17-不一样的体验：交互设计和页面布局

你好，我是四火。

前几讲我们一直在 JavaScript 的代码中游走，这一讲我们来换换脑子，聊一聊界面设计，讲一讲交互和布局。这部分对于基于 Web 的全栈工程师来说，不只是技术栈特殊的一部分，还是一个能够给个人发展格局带来更多可能的部分。

1. 单页面应用

SPA，Single-Page Application，即单页应用，是一种互联网应用在用户交互方面的模型。

用户打开一个页面，大部分操作都在单个页面里面完成。和传统的 MPA（Multiple-Page Application）相比，用户体验上，SPA 省去了页面跳转的突兀感受和等待时间，用户体验更加桌面化，操作迅速、切换无缝；软件架构上，SPA 可以更彻底地落实前后端分离，后端彻底变成了一个只暴露 API 的服务，将不同终端的视图层的工作，特别是页面聚合搬到前端来完成（如有遗忘请回看 [第 09 讲]）。

对于许多大型应用，SPA 和 MPA 往往是在一定程度上结合起来使用的，比如新浪微博的网页版，用户可以浏览时间线，并作出一些转发评论、媒体播放等等操作，但是也可以跳到单独的单条微博的页面去。

但 SPA 也有着先天的缺陷。比如说，网页的 SEO（Search Engine Optimization）就是一个例子，单页应用上的操作，引起页面的状态改变，很难被搜索引擎敏锐地捕获到，并使之成为“可搜索到的”，因此 SPA 页面的 SEO 需要做额外的工作，关于这部分我会在第五章中介绍一些相关知识。

再比如，浏览器上网页操纵的功能，像前进、后退的功能，也需要一些特别的技巧才能支持，毕竟，和互联网整体翻天覆地的改变相比，浏览器的核心设计一直以来都变化缓慢，而地址栏、页面控制功能等等可都不是为着 SPA 考虑的。最后，SPA 通常会伴随着大大增加的在线应用复杂程度。

2. 渐进式增强

渐进式增强，即 Progressive Enhancement，强调的是可访问性，即允许不同能力的设备都能够访问网页的基本内容，使用网页的基本功能；但是，当用户使用更加先进的设备时，它能够给用户带来更强大的功能和更好的体验。

举一个小例子，对于非常极端和基础的设备，网页可以只加载 HTML；更高级一点的设备，则可以加载 CSS；而对于多数高级设备，则还需加载 JavaScript。这三种设备都可以使用网站的核心功能，但是只有最后一类设备可以使用全部功能。当然，实际的增强效果递进未必会那么极端，对于每一个递进，可以只是部分样式，部分标签，部分脚本等等。

渐进式增强的理念不仅仅可以对不同设备的访问带来普适性，对低端设备的访问更为友好，它还有一些其它优势。比如说，立足于网页基础的 HTML，可以让网页对于搜索引擎更加友好，因为 CSS 样式和 JavaScript 行为不一定能被搜索引擎解释执行并收录。

在这个过程中，HTML 的语义化变得越来越重要。语义化，指的就是让 HTML 页面体现出内容的结构性，即让 HTML 具备自解释能力，而不是一堆冷冰冰的缺乏语义的标签。你看下面的例子：

```
<article>
```

