

# 网易微专业之《前端开发工程师》

## 学习笔记

开始时间：2016.03.03

## 《DOM 编程艺术》

### 实践篇

#### 组件操作

什么是组件？

“在用户界面开发领域，它是一种面对用户的、独立的可复用交互元素的封装”

组件 = **HTML**（结构）+ **js**(逻辑) + **CSS**(样式)

常用的组件包括：

- Mask:遮罩组件
- Datepicker:日期选择器组件
- Carousel:轮播组件
- Modal:模态弹窗组件
- Pager:翻页器组件
- Editor:富文本编辑器组件

#### 组件流程（**Common process**）

分析：交互意图以及需求

结构：HTML+CSS 实现静态结构

接口：定义公共接口

实现：从抽象到细节，实现功能接口、暴露事件

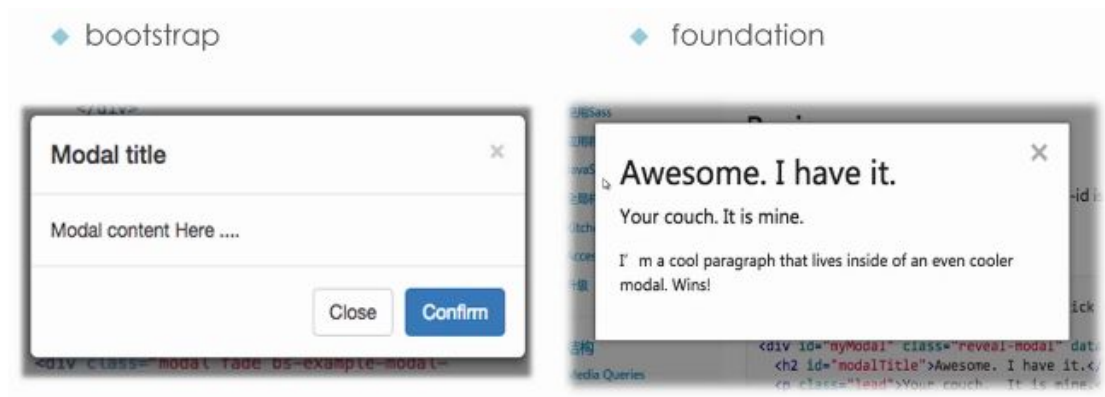
完善：便利接口、插件封装、重构等

#### 弹窗&模态组件（**Modal**）

模态是最常用的组件，它通过弹出一个高聚焦性的窗口来立刻捕获到当前用户的注意力。

Modal 举例：

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>



## Modal 范例:

最终视觉效果:



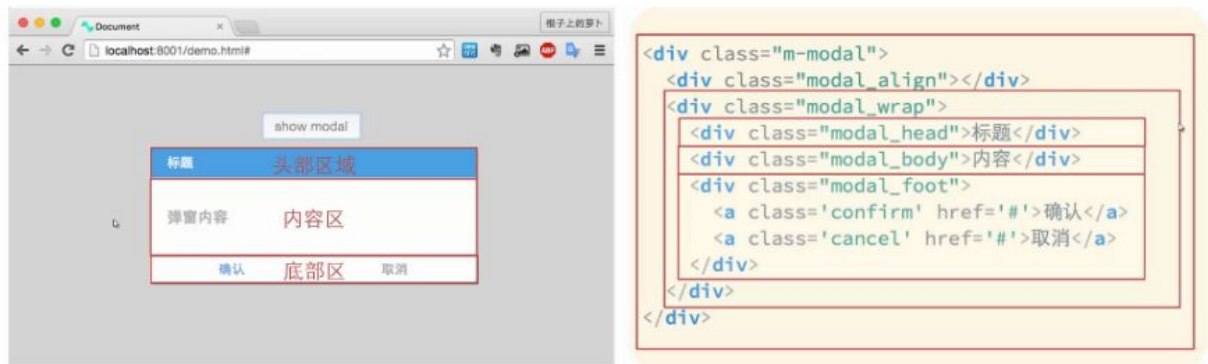
## 案例的实现流程:

