Table of Contents

概述

Rainbow 是基于 HTML5、CSS3 和 JavaScript 对 EAD 服务引擎提供的 Rest Service API 进行解析和渲染，动态生成 Web 用户交互界面的前端应用程序框架。本文对 Rainbow 前端框架进行详细说明，包含了开发 Rainbow 所使用的关键技术介绍，Rainbow 关键术语定义。分别从架构概述、核心框架选型、模块及程序运行流程、视图、组件、界面主题、国际化语言支持、扩展开发、生成工具等方面综合进行概要说明。本文旨在描述 Rainbow 的设计开发思想和运行原理及相关生态依赖，如果要涉及实质的扩展开发请参考《扩展开发指南》。

关键技术

HTML 5

HTML5 是 HTML 最新的修订版本，2014年10月由万维网联盟（W3C）完成标准制定。目标是取代1999年所制定的 HTML 4.01 和 XHTML 1.0 标准，以期能在互联网应用迅速发展的时候，使网络标准达到符合当代的网络需求。广义论及 HTML5 时，实际指的是包括 HTML、CSS 和 JavaScript 在内的一套技术组合。它希望能够减少网页浏览器对于需要插件的丰富性网络应用服务（Plug-in-Based Rich Internet Application，RIA），例如：AdobeFlash、Microsoft Silverlight 与 Oracle JavaFX 的需求，并且提供更多能有效加强网络应用的标准集。

CSS 3

CSS 即层叠样式表（Cascading StyleSheet）。 在网页制作时采用层叠样式表技术，可以有效地对页面的布局、字体、颜色、背景和其它效果实现更加精确的控制。 只要对相应的代码做一些简单的修改，就可以改变同一页面的不同部分，或者页数不同的网页的外观和格式。CSS3 是 CSS 技术的升级版本，CSS3 语言开发是朝着模块化发展的。以前的规范作为一个模块实在是太庞大而且比较复杂，所以，把它分解为一些小的模块，更多新的模块也被加入进来。这些模块包括： 盒子模型、列表模块、超链接方式 、语言模块 、背景和边框 、文字特效 、多栏布局等。

JavaScript

JavaScript 一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在 HTML（标准通用标记语言下的一个应用）网页上使用，用来给 HTML 网

页增加动态功能。

JSON

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于 ECMAScript 的一个子集。 JSON 采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C、C++、C#、Java、JavaScript、Perl、Python等）。这些特性使JSON成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成(一般用于提升网络传输速率)。

AMD

AMD 全称是 Asynchronous Module Definition，即异步模块加载机制。AMD 规范是 JavaScript 开发的一次重要尝试，它以简单而优雅的方式统一了 JavaScript 的模块定义和加载机制，并迅速得到很多框架的认可和采纳。这对开发人员来说是一个好消息，通过AMD我们降低了学习和使用各种框架的门槛，能够以一种统一的方式去定义和使用模块，提高开发效率，降低了应用维护成本。

Web Storage

Web Storage实际上由两部分组成：sessionStorage 与 localStorage。 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。 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。

MVVM

Rainbow 基于 API 模型动态渲染用户交互视图，通过动态渲染的数据表单对视图模型进行更新，利用 Backbone 实现服务端和客户端数据同步交互

术语定义

View

Rainbow 的核心业务模块抽象，分为视图模型（View Model）和视图（View），视图模型负责与服务端 API 进行数据同步，视图模型收到 API 返回的数据后根据视图类型定义选择相应类型的视图进行界面动态渲染。视图是一系列界面组件的容器组件，一般视图由

