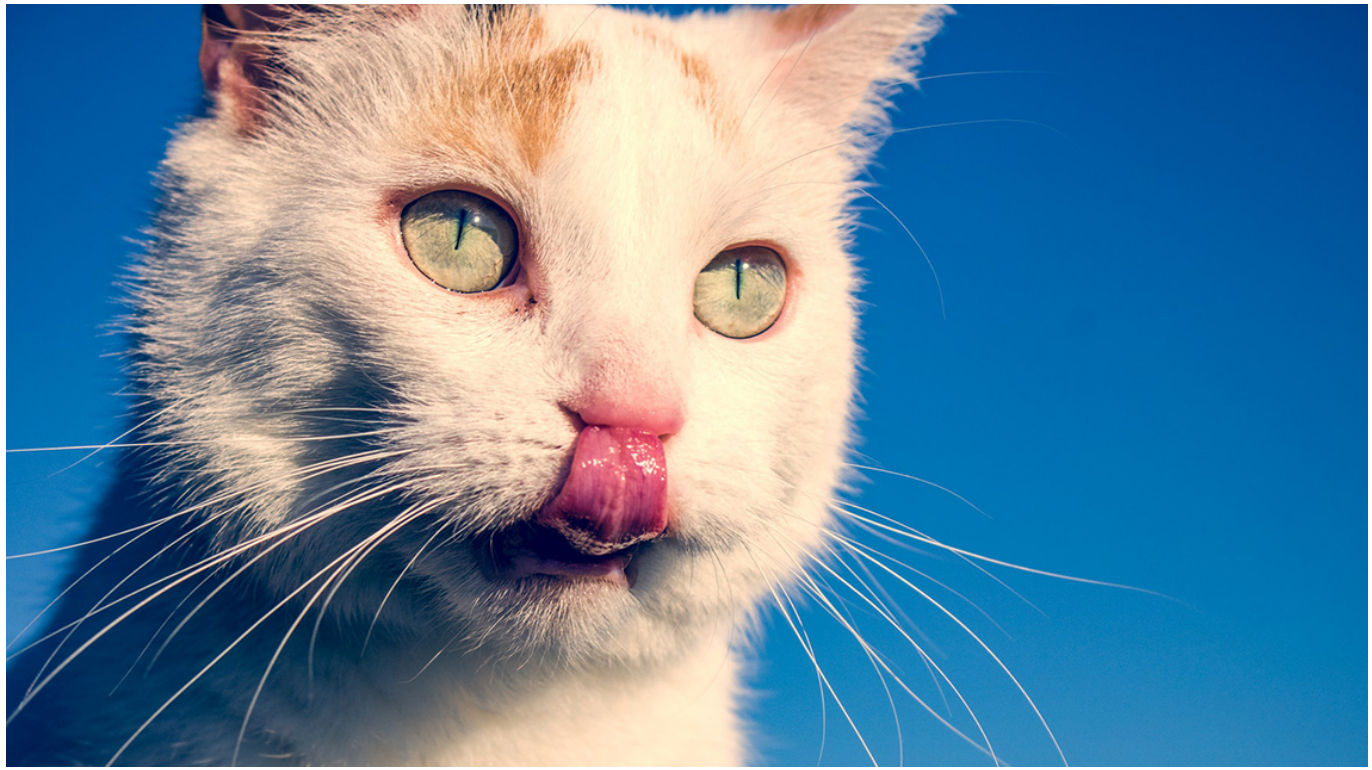
46 | 搭建系统：大量的低价值需求应该如何应对？ | 极客时间

开篇词 | 从今天起，重新理解前端
01 | 明确你的前端学习路线与方法
02 | 列一份前端知识架构图
03 | HTML语义：div和span不是够用了吗？
04 | HTML语义：如何运用语义类标签来呈现Wiki网页？
05 | JavaScript类型：关于类型，有哪些你不知道的细节？
06 | JavaScript对象：面向对象还是基于对象？
07 | JavaScript对象：我们真的需要模拟类吗？
08 | JavaScript对象：你知道全部的对象分类吗？
新年彩蛋 | 2019，有哪些前端技术值得关注？
09 | CSS语法：除了属性和选择器，你还需要知道这些带@的规则
10 | 浏览器：一个浏览器是如何工作的？（阶段一）
11 | 浏览器：一个浏览器是如何工作的？（阶段二）
12 | 浏览器：一个浏览器是如何工作的（阶段三）
13 | 浏览器：一个浏览器是如何工作的？（阶段四）
14 | 浏览器：一个浏览器是如何工作的？（阶段五）
15 | HTML元信息类标签：你知道head里一共能写哪几种标签吗？
16 | JavaScript执行（一）：Promise里的代码为什么比setTimeout先执行？
17 | JavaScript执行（二）：闭包和执行上下文到底是怎么回事？
18 | JavaScript执行（三）：你知道现在有多少种函数吗？
19 | JavaScript执行（四）：try里面放return，finally还会执行吗？
20 | CSS 选择器：如何选中svg里的a元素？
21 | CSS选择器：伪元素是怎么回事儿？
22 | 浏览器DOM：你知道HTML的节点有哪几种吗？
23 | HTML链接：除了a标签，还有哪些标签叫链接？
24 | CSS排版：从毕升开始，我们就开始用正常流了
25 | 浏览器CSSOM：如何获取一个元素的准确位置