### 1. 需求分解:

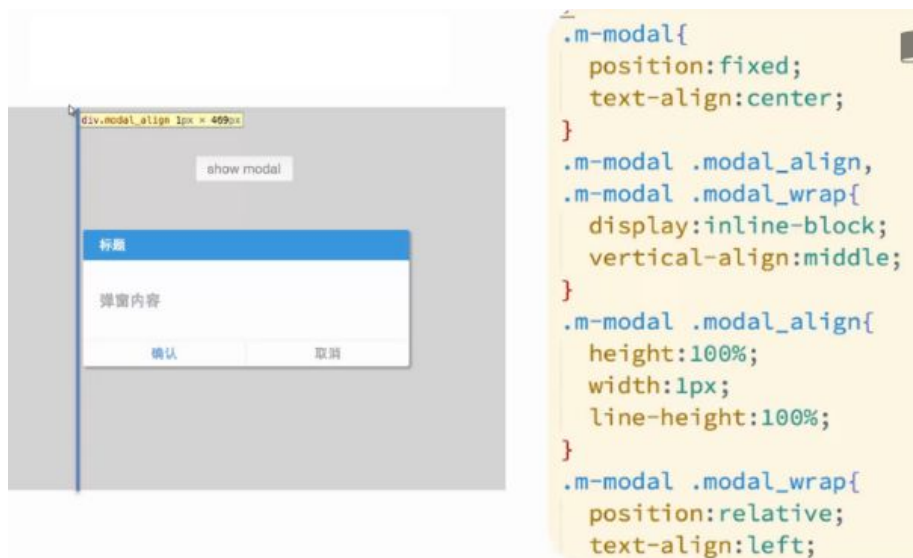
模态窗口垂直水平居中(宽高不固定)  
需要半透明的遮罩背景  
可自定义弹窗内容和标题  
提供确认和取消操作



### 2. 页面-结构分解



## 2.1 页面-绝对居中



## 3. 定义公共接口



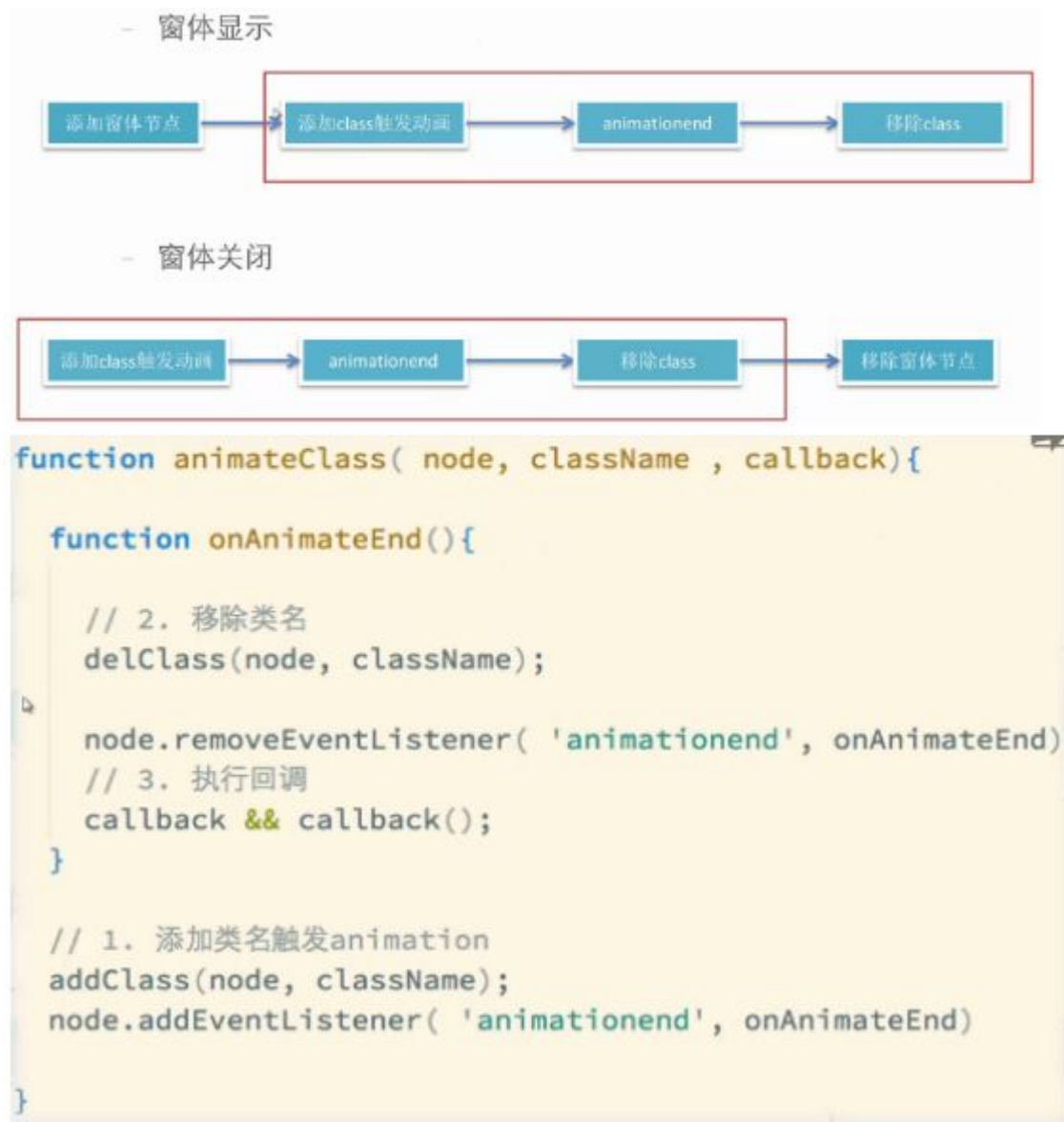
## 4. 实现思路：从抽象到细节

```
function Modal(){  
  
}  
  
Modal.prototype.show = function(){  
    // 显示逻辑  
}  
Modal.prototype.hide = function(){  
    // 隐藏逻辑  
}
```