动作栏、过滤器栏、数据列表、状态栏、分页栏、设置栏等功能组件组成。各功能组件彼此独立，根据对视图模型的事件监听来响应用户操作。

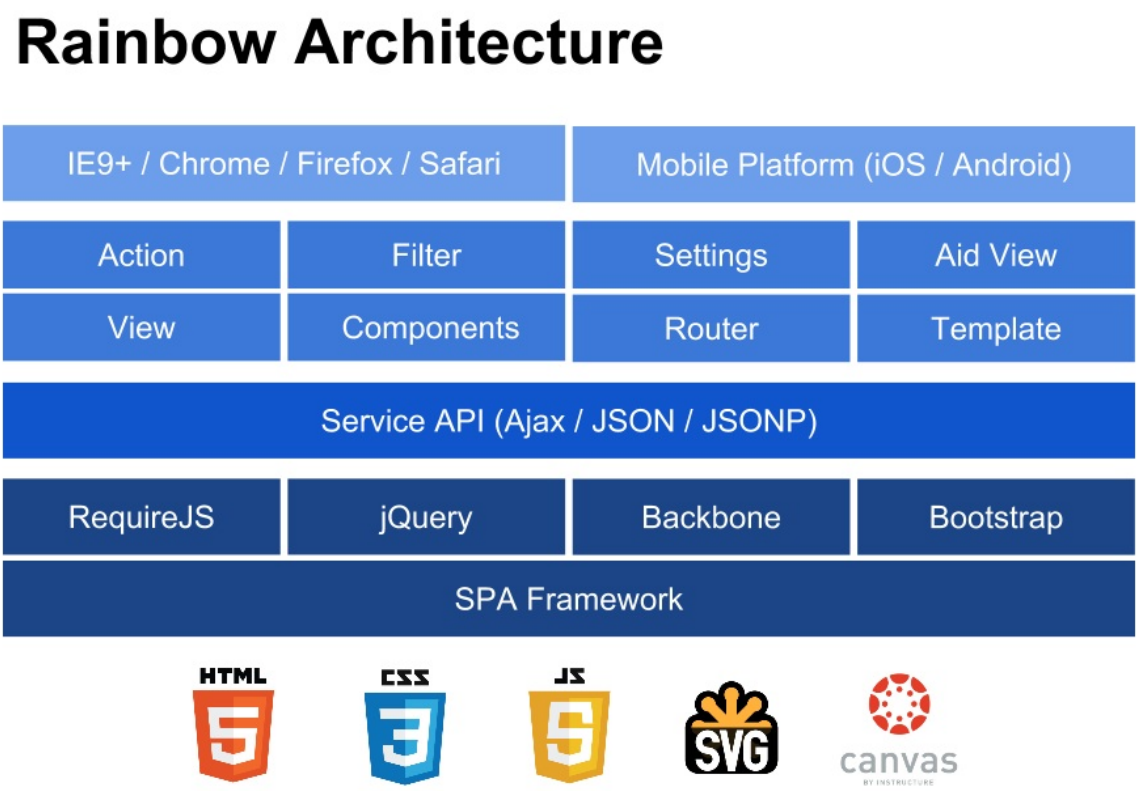
Action

Rainbow 的数据操作业务模块抽象，Action 作为视图的核心组件，根据视图模型的范式定义（Schema）动态渲染数据表单和进行数据校验，用户通过数据表单输入或变更数据时，自动同步更新视图模型实现数据绑定，最后通过 Restful 与服务端 API 交互完成数据操作。Action 是 Rainbow 中最复杂和业务多样式要求最多的模块，系统内置了创建（POST）、更新（PUT）、删除（DELETE）、单条数据处理（one）、多条数据处理（batch）、单文件上传（file）、多文件上传（upload）、下载（download）等常用动作组件，也允许通过扩展驱动的方式实现复杂数据操作动作的自定义。

参考资料

- 《平台架构说明》；
- 《扩展开发指南》；
- 《百度百科》；

Rainbow 架构



Framework

采用 SPA（单页 Web 应用框架）以 Ajax 为核心的服务请求模式，利用 RequireJS 基于 AMD 规范进行模块化开发和依赖管理，使用 jQuery 作为 DOM 操作类库，使用 Backbone 作为前端 MVC 框架，结合 Bootstrap 构建前端 UI 组件形成根据服务 API 动态渲染交互视图的 Web App 框架。

