mobile	移动设备断点，视窗宽度 ≤ 768 px
tablet	平板电脑设备断点，视窗宽度 ≥ 769 px
desktop	桌面电脑断点，视窗宽度 ≥ 1024 px
widescreen	宽屏电脑断点，视窗宽度 ≥ 1216 px
fullhd	高清宽屏电脑断点，视窗宽度 ≥ 1408 px

在实际工作中，「样式断点」的制定需要我们同视觉设计师一起沟通确认，因为视觉设计师可能需要根据不同的断点为页面设计不同的视觉表现。

一个小例子

如果针对 tablet 及以上的设备定制样式，我们就可以这样写了：

```
@media (min-width: 769px) {
  /* tablet 及以上的设备，页面背景色设置为红色 */
  body {
    background-color: red;
  }
}
```

课外作业

1. 使用桌面版的 Chrome 浏览器，打开 Google 的 在线 Media Queries 例子 (https://googlesamples.github.io/web-fundamentals/fundamentals/design-and-ux/responsive/media-queries.html) 直观感受下使用 Media Queries 的效果（请注意缩放浏览器窗口观察页面展示效果）
2. 了解 Bulma (https://bulma.io/) 框架

扩展阅读

- Google 《Responsive Web Design Basic》 (https://developers.google.com/web/fundamentals/design-and-ux/responsive/)

步骤 3 – 使用 Viewport 单位及 rem

Media Queries 只解决了「为不同特性的浏览器视窗使用不同的样式代码」的问题，而 Viewport 单位及 rem 的应用，则是为了解决第二个问题：让页面元素的尺寸能够依据浏览器视窗尺寸变化而平滑变化。

关于 Viewport 单位及 rem 单位的基本概念，可通过下面的扩展阅读进行回顾复习。

BTW：本文所提及的 Viewport，译为「视窗」，其含义与扩展阅读中相关文章中的「视口」一致。

方法 1 – 仅使用 vw 作为 CSS 长度单位

在仅使用 vw 单位作为唯一 CSS 单位时，我们需遵守：

1. 利用 Sass 函数将设计稿元素尺寸的像素单位转换为 vw 单位

```
// iPhone 6尺寸作为设计稿基准
$vw_base: 375;
@function vw($px) {
    @return ($px / $vm_base) * 100vw;
}
```

2. 无论是文本字号大小还是布局高宽、间距、留白等都使用 vw 作为 CSS 单位

```
.mod_nav {
    background-color: #fff;
    &_list {
        display: flex;
        padding: vw(15) vw(10) vw(10); // 内间距
        &_item {
            flex: 1;
            text-align: center;
            font-size: vw(10); // 字体大小
            &_logo {
                display: block;
                margin: 0 auto;
                width: vw(40); // 宽度
                height: vw(40); // 高度
                img {
                    display: block;
                    margin: 0 auto;
                    max-width: 100%;
                }
            }
            &_name {
                margin-top: vw(2);
            }
        }
    }
}
```

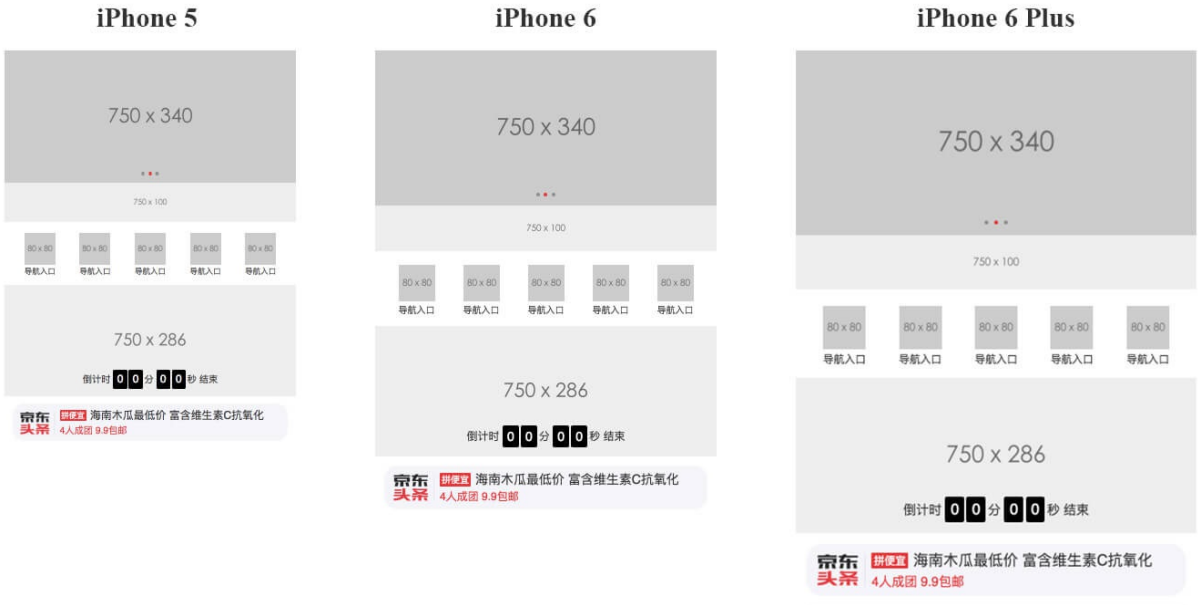
3. 1 物理像素线（也就是普通屏幕下 1px ，高清屏幕下 0.5px 的情况）采用 transform 属性 scale 实现

```
.mod_grid {
    position: relative;
    &::after {
        // 实现1物理像素的下边框线
        content: '';
        position: absolute;
        z-index: 1;
        pointer-events: none;
        background-color: #ddd;
        height: 1px;
        left: 0;
        right: 0;
        top: 0;
        @media only screen and (-webkit-min-
device-pixel-ratio: 2) {
            -webkit-transform: scaleY(0.5);
            -webkit-transform-origin: 50% 0%;
        }
    }
    ...
}
```