## 5. 不足之处

- 没有过渡动画，体验不佳
- 缺乏组件事件支持
- 窗口在内容过高时会失效

### 5.1 动画流程：使用 css3 animation



## 5.2 使用事件 mixin

### 监听者模式：confirm 为例

又称为订阅发布模式，其主要作用是解耦。它可以将调用与被调用者解耦，并实现多个监听者对同一个对象的监听。

```
var modal = new Modal();

modal.on('confirm', function(){
  console.log('confirm')
})

_onConfirm: function(){
  // this.onConfirm();
  this.emit('confirm');
  this.hide();
},
```

实现

```
3 var emitter = {
4   // 注册事件
5   on: function(event, fn) {
12  },
13   // 解绑事件
14   off: function(event, fn) {
33  },
34   // 触发事件
35   emit: function(event){
44  }
45 }

// 使用混入Mixin的方式使得Slider具有事件发射器功能
extend(Modal.prototype, emitter);
```

## 要点总结:

### 1. 基于‘类’组织

```
function Modal(options){

  options = options || {};

  this.container = this._layout.cloneNode(true);
  this.body = this.container.querySelector('.modal_body');
  this.wrap = this.container.querySelector('.modal_wrap');

  // 将options 复制到 组件实例上
  extend(this, options);

  this._initEvent();

}

extend(Modal.prototype, {
  _layout: html2node(template),
```

合理使用Mixin

### 2. 结构复用

```
// 将HTML转换为节点
function html2node(str){
  var container = document.createElement('div');
  container.innerHTML = str;
  return container.children[0];
}

Modal.prototype._layout = html2node(template);

this.container = this._layout.cloneNode(true);
```

```
var template =
'<div class="m-modal">\
  <div class="modal_align"></div>\
  <div class="modal_wrap animated">\
    <div class="modal_head">标题</div>\
    <div class="modal_body">内容</div>\
    <div class="modal_foot">\
      <a class="confirm" href="#">确认</a>\
      <a class="cancel" href="#">取消</a>\
    </div>\
  </div>\
</div>';
```

