滑屏应用开发

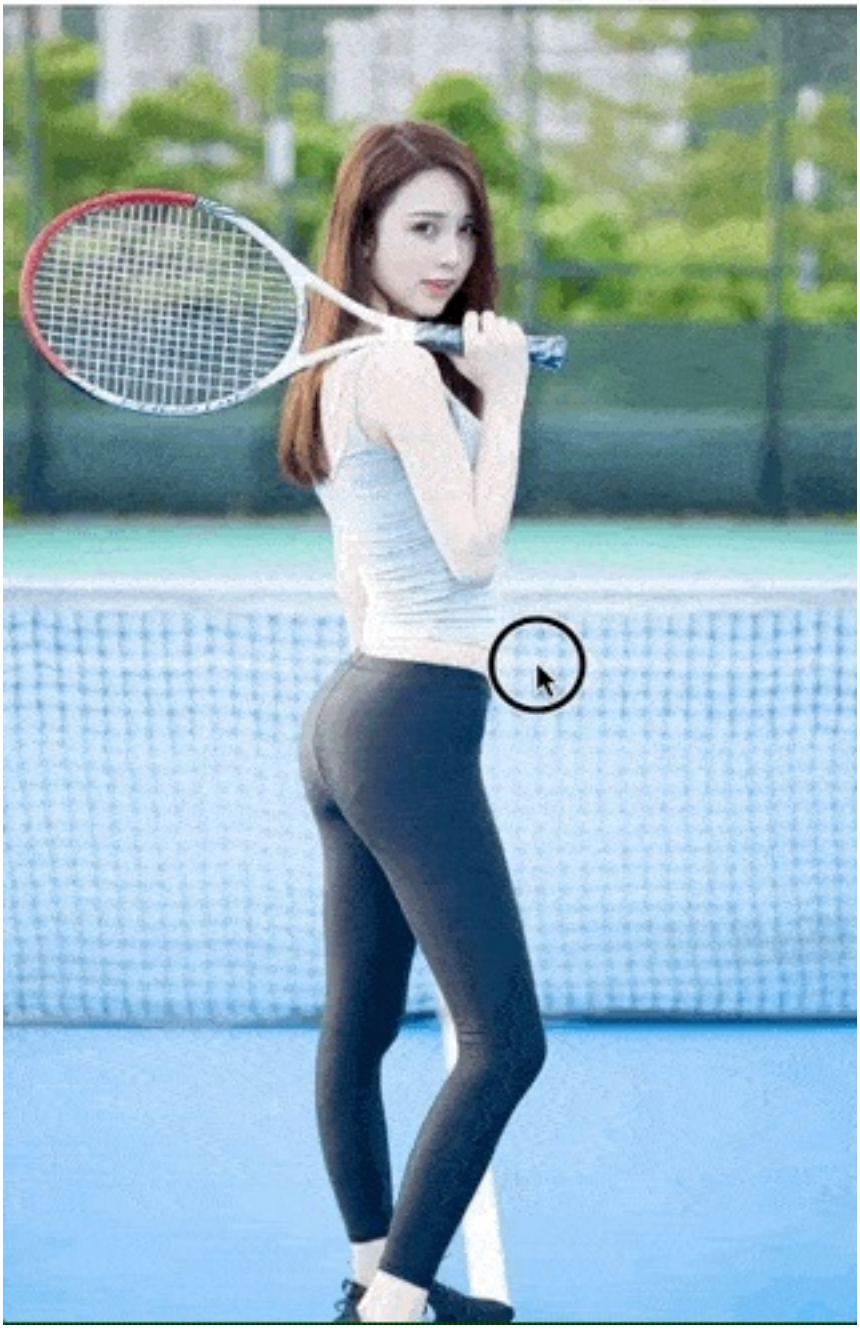
滑屏应用开发的能力可以定义为：

> 利用 JavaScript 和 CSS3 来实现单页面应用的滑屏效果，包括上下滑屏、左右滑屏，以及局部元素的滑动切换效果。

滑屏 H5 应用在国内是异常火爆的一种内容展示交互形式，被广泛用于各类线上营销活动场景中。

滑屏应用开发要求我们：

- 至少要掌握主流滑屏组件（如 Swiper）的具体用法
- 能不依赖已有组件实现简易的滑屏效果，了解滑屏的技术细节



上图为基础的滑屏页面效果示例。

善用利器

在平时工作过程中，考虑到项目的紧迫性和实现成本，我们大多数时候会使用业界已有滑屏组件，如：

- Swiper (https://github.com/nolimits4web/Swiper): Most modern mobile touch slider with hardware accelerated transitions.