4. 对于需要保持高宽比的图，应改用 padding-top 实现

```
.mod_banner {
    position: relative;
    // 使用padding-top 实现宽高比为 100:750 的图片区域
    padding-top: percentage(100/750);
    height: 0;
    overflow: hidden;
    img {
        width: 100%;
        height: auto;
        position: absolute;
        left: 0;
        top: 0;
    }
}
```

由此，我们不需要增加其他任何额外的脚本代码就能够轻易实现一个常见布局的响应式页面，效果如下：



体验地址：[视口单位布局 —— vw 单位 (https://jdc.jd.com/demo/ting/vw_layout.html)](https://jdc.jd.com/demo/ting/vw_layout.html)

友情提醒：桌面版 Chrome 支持的字体大小默认不能小于 12PX，可通过 「chrome://settings/显示高级设置－网络内容－自定义字体－最小字号（滑到最小）」设置后再到模拟器里体验 DEMO。

方法 2 – vw 搭配 rem，寻找最优解

方法 1 实现的响应式页面虽然看起来适配得很好，但是你会发现由于它是利用 Viewport 单位实现的布局，依赖于视窗大小而自动缩放，无论视窗过大还是过小，它也随着视窗过大或者过小，失去了最大最小宽度的限制，有时候不一定是我们所期待的展示效果。试想一下一个 750px 宽的设计稿在 1920px 的大屏显示器上的糟糕样子。

当然，你可以不在乎移动端页面在 PC 上的展现效果，但如果有低成本却有效的办法来修复这样的小瑕疵，是真切可以为部分用户提升体验的。

我们可以结合 rem 单位来实现页面的布局。rem 弹性布局的核心在于根据视窗大小变化动态改变根元素的字体大小，那么我们可以通过以下步骤来进行优化：

1. 给根元素的字体大小设置随着视窗变化而变化的 vw 单位，这样就可以实现动态改变其大小
2. 其他元素的文本字号大小、布局高宽、间距、留白都使用 rem 单位
3. 限制根元素字体大小的最大最小值，配合 body 加上最大宽度和最小宽度，实现布局宽度的最大最小限制

核心代码实现如下：

```
// rem 单位换算：定为 75px 只是方便运算，750px-75px、640-64px、1080px-108px，如此类推
$vw_fontsize: 75; // iPhone 6尺寸的根元素大小基准值
@function rem($px) {
     @return ($px / $vw_fontsize ) * 1rem;
}
// 根元素大小使用 vw 单位
$vw_design: 750;
html {
    font-size: ($vw_fontsize / ($vw_design / 2)) * 100vw;
    // 同时，通过Media Queries 限制根元素最大最小值
    @media screen and (max-width: 320px) {
        font-size: 64px;
    }
    @media screen and (min-width: 540px) {
        font-size: 108px;
    }
}
// body 也增加最大最小宽度限制，避免默认100%宽度的 block 元素跟随 body 而过大过小
body {
    max-width: 540px;
    min-width: 320px;
}
```

体验地址：https://jdc.jd.com/demo/ting/vw_rem_layout.html (https://jdc.jd.com/demo/ting/vw_rem_layout.html)

扩展阅读

- Mozilla《Length》(https://developer.mozilla.org/en-US/docs/Web/CSS/length)

- 凹凸实验室 《利用视口单位实现适配布局》(https://aotu.io/notes/2017/04/28/2017-4-28-CSS-viewport-units/)

小结

在实际工作过程中，考虑到设计以及开发成本，视觉设计师是不大可能为每种不同分辨率的设备分别设计不同的稿子的，拿移动端页面来说，通常会以 iPhone 7 的分辨率（宽为 750 px）作为基准分辨率来出设计稿。因此「响应式页面开发」也便成为了移动互联网时代「H5 开发」的必备技能。

本小节所介绍的「利用 Viewport 单位及 rem 实现响应式页面」，相对于传统的 JavaScript 脚本结合 rem 的方式来得更简单优雅。