## 本节案例涉及知识点回顾

- addEventListener, cloneNode, querySelector 等常用 API
- 绝对居中（垂直+水平）的一种解法

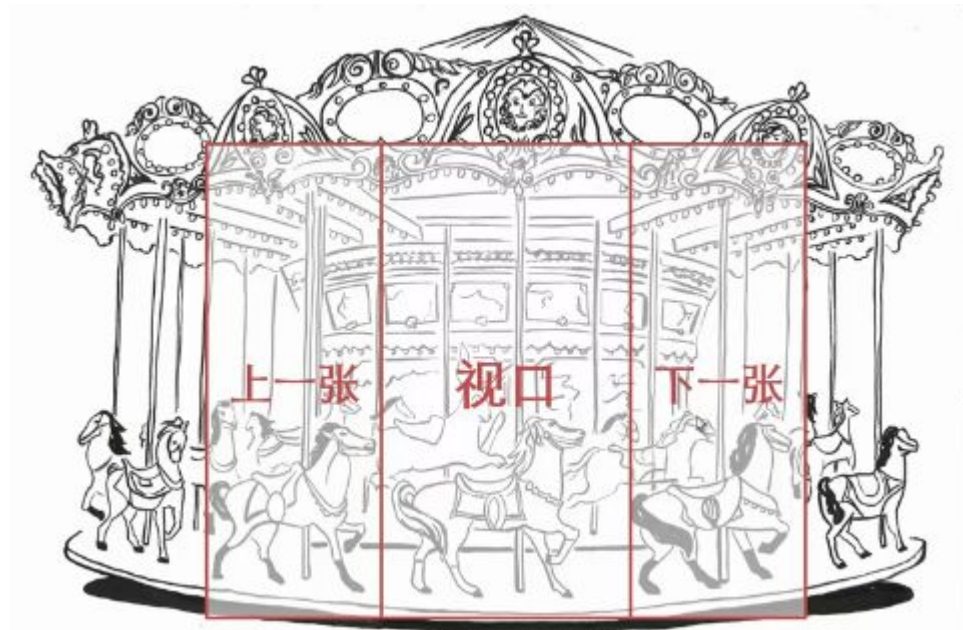
- 基于 `css3` 的动画结合方案
- 熟悉实现一个组件的一般流程
  - 分析需求
  - 静态结构
  - 接口设计
  - 代码实现
  - 完善细节