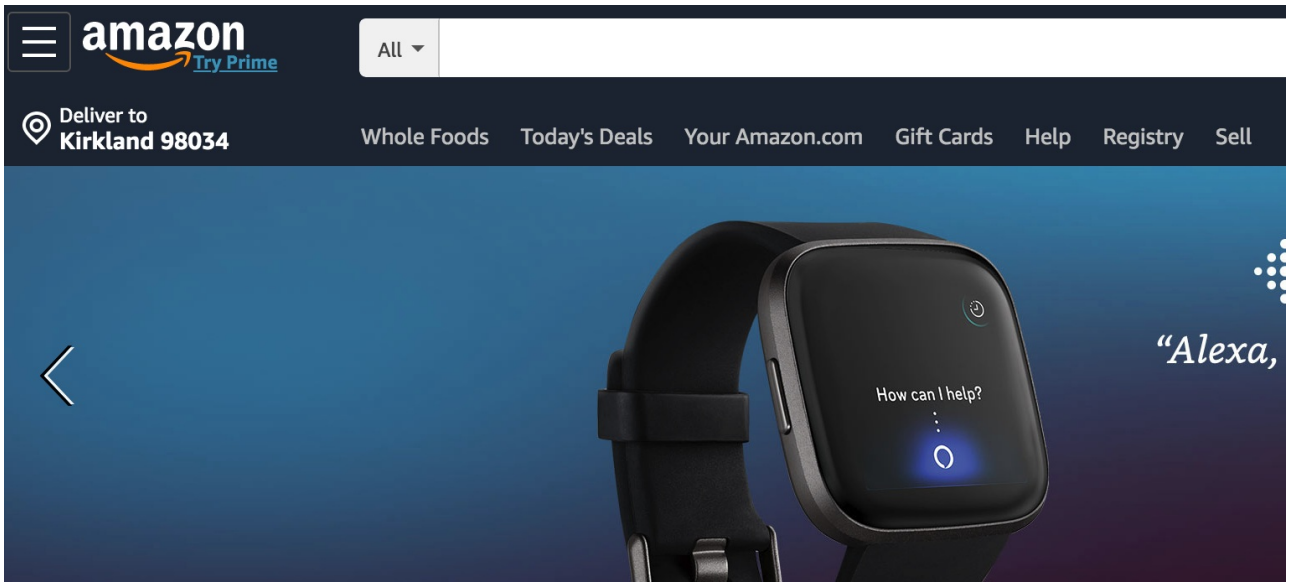
```
  <h1>Article Title</h1>
  <p>This is the content. Please read <em>carefully</em>.</p>
</article>
```

其中的 article 标签，就是具备语义的，表示着其中的内容是一篇文章，我们当然可以使用没有语义的 div 标签来代替，但 article 标签可以让阅读代码的人，浏览器，以及搜索引擎理解这部分的内容。h1 标签表示标题，p 标签表示段落，而其中的 em 标签表示的是需要强调（如果想使用无语义的标签，可以使用斜体标签 i 来代替），这些用在这里都是具备恰当语义的。

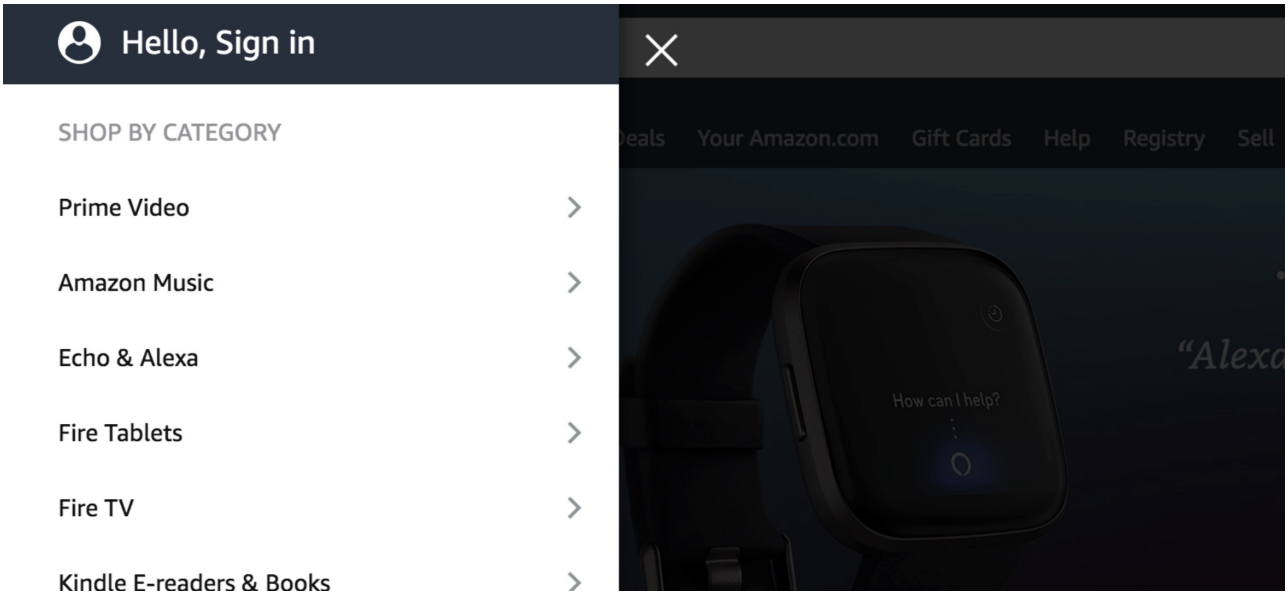
和渐进式增强一起谈论的，还有一个相反过程的设计理念，叫做优雅降级，Graceful Degradation，本质上它们说的是一个事儿。