API

以 JSON / JSONP 的方式与服务端引擎 API 发起请求并异步响应返回数据接口。

浏览器

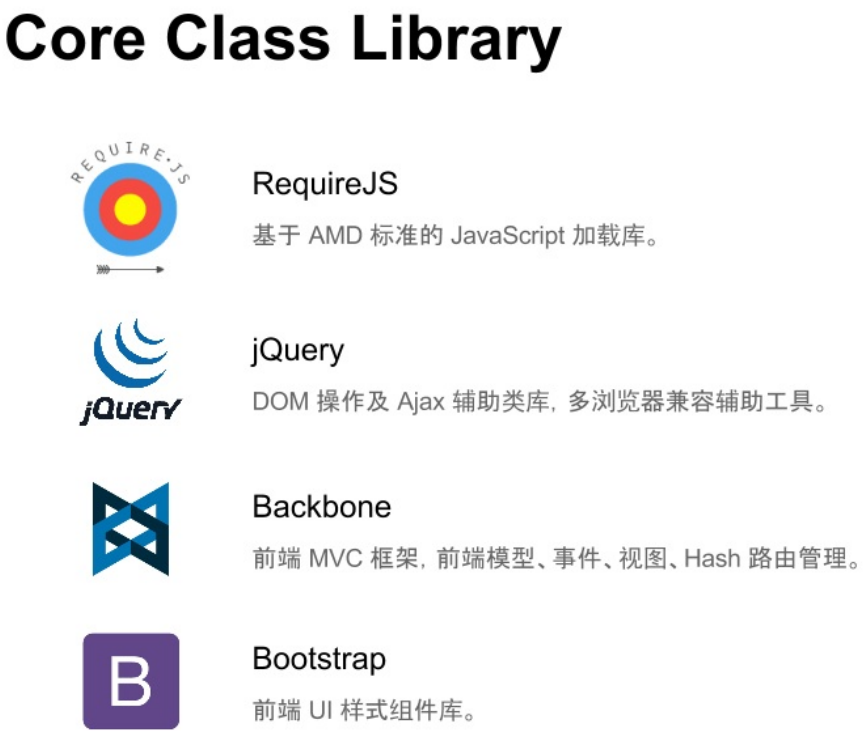
桌面（Desktop）

IE 10+ 、 Chrome、Firefox、Safari

移动端（Mobile）

iOS、Android、Windows Phone

Framework



RequireJS

基于 AMD 规范的 JavaScript 浏览器端模块依赖加载类库，支持全局配置定义，异步按需加载，模块命名空间定义，代码合并及混淆压缩，插件扩展。目前较为主流的 JavaScript 类库（例如：jQuery）都已经开始支持 AMD 规范，引入 RequireJS 能使前端工程更加良好的进行工程管理，方便与开源生态的结合。

参考地址：http://requirejs.org/

jQuery

jQuery是一个兼容多浏览器的 Javascript 库，核心理念是 Write less,Do more(写得更少,做得更多)。它是轻量级的 JS 库 ，它兼容 CSS3，还兼容各种浏览器（IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+），jQuery2.0 及后续版本将不再支持 IE6/7/8 浏览器。jQuery 使

用户能更方便地处理HTML（标准通用标记语言下的一个应用）、events、实现动画效果，并且方便地为网站提供 AJAX 交互。jQuery ，顾名思义，也就是 JavaScript 和查询（Query），即是辅助 JavaScript 开发的库。

参考地址：http://jquery.com/

Backbone

Backbone 为复杂 Javascript 应用程序提供模型(Models)、集合(Collections)、视图(Views)的结构。其中模型用于绑定键值数据和自定义事件；集合附有可枚举函数的丰富 API 视图可以声明事件处理函数，并通过 RESTful JSON 接口连接到应用程序。