## 轮播组件（Carousel/Slider）

### 什么是轮播组件？

轮播组件可以实现在有限区域内，对多个图片（或内容）的循环播放展示，通常会用于广告、图片墙等场景。



### 案例效果图：



### 实现原理：

#### 1.需求分解

- 滚动内容垂直水平居中
- 滚动条目数不受限制
- 前后翻动，并支持拖拽
- 可直接定位

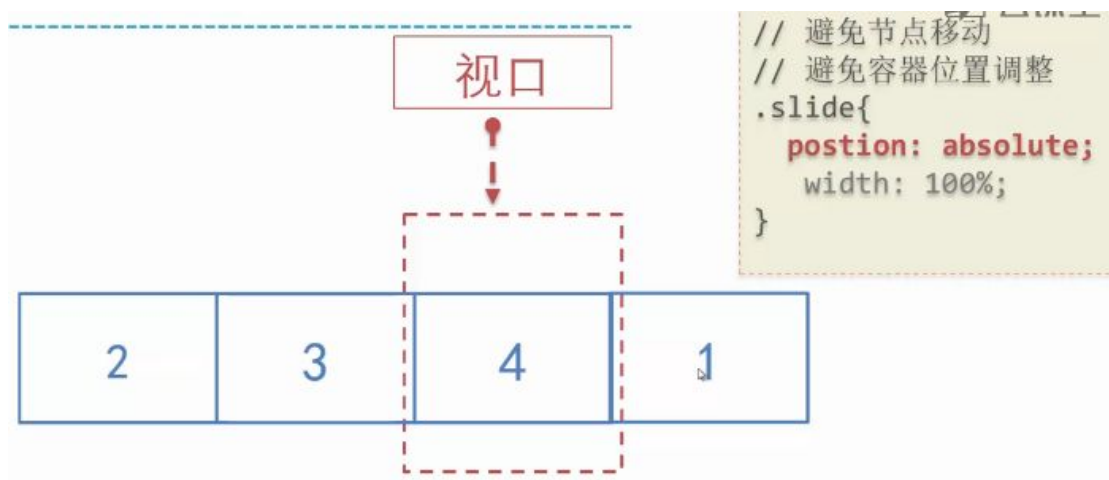


## 2.需求分解:

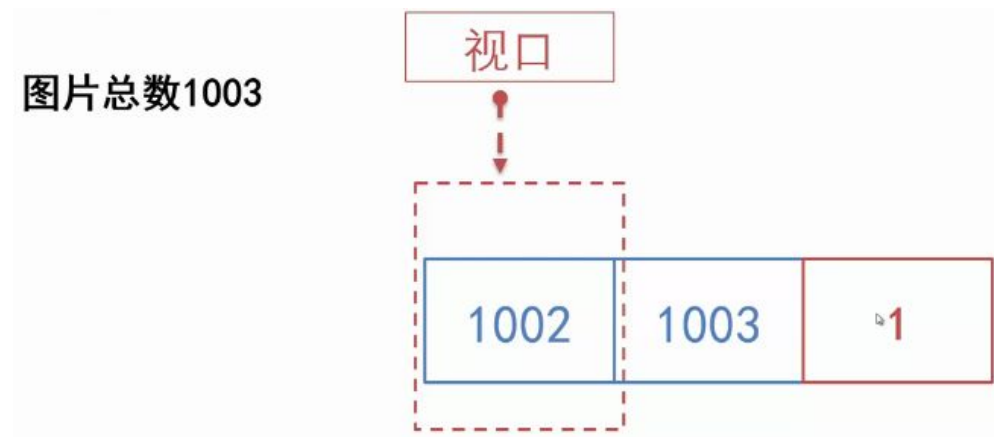
2.1 方案 1: 首尾过渡不满足要求，会出现快速回退效果。



2.2 方案 2: 当轮播图片数量很大时，会严重影响性能



2.3 最终方案: 无需关注图片数量，展示的节点只有 3 个，但对逻辑实现要求更高，



## 2.4 页面结构分解

轮播容器(传送带)

视口-- overflow:hidden

轮播图 position: absolute

```
<body style='overflow:hidden'>
  <div class="m-slider"
    style="transition-duration: 0.5s;
    transform: translateX(-3000%) translateZ(0px);">
    <div class="slide" style="left: 3100%;">
      
    </div>
    <div class="slide" style="left: 3200%;">
      
    </div>
    <div class="slide z-active" style="left: 3000%;">
      
    </div>
  </div>
</body>
```

滚动带

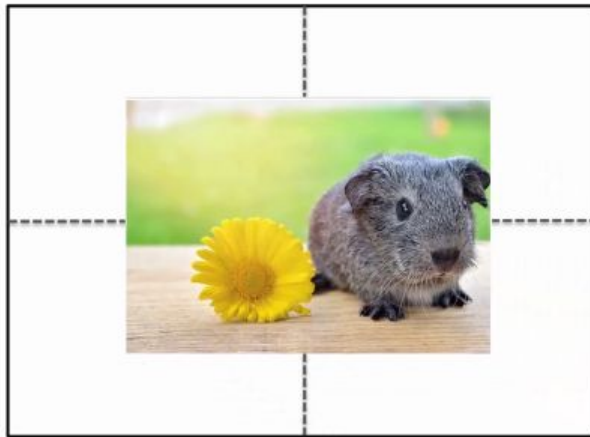
```
.m-slider{
  position: relative;
  transition-property: transform;
  transition-duration: 1s;
  transition-timing-function: ease-out;
}

.m-slider,
.m-slider .slide{
  height: 100%;
  width: 100%;
}

.m-slider .slide{
  position: absolute;
  top: 0;
  left: 0;
}
```