优雅降级指的是，当高级特性和增强内容无法在用户的设备上展现的时候，依然能够给用户提供一个具备基本功能的，可以工作的应用版本。值得注意的是，优雅降级并非一定发生在用户设备的能力不匹配时，还有可能发生在服务器端资源出现瓶颈的时候，比如某些访问量特别大或者系统出现问题的时刻，资源紧张，服务端可以关闭某一些次要功能，降低一些用户体验，用几种核心资源来保证基础功能的正常运行。

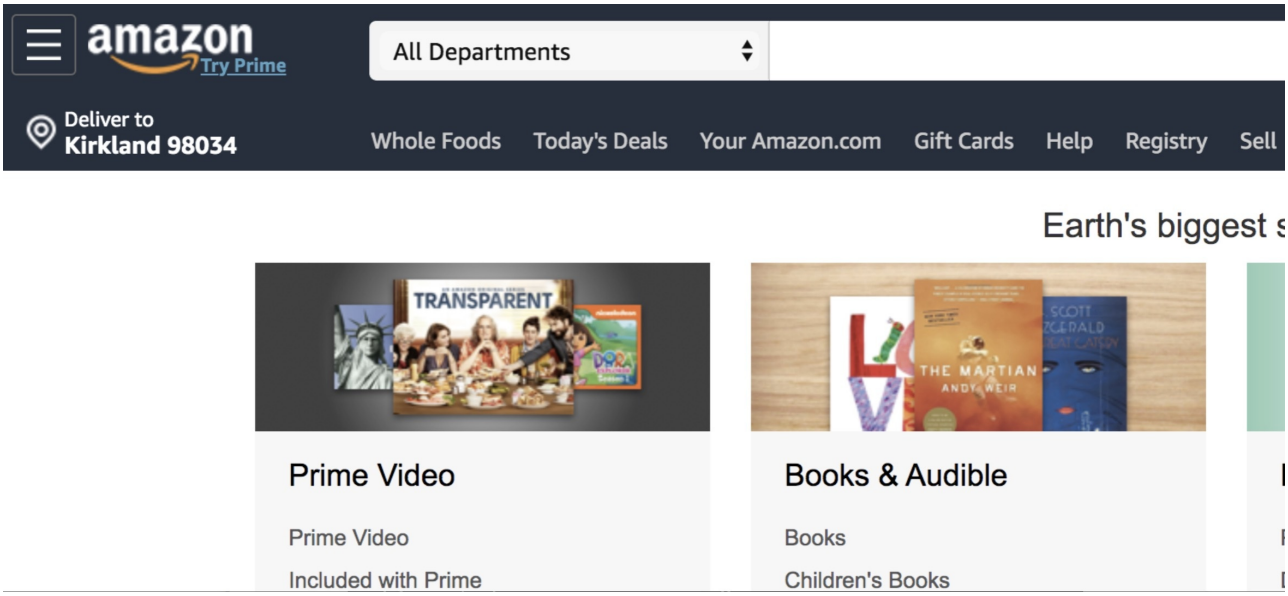
关于渐进增强和优雅降级，我来举一个 Amazon 网站设计的例子，希望它能帮助你进一步理解。如果你使用先进的 Web 浏览器访问 amazon.com，你会看到完整的功能：



点击左上角三横线的菜单图标，你将能看到利用 JavaScript 做出的弹出层的效果：



现在，我们点击 Chrome 地址栏的锁图标，点击 Site Settings，在设置中关闭对网站的 JavaScript 支持，刷新页面，显示还差不多。但是如果点击左上角的菜单图标，你就会发现，由于没有了 JavaScript，无法使用弹出层效果，它变成了一个链接，跳转到了商品分类：



你看，虽然没有了 JavaScript，遵循这种设计理念，在损失一部分用户体验的情况下 ，你可以继续使用网站，并且可以继续购物，其中的核心功能并没有丢失。

此外，还有一种可以拿来类比的设计理念，叫做**回归增强**，Regressive Enhancement。它要求为系统的特性设定基线，并应用到较老的设备和浏览器中。于是在设计网页特性时，我们可以按照高级设备的能力来进行，但是在实际开发的实施过程中，对于较低级的设备，提供一些其它的替代方法，让它们也模拟支持这些新特性。

比方说，HTML 5 的一些特性在偏老的 IE 浏览器中不支持，那么就可以使用 JavaScript 等替代方案实现出相似的效果。我们提到过的类库 jQuery 就遵循着回归增强的设计理念，在一定程度上屏蔽了不同浏览器的差异性。