其它更多滑屏组件的选择可查阅《awesome-javascript》(https://github.com/sorrycc/awesome-javascript#sliders)。

基于 Swiper 组件，只需数行代码即可创建一个基础的「滑屏应用」，以上下滑屏为例:

HTML (Jade)：约定的 HTML 结构

```
div.swiper-container
  div.swiper-wrapper
    div.swiper-slide
    div.swiper-slide
    div.swiper-slide
```

CSS (SCSS)：指定滑屏的尺寸为视窗大小

```
.swiper-container {
  width: 100vw;
  height: 100vh;
}
```

JavaScript：初始化为竖直方向上的滑屏应用

```
new Swiper('.swiper-container', {
  direction: 'vertical',
})
```

善用成熟的组件可以让我们免去从零实现滑屏效果的成本，并能保证开发速度和稳定性，解决 80% 的应用场景，但缺点是定制化成本较高，另外就是代码冗余，有时候你只用了它 20% 的功能，但却要加

载其 100% 的体积。当然，如果我们的滑屏应用场景不需要做定制化效果，也不用太在意那几十 KB 的体积的时候，利用业界成熟组件是首选方式。

知其所以然

善用利器，我们只做到了知其然。有些时候现有的滑屏组件不一定能满足个性化的业务需求，我们不得不去做二次开发，甚至需要根据个性化需求重新开发一款适用当前应用场景的滑屏组件，这就要求我们知其所以然。

作为示例，接下来我们探索下如何实现一个简单的 swiper。

我们将滑屏应用的实现分为两大部分：

- 判断用户的手势动作
- 根据手势动作执行相应滑屏过渡动画

为了更易理解，大家可以先在移动端体验以下三个例子，然后再阅读下面内容。

- 滑屏应用例子 swiper.js 版本 (http://jdc.jd.com/lab/swiper/swiper_plugin/)
- 滑屏应用例子 hammer.js 版本 (http://jdc.jd.com/lab/swiper/swiper_hammer/)
- 滑屏应用例子 无依赖版本 (http://jdc.jd.com/lab/swiper/swiper_pure/)

手势动作判断

手势动作判断是实现滑屏应用的核心逻辑。
对于上下滑屏应用，我们主要实现的手势动作有：瞬间的上下滑动和按住拖拽滑动。

瞬间的上下滑动除了要考虑滑动的始末位置，还要考虑时间间隔，即滑动速度。若满足一定的速度则代表用户是果断切换上下屏的动作，反之，则是犹豫保留在当前屏的动作。

我们看下核心代码，

```
var _this = this
var drag = false
var y0 = 0
var deltaY = 0
var time0 = 0

this.$swiper.on('touchstart', function (ev) {
  drag = _this.$swiper
  y0 = ev.touches[0].pageY
  time0 = new Date()
})

this.$swiper.on('touchend', function (ev) {
  var interval = new Date() - time0
  drag = false
  // 拖拽完成后，判断移动方向、移动速度和移动距离等
  // 若划动速度满足，则执行上划或下划过渡动画。
  // 若划动速度不满足，则判断是否划动距离是否大于阈值(如Swiper 容器的高度的一半)，若大于则执行上划或下划过渡画面，反之回到当前活跃块。
  _this.panEnd(deltaY, deltaY / interval)
})

this.$swiper.on('touchmove', function (ev) {
  if (drag) {
    deltaY = ev.touches[0].pageY - y0
    // 设置 .swiper-wrapper 按住拖拽的位移。
    _this.pan(deltaY)
  }
})
```

事实上，业界已经有许多很好用的判断手势动作的插件，如知名的 hammer.js (https://hammerjs.github.io/) 或 zepto (http://zeptojs.com/) 的 touch 模块。大家也可以通过阅读 hammer.js (https://hammerjs.github.io/) 的源码来进一步学习手势动作判断处理的逻辑。