参考地址：

- http://backbonejs.org/
- http://underscorejs.org/

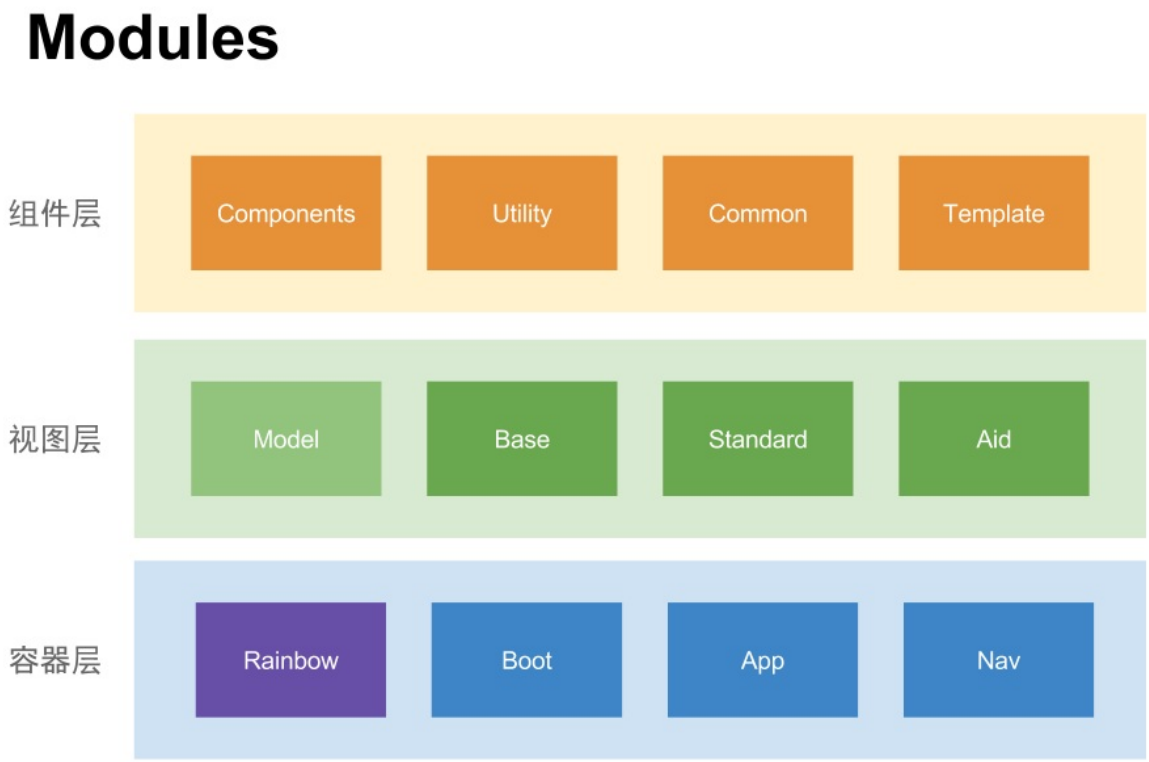
Bootstrap

遵循 HTML5 规范，以移动优先原则设计，支持响应式布局、网格系统、字体图标、基本 UI 组件、jQuery 扩展 UI 组件的 Web 开发通用 UI 框架。