举个实际例子，input 标签如果在偏老的浏览器中不支持 placeholder 属性，我们可以利用灰度字体的样式在 input 标签中显示实际内容来模拟这个功能。当用户将输入焦点移到 input 标签中，再将其从 input 中清空，以便用户能输入实际内容。

无论是渐进式增强、优雅降级，还是回归增强，都是为了在一定程度上照顾更多的不同能力的设备和浏览器，给用户带来“尽量好”的体验。但是我们在应用这样的设计理念时，需要把握这个度，毕竟，它不是无代价的，而是会增加前端设计开发的复杂性。

3. 响应式布局

响应式网页设计，即 RWD，Responsive Web Design，也有称之为**自适应性网页设计**，Adaptive Web Design，是一种网页设计方法，**目的是使得同一份网页，在不同的设备上有合适的展现。**几乎页面上所有的元素都可以遵循响应式布局，在不同的设备上产生不同的呈现，包括字体和图像等，但是我们讨论得最多的，却是布局。

我记得我刚参加工作的那几年，我们对于同一个页面在不同设备上的展示，考虑的最多的问题还是终端适配，并且这种适配还是基于协议的。例如，服务端是返回 Web 页面，WAP 1.0 页面（WML 语言描述），还是 WAP 2.0 页面（XHTML 语言描述）？那时候我们还很难去谈论用户体验有多么“合适”，对于这些低端的移动设备，我们充其量只能关心功能的实现是否能保证。

这部分，我们改变一下学习策略，来动动手，实现下简单的响应式布局页面。假如说我们需要实现一个具有 header、footer 的页面，并且他们需要填满宽度。而中间的主页面部分采用三列布局，左边列定宽，右边列也定宽，中间列宽度自由设置，但是要保证这三列排列起来也填满浏览器的宽度。

在往下阅读之前，你能否先想想，这该如何实现？

现在，我们在任意的工作文件夹下建立一个 responsive layout.html 文件，填写如下内容：

```
<!DOCTYPE html>
<html>
<head>
  <link href="style.css" type="text/css" rel="stylesheet">
</head>
<body>
  <header>header</header>
  <aside class="left">aside</aside>
  <aside class="right">aside</aside>
  <main class="middle">main</main>
  <footer>footer</footer>
</body>
</html>
```

你看，这个文件结构是很简单的，但是具备了我们所需要的要素，包括 header、中间三列以及 footer。这个页面将引入 style.css，因此，我们在同一目录下，建立 style.css：

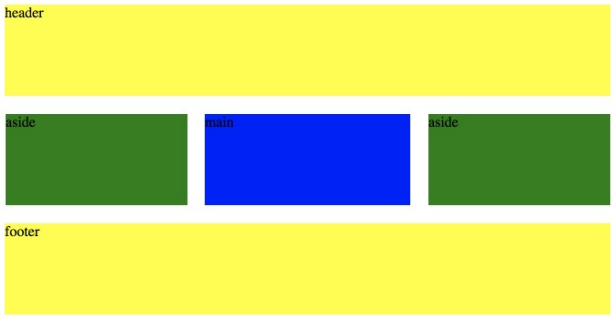
```
* {
  height: 100px;
  margin: 10px;
}
header, footer {
  background-color: YELLOW;
}
main {
  background-color: BLUE;
}
aside {
  background-color: GREEN;
}

.left {
  width: 200px;
  float: left;
}
.middle {
  margin: 20px 230px 20px 230px;
}
.right {
  width: 200px;
  float: right;
}
```

我来简单解读一下这个 CSS 文件。为了演示效果，所有的 DOM 对象都具备 100px 的高度，左边栏向左侧浮动排列，右边栏向右侧浮动排列，中间一列使用 margin 的方式给左右边栏留足位置。在排列这三列时，DOM 的顺序是左边栏 - 右边栏 - 中间栏，原因是，左右边栏是浮动样式，需要给他们排好以后，中间栏位无浮动，自动填满所有剩余空间。

看看效果吧，你可以拖动浏览器的边界，调整窗口的宽度，来模拟不同宽度的浏览器窗口下的效果。在较宽的浏览器下，它是这样的：

而在较窄的浏览器下，它是这样的：



注意这里的图片有缩放，但是每个矩形的高度实际上都是 100px。也就是说，中间蓝色的区域可以根据实际的宽度需要进行自适应的横向缩放，但是布局始终保持填满浏览器的宽度，也就是说，绿色的部分，始终是固定不变的。