transform:translateX(-3000%) translateZ(0px)中 translateZ()作用:硬件加速

## 2.5 页面-绝对居中



```
.m-slider .slide img{  
  
    position: absolute;  
    max-width: 100%;  
  
    top: 50%;  
    left: 50%;  
  
    transform: translate(-50%, -50%);  
  
}
```

transform 是相对自身尺寸而言的，与父容器尺寸无关。

### 3. 公共 API 接口

#### 3. 公共API接口

##### ◆ 初始化轮播

##### ◆ API调用

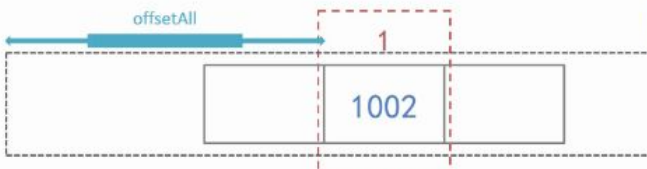
```
// 通过监听`nav`事件来完成额外逻辑  
slider.on('nav', function( ev ){ })  
  
// 3s 自动轮播  
setInterval(function(){  
    // 下一页  
    slider.next();  
}, 3000)  
  
// 直接跳到第二页  
slider.nav(2)
```

```
var slider = new Slider({  
    // 视口容器  
    container: document.body,  
    // 图片列表  
    images: [  
        "./imgs/pic01.jpg",  
        "./imgs/pic02.jpg",  
        "./imgs/pic03.jpg",  
        "./imgs/pic04.jpg",  
        "./imgs/pic05.jpg",  
        "./imgs/pic06.jpg"  
    ],  
  
    // 当前页  
    pageIndex: 2,  
  
    // 是否允许拖拽  
    drag: true  
});
```

### 4. 实现要点:

#### 4.1 数据定义

- `pageIndex[0~pageNum]`: 当前图片下标
- `slideIndex[0~2]`: slide下标
- `offsetAll`: 容器(.m-slider)的偏移下标



```
<div class="m-slider" style="transition-duration: 0.5s; transform: translateX(300%) translateZ(0px);">
  <div class="slide z-active" style="left: -300%;">
    
  </div>
  <div class="slide" style="left: -200%;">...</div>
  <div class="slide" style="left: -400%;">...</div>
</div>
```

- `pageIndex[0~pageNum]`: 当前图片下标
- `slideIndex[0~2]`: slide下标
- `offsetAll`: 容器(.m-slider)的偏移下标

```
<div class="m-slider" style="transition-duration: 0.5s; transform: translateX(300%) translateZ(0px);">
  <div class="slide z-active" style="left: -300%;">
    
  </div>
  <div class="slide" style="left: -200%;">...</div>
  <div class="slide" style="left: -400%;">...</div>
</div>
```