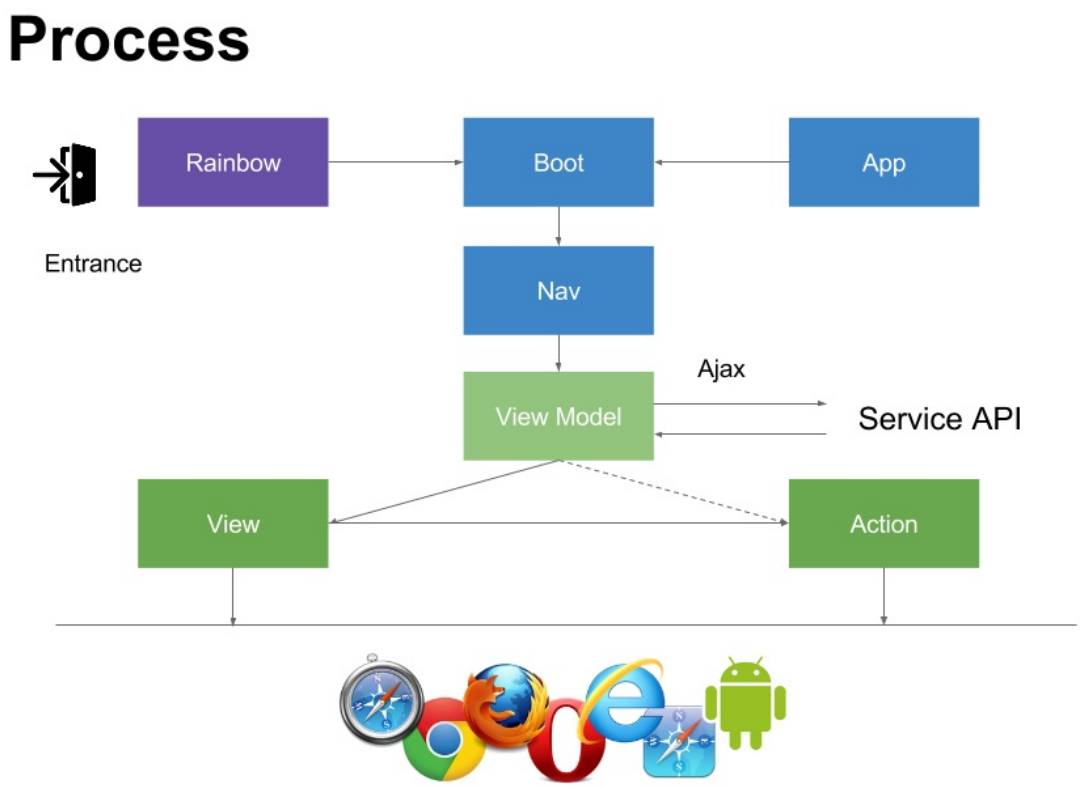
参考地址：http://getbootstrap.com/

模块及流程

分层模块架构



模块运行时流程



> 备注：如下描述的模块路径中不以“/”结尾的都为 .js 后缀的文件，为了便于抽象描述予以省略，例如：src/rainbow 在源码结构中为 src/rainbow.js 文件。以“/”结尾表示目录包。

1. 基础应用

包含框架入口、全局应用容器定义，全局模块加载及初始化。

Rainbow（框架入口）

模块路径：`src/rainbow`

加载框架全局类库（Backbone、jQuery、Underscore）和启动模块。

Boot（启动模块）

模块路径：`src/boot`

加载应用容器（App）、导航（Nav）和判断执行登录模块。

App（应用容器）

模块路径：`src/base/app`

定义全局方法和变量，初始化路由器和浏览器变化监听。在框架内部可以通过 `rainbow` 命名空间调用应用容器内的所有属性和方法。

Nav（导航模块）

模块路径：`src/base/nav`

资源导航，请求应用资源 API，渲染导航菜单并注册和监听资源路由，初始化应用布局容器。根据用户的前端交互事件，监听回调方法根据资源类型执行视图模型（View Model）初始化。

2. 视图

支持多实例的单元业务模块组件，通过导航模块初始化后对服务端 API 发起请求，服务端返回视图 API 数据后根据返回的视图定义和业务数据实时渲染当前业务模块的交互界面。

Model（视图模型）

模块路径：`src/view/model`

根据 URL 参数请求向服务端 API 请求并响应数据，注册并监听视图过滤器，根据过滤器模型的变化主动向服务端发起数据更新请求。服务端返回数据成功后根据视图定义数据实例化相应的视图，执行视图渲染后将视图追加到应用布局容器。