但是，这样的显示有一个问题，在屏幕宽度较小时，比如手机屏幕，中间的蓝色区域会被挤得看不见。因此，我们希望在浏览器宽度小到一定程度的时候，显示成多行格式，而不进行左中右栏位的划分了，即从上到下包含 5 行：header、left aside、main、right aside 和 footer。

那么，这又该怎么实现？

其实也不难，我们需要先在 HTML 的头部增加：

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
```

这个 meta 标签指定了视口（View Port）的宽度为设备宽度，避免了任何手机端自动缩放的可能，同时也关闭了用户手动缩放的功能，这样网页会更像一个原生 app。

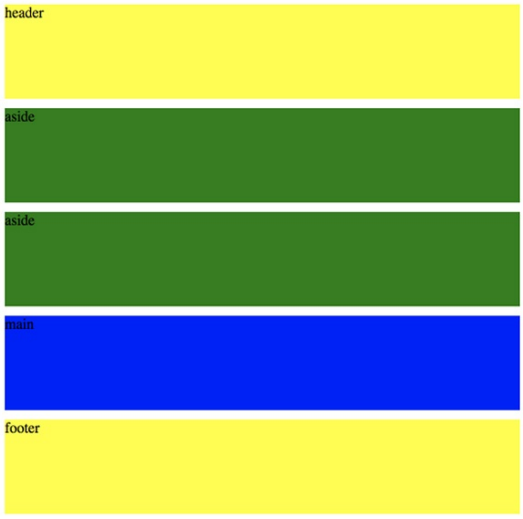
接着，需要把现有的 css 中 .left, .right, .middle 三个样式放到屏幕宽度大于 640px 的条件下启用，而在宽度小于 640px 的条件下，我们将启用另外三组样式，这三组是将现有的三列以行的方式来展示：

```
@media screen and (min-width: 640px) {
  .left {
    width: 200px;
    float: left;
  }
  .middle {
    margin: 20px 230px 20px 230px;
  }
  .right {
    width: 200px;
    float: right;
  }
}

@media screen and (max-width: 640px) {
  .middle {
    margin: auto;
  }
  .left, .right {
    width: auto;
```

```
    float: none;
  }
}
```

完工，我们一起看看效果。调整浏览器的右侧边界，逐渐缩小宽度，直到其低于 640px，你将看到如下效果：



总结思考

今天我们学习了一些网页交互设计的理念，知道了怎样通过渐进式增强来照顾到尽可能多的设备和浏览器，也通过例子实际动手了解了怎样实现网页的响应式布局，希望你有所收获。

现在，我来提两个问题吧：

- 在你的实际工作中，是否有考虑过不同能力的设备和浏览器的兼容适配问题，你又是怎样解决这样的问题呢？
- 给你这样几个 HTML 标签，你能否说出哪些是有语义的，哪些是无语义的呢？div、section、span、nav、summary、b。

好，今天就到这里，感谢你的阅读和思考，期待你的打卡！

扩展阅读

- 【基础】对于 CSS 不熟悉的程序员朋友，可以通过 MDN 上的 CSS 教程进行系统地学习。
- 文中提到了 SPA 环境下，对于浏览器的前进、后退功能，需要一些特别的技巧才能实现，其中一个技巧就是使用内嵌 iFrame，这个机制的原理在 Back Button Behavior on a Page With an iframe 这篇文章中有介绍（虽然这个机制本身当时给作者带来的是一个问题而不是一个解决方法），文中还附带了一个可尝试的小例子，另外的一个技巧我们将在第五章学到。
- 文中介绍了渐进增强和优雅降级的概念，在 Progressive Enhancement vs Graceful Degradation 这篇文章中，我们可以看到对它的进一步形象的阐释；而 CSS “渐进增强”在web制作中常见应用举例一文则举了几个 CSS 具体应用渐进增强的例子。
- 对于 HTML 语义的介绍，Semantic HTML 是一个非常好的用于介绍基础知识的胶片。

精选留言：

- 靠人品去赢 2019-10-18 11:46:39
  这个适配的问题，我看很多种方案都是先拿到宽度，然后定个数比如是100得到单位长度，然后不管多大分辨率，都可以标准化，老师有没有更好的适配方案，还有现在回头用原生的JS写页面好难受，不知道什么原因赋值不成功，不起作用很难受。

- leslie 2019-10-18 08:02:46
  我记得样式和加载不同：有时会听到开发PM抱怨用户用小众型的浏览器，甚至许多都是根本不曾听闻的，导致效果不一致缺东西。
  其实这块最大的坑在于有时不知缺啥且做了个性化导致：个性化其实有时导致的兼容性问题。