滑屏过渡动画

过渡动画是让滑屏效果更自然的必要手段。

实现过渡动画的常见方式有两种：CSS3 或 JavaScript 动画，我们下面以 CSS3 动画作为示例。

这个过程我们需要关注的是什么时候触发动画，以及动画偏移的量为多少。

假设当前活跃块的索引为 activeIndex，将其与 swiper 容器的高度相乘并取反，可得到 .swiper-wrapper 的偏移量，然后设置适当的 CSS transition-duration 属性即可轻松实现过渡动画效果：

```
this.translate = -(this.activeIndex *
this.swiperHeight)
this.$swiperWrapper.css({
  'transform': 'translate3d(0, '+ this.translate
+'px, 0)',
  'transition': 'transform '+ 0.3 +'s'
})
```

若想保证一个过渡完成后，才能接收用户的下一个操作，则可以增加 animating 属性。动画过渡前就将其设置为 true，然后在 .swiper-wrapper 的 transitionend 事件触发时再将其设置为 false：

```
this.$swiperWrapper.on('transitionend',
function(ev) {
  if (ev.propertyName === 'transform') {
    _this.animating = false
  }
})
```

上述是最基本的位移过渡动画，你还可以依样画葫芦实现渐隐渐现、3D 翻转动画等。

源码分享

前面 3 个体验例子的源码放在 Coding.net 上，源码附注释，读者可以课后查阅。

- 滑屏应用例子源码 Swiper 版本 (https://coding.net/u/Jcc/p/swiper_plugin/git)
- 滑屏应用例子源码 hammer.js 版本 (https://coding.net/u/Jcc/p/swiper_hammer/git)
- 滑屏应用例子源码 无依赖版本 (https://coding.net/u/Jcc/p/swiper_pure/git)

一个实际案例

最后放一个典型的滑屏 H5 应用的实际案例 —— 京东 2018 校园招聘，供大家体验参考。

(京东 2018 校园招聘
(http://wqs.jd.com/promote/201707/2018campus/index.htr

值得一提的是，这个滑屏 H5 案例使用了凹凸实验室自研开源的 HTML5 构建工具 ELF (https://elf.aotu.io/) 进行构建，利用 ELF 提供的命令行工具，我们只需敲一个命令就可以搭建一个具有基础功能的滑屏应用，大大提升开发效率。

如果说 Swiper (https://github.com/nolimits4web/Swiper) 是开发滑屏应用的利器，那么 ELF (https://elf.aotu.io/) 则是开发 HTML5 应用的利器。

如果没有 Swiper，那么我们每次开发一个滑屏应用的时候，都要重新去写一遍滑屏手势的判断逻辑，还要重新写一遍滑屏的过渡动画。ELF 解决的是类似的问题，将搭建 HTML5 应用的重复过程自动化，将功能组件化、将交互形式模板化。它基于 Webpack 进行自动化构建，提供基础功能组件和脚手架案例模板；利用 ELF 提供的命令行工具 elf-cli，开发者可自由通过模板和组件的组合来快速定制开发各种 HTML5 场景应用。

实际上 ELF 本身基本上浓缩了前一个小节「响应式页面开发」和本小节「滑屏应用开发」的大部分内容，读者有兴趣可以移步 ELF 官网 (https://elf.aotu.io/) 进一步了解下 ELF 的功能特性。

性能贴士

在开发滑屏应用的时候，我们应该尽可能做到以下几点来保证页面的顺畅体验：

1. 做到延迟加载，避免浪费资源和并发加载资源数过高。
2. 做到预加载，预加载必要的资源，避免白屏。

在滑屏动画过渡期间，不要做繁重的任务，避免因占用资源过高而导致卡顿。

小结

本小节通过案例以及源码的方式，为大家介绍了如何利用业界优秀的滑屏开源组件「Swiper (https://github.com/nolimits4web/Swiper)」来快速开发滑屏应用，同时也解读了滑屏应用的关键处理逻辑：手势判断处理。

至此，我们学习了 「H5 开发」能力模型的前三种基础能力，在实际工作过程中，当视觉设计师妹子扔给我们一个设计稿（PSD）时，至少心里不会慌慌了。回想下第 2 小节「基础页面开发」，我们知道如何将一个设计稿转换成为高保真网页；如果这个设计稿是一个移动端的稿子，需要适配各种手机设备，那么第 3 小节「响应式页面开发」给我们指引了方向；而如果它是一个单页面的滑屏 H5 活动，第 4 小节「滑屏应用开发」里可以找到相应的开发思路。

但是，如果这个设计稿里面包含了许多动画效果的设计元素，需要我们做各种动效，该怎么办?