Base（基础抽象视图）

模块路径：`src/view/base`

视图组件基础抽象，实现了视图高度设置、显示、隐藏、销毁等通用或生命周期管理方法供普通视图扩展开发使用。

Standard（标准视图）

模块路径：`src/view/standard`

标准数据管理视图，根据模型视图数据定义组合动作栏、过滤器栏、设置栏、数据列表、状态栏、分页栏、辅助视图等基础交互组件动态构建并渲染单元业务模块的用户交互界面。

Aid（辅助视图）

模块路径：`src/view/aid`

结合视图动作辅助完成数据录入或局部数据内容渲染的辅助类视图抽象，通常在复杂视图动作场景以嵌套的方式加载到标准数据管理视图中使用。

3. 通用模块和基础组件

Components（组件包）

包路径：`src/components/`

通用交互界面基础组件包，一般通过组合多个基础组件构建应用视图，包括基础抽象、列表组件、表单组件、表单控制器组件、编辑器组件、页面小组件。

Utility（工具包）

包路径：`src/utility/`

通用工具模块，包括 Cookie、下钻、模式窗口、链接、全屏等辅助操作模块，可以在具体的业务场景中依赖复用。

Common（通用业务模块包）

包路径：`src/common/`

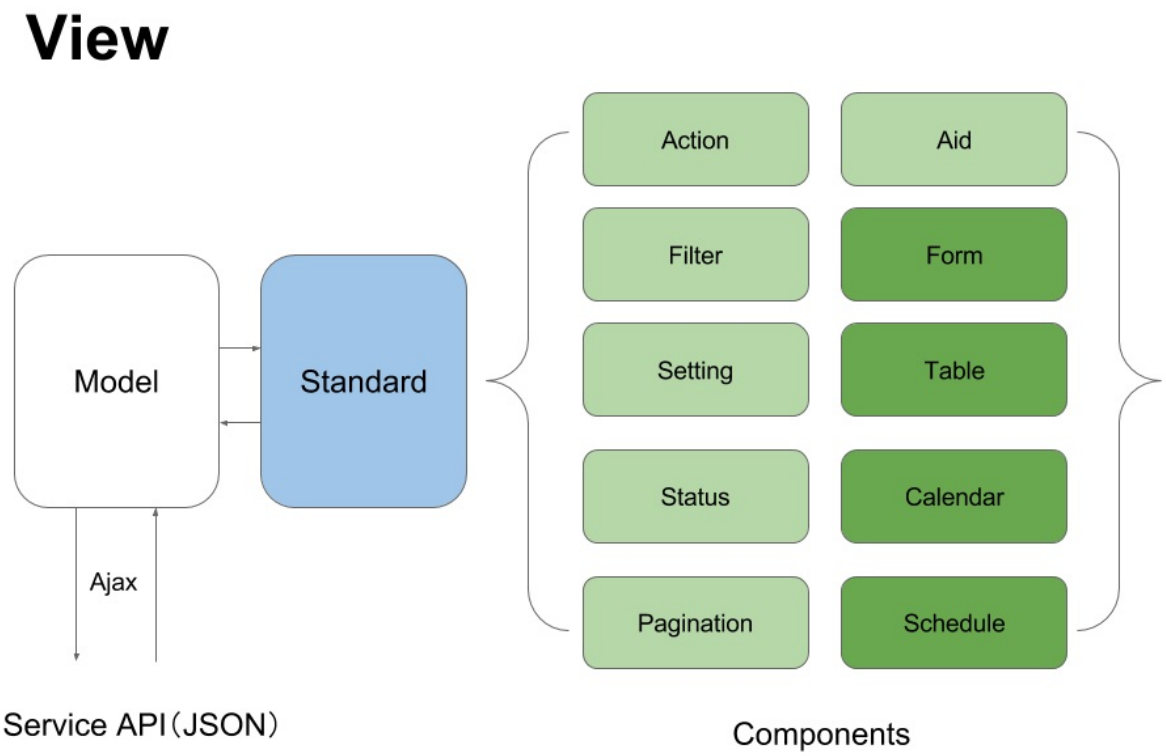
全局或通用业务模块包，包括登录、注销、通知等模块。

Theme（主题包）

包路径：`src/theme/`

应用主题皮肤模块包，目前内置 `classic` 和 `tech` 主题，可以通过扩展开发 CSS 和 Template 实现自定义主题扩展。

视图



生命周期

1. 导航模块监听路由事件实例化资源为视图模型；
2. 视图模型根据触发路由事件的资源对象组装 API 地址并向服务端发起 HTTP Rest 请求，视图模型根据服务端返回的模型数据实例化业务视图并初始化过滤器模型；
3. 视图依据视图模型数据组合动作栏、过滤器栏、设置栏、数据列表、状态栏、分页栏、辅助视图等基础交互组件动态构建并渲染单元业务模块的用户交互界面；
4. 视图组件依靠对视图模型的操作和事件监听通知或更新界面；