三个值可以唯一缺点 Slide 展现

## 4.2 流程简析



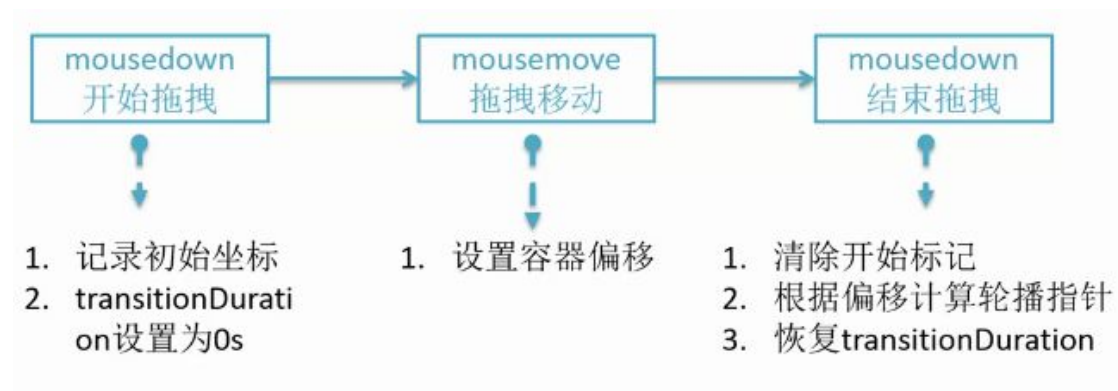
## 4.3 数据驱动的 UI 开发

- 将 UI 抽象为数据，是保证组件可测性的关键一步
- 更易维护，只需关注单一入口\_calsSlide

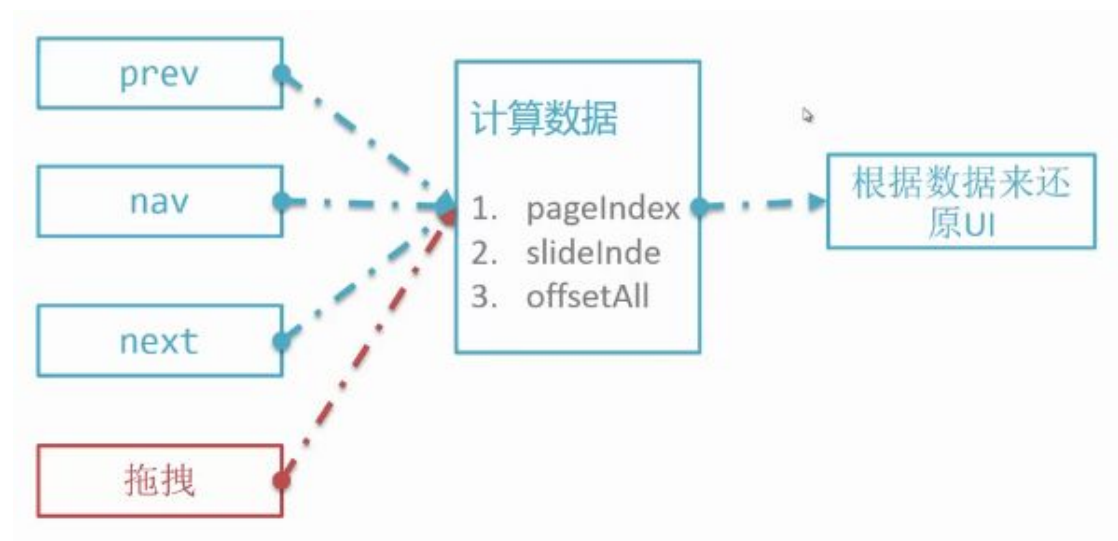
## 5.不足

- 需求的拖拽仍未实现
- 自动进行与手动切换的冲突未解决
- 如果持续调用 `next` 和 `prev` 将导致偏移非常大

## 5.1 推拽手势支持



## 5.2 加入推拽之后的流程



## 5.3 继承

### 3. 继承

- 继承重写\_onNav
- 精简基类\_onNav职责

```
function Slider2(opt){
  Slider.call(this, opt)
  this.pageNum = this.contents.length;
}

Slider2.prototype = Object.create( Slider.prototype );

Slider2.prototype._onNav = function(pageIndex, slideIndex){
  var slides = this.slides;
  var contents = this.contents;

  [-1, 0, 1].forEach(function(i){
    // ....other logic

    slide.innerHTML = contents[curPageIndex];
  }).bind(this))

Slider.prototype._onNav.apply(this, arguments)
}
```

## 本节涉及知识点

- 绝对居中（垂直+水平）的另一种解法
- 基于继承的组件扩展复用

本笔记由西风潇潇编写，欢迎浏览博客访问更多内容：<http://www.xifengxx.com>

- transform,transition 应用以及硬件加速
- 拖拽操作的一般思路
- 小试数据驱动的 UI 开发

## 组件流程回顾（**Common process**）

- 分析：交互意图以及需求
- 结构：HTML+CSS 实现静态结构
- 接口：需求的代码层抽象，指导实现
- 实现：从抽象到细节，实现功能接口、暴露事件
- 完善：便利接口、插件封装、重构等

具体的 DEMO,详见附件。