视图组件

内容组件

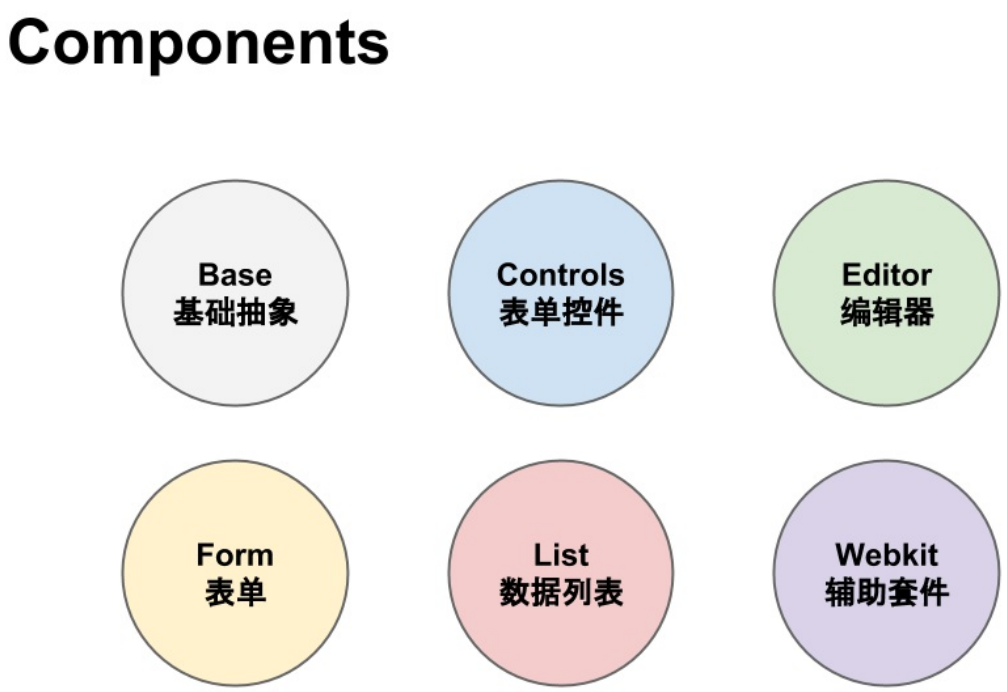
- **Table** / 表格列表;
- **Form** / 表单;

- **Calendar** / 日历列表;
- **Schedule** / 排程列表;

辅助组件

- **Action** / 动作栏;
- **Filter** / 过滤器栏;
- **Setting** / 设置栏;
- **Status** / 状态栏;
- **Pagination** / 分页栏;
- **Aid** / 辅助视图;

组件



Base（基础抽象）

通用基础组件抽象，例如 Cell、Row、List 等。

Controls（表单控件）

用户数据录入或选择的通用表单控件，包括文本框、文本域、隐藏域、普通下拉列表、日期选择、日期时间选择、搜索下拉选择器、多选选择器、月份、按钮组等控件组件。

Editor（编辑器）

批量数据或富文本编辑器组件，包括自定义视图和自定义内容编辑器。

Form（表单）

数据操作或查询控件集合组件，包括普通表单、文件上传表单、批量上传表单，后续会支持步骤表单。

List（列表）

列表型数据预览组件，包括表格、树形表格、日历、排程、网盘等形式的列表。

Webkit（通用页面组件）

页面增强型组件，包括树形组件。

主题

LESS

LESS 做为 CSS 的一种形式的扩展，它并没有减少 CSS 的功能，而是在现有的 CSS 语法之上，添加了许多其它的功能。LESS 通过变量、混合、嵌套规则、函数与计算实现 CSS 的模块化配置开发。

CSS

使用 LESS 可以分模块依赖设计不同风格的主题风格文件。

Template

使用模板标记语言实现 HTML 与数据的绑定。

```
<script type="text/template" id="tpl-example">
    <div>
        <p><%=name%></p>
        <div class="rb-actionbar"></div>
    </div>
</script>
```

源码包结构

```
├─assets
│  ├─bootstrap
│  ├─css
│  ├─fonts
│  └─img
├─dist
│  └─js
├─js
│  ├─action
│  ├─vendor
│  │  ├─bootstrap-daterangepicker
│  │  ├─bootstrap-datetimepicker
│  │  ├─highlight
│  │  ├─jquery
│  │  ├─kindeditor-4.1.9
│  │  ├─requirejs
│  │  ├─select2-4.0.0
│  │  ├─soundmanager
│  │  ├─sweetalert
│  │  └─webuploader-0.1.5
│  └─view
├─nls
│  ├─en
│  └─zh
├─src
│  ├─base
│  ├─common
│  ├─components
│  │  ├─base
│  │  ├─controls
│  │  ├─editor
│  │  ├─form
│  │  ├─list
│  │  └─webkit
│  ├─utility
│  └─view
│      └─kit
│         └─action
└─theme
    ├─classic
    └─tech
```

- assets： 资源包
  - bootstrap： Bootstrap

    - css： CSS 样式文件
    - fonts： 字体
    - img： 图片
- dist
    - js： 打包 JS 生成目录
- js
    - action： 扩展 Action 存放路径
    - vendor： 第三方组件目录
        - bootstrap-daterangepicker： 日期区间选择器
        - bootstrap-datetimepicker： 日期选择器
        - highlight： 代码高亮
        - jquery： jQuery
        - kindeditor-4.1.9： 富文本编辑器
        - requirejs： AMD 加载器
        - select2-4.0.0： 增强选择器表单控件
        - soundmanager： 声音播放
        - sweetalert： 消息提示
        - webuploader-0.1.5： Web 上传组件
    - view： 扩展视图存放路径
- nls： 语言包
    - en： 英文语言包目录
    - zh： 中文语言包目录
- src： 源码目录
    - base： 基础模块
    - common： 通用模块
    - components： 组件
        - base： 基础抽象
        - controls： 表单控件
        - editor： 富交互编辑器
        - form： 表单
        - list： 列表
        - webkit： 小组件
    - utility： 辅助函数
    - view： 视图
        - kit： 部件
            - action： 视图动作
- theme： 主题包存放路径
    - classic： 经典主题

- tech： 科技主题

国际化

I18N

利用 Requirejs I18N 插件根据本地语言设置自动加载 `nls` 目录下的不同语言包文件。

语言包

在 `nls` 目录下放置 JSON 格式的语言包。

扩展支持

AMD 模块定义

```
define(function(require, exports, module){
    var fn = function(){

    };

    return fn;
});
```

API 参考：http://requirejs.org/docs/api.html#cjsmodule

视图扩展

扩展脚本存放路径，`/js/view/extend-exsamp.js`，在开发者中心配置视图扩展为“extend-exsamp”会自动加载该扩展并应用执行。

动作扩展

扩展脚本存放路径，`/js/action/extend-exsamp.js`，在开发者中心配置动作事件为“e.extend-exsamp”会自动加载该扩展并应用执行。

生成工具

Node.js

Node.js是一个基于Chrome JavaScript运行时建立的平台， 用于方便地搭建响应速度快、易于扩展的网络应用。Node.js 使用事件驱动， 非阻塞I/O 模型而得以轻量和高效，非常适合在分布式设备上运行的数据密集型的实时应用。

登录官方主页，https://nodejs.org/ 选择合适的版本下载并安装。

LESS

LESS 做为 CSS 的一种形式的扩展，它并没有减少 CSS 的功能，而是在现有的 CSS 语法之上，添加了许多其它的功能。LESS 通过变量、混合、嵌套规则、函数与计算实现 CSS 的模块化配置开发。

工具安装

```
npm install lessc -g
```

打包命令（依赖于 Node 环境）

```
lessc style.less style.css
```

Requirejs

`r.js` 下载地址：http://requirejs.org/docs/release/2.2.0/r.js ，（另存为链接，下载存储到 Rainbow 根目录）。

Rainbow 根目录，执行打包命令（依赖于 Node 环境）。

```
node r.js -o _built.js
```

调试

生产模式

生产环境加载 `app.html`，通过 EAD 引擎通过动态替换应用参数完成初始化加载，该模式下加载合并和混淆后的 JS 文件。

开发模式

开发模式通过直接访问 `index.html` 进行调试访问，该模式下通过异步加载的方式按需加载 Rainbow 模块，方便代码调试。