26 | JavaScript词法：为什么12.toString会报错?
27 |（小实验）理解编译原理：一个四则运算的解释器
28 | JavaScript语法（预备篇）：到底要不要写分号呢?
用户故事 | 那些你与“重学前端”的不解之缘
29 | JavaScript语法（一）：在script标签写export为什么会抛错?
期中答疑 | name(){}与name: function() {}，两种写法有什么区别吗?
30 | JavaScript语法（二）：你知道哪些JavaScript语句?
31 | JavaScript语法（三）：什么是表达式语句?
32 | JavaScript语法（四）：新加入的**运算符，哪里有些不一样呢?
33 | HTML替换型元素：为什么link一个CSS要用href，而引入js要用src呢?
34 | HTML小实验：用代码分析HTML标准
35 | CSS Flex排版：为什么垂直居中这么难?
36 | 浏览器事件：为什么会有捕获过程和冒泡过程?
37 | 浏览器API（小实验）：动手整理全部API
38 | CSS动画与交互：为什么动画要用贝塞尔曲线这么奇怪的东西?
答疑加餐 | 学了这么多前端的“小众”知识，到底对我有什么帮助?
39 | HTML语言：DTD到底是什么?
40 | CSS渲染：CSS是如何绘制颜色的?
41 | CSS小实验：动手做，用代码挖掘CSS属性
加餐 | 前端与图形学
加餐 | 前端交互基础设施的建设
42 | HTML·ARIA：可访问性是只给盲人用的特性么?
43 | 性能：前端的性能到底对业务数据有多大的影响?
44 | 工具链：什么样的工具链才能提升团队效率?
45 | 持续集成：几十个前端一起工作，如何保证工作质量?
winter 2019-05-16



你好，我是 winter。

不知道你在工作中有没有遇到过这样的事情：一个运营找过来说，有一个紧急又简单的临时活动页面要做，希望打断现有的产品开发节奏临时插入。

这类页面技术难度不高，业务上通常属于“紧急不重要”的事情。

这些需求技术上没挑战，线上存在时间短，上线时间紧又没有任何调整空间，它们往往会成为前端团队里人人都不喜欢的“垃圾需求”，谁要是接了这种需求，就只能自认倒霉。

但是，这些真的是垃圾需求吗？换个视角来看，我认为它反而是宝藏。

所谓工程师，就是为了解决这些问题而存在的岗位，我们从工程的视角来看，“大量紧急不重要的页面”，才是真正的需求，现在需求有了，我们就应该按照工程的方式，定目标、设计方案、做实施、拿结果来解决问题。这就是我们今天要讲的搭建系统。

搭建系统的目标

搭建系统的目标是解决大量的简单页面生产问题。衡量这个目标的指标应该是生产页面的数量，这部分非常的明确，你如果要做搭建系统，可以根据业务的体量和服务的范围来决定具体的指标要求。

搭建系统的设计

搭建系统设计大概有几种流派，这里我介绍几种常见的搭建系统的设计：

第一种，是模板化搭建，由前端工程师生产页面模板，再由运营提供数据来完成页面，可以用以下公式来理解：

> 模板 + 数据 = 页面

模板化搭建是一种简单的思路，它的优点是整个系统实现简单。

第二种思路是，模块化搭建，由前端工程师生产模块，由运营把模块和数据组织成页面。

第三种思路，是数据驱动界面，这是一种比较新的思路，即数据中包含了展现自身所需要的模块相关的信息，本身决定了界面。

但是不论何种流派，都可以认为是数据、模块、模板、页面几种实体的相互作用，下面我就来详细讲解一下这几样实体。

数据

数据是用于展现界面所需要的信息。

我们按照数据用途，可以分成界面配置数据和内容数据。

- 界面配置数据：决定了页面上颜色、尺寸、位置、图片、文字等展现形式的数据，通常是以页面为单位的配置。
- 内容数据：页面要展示的信息，如电商活动页面的商品信息、文章的文字信息等。

按照数据来源，我们又可以分成运营人员手工填写的数据和来自 API 产生的数据。

- 运营手工填写固定数据：运营人员依靠自己的专业技能决定的数据，可能包含线下招商信息、商品选品、文章等。
- 来自 API 的数据：

- 固定数据，由服务端逻辑到指定存储处获取的数据；
- 用户相关数据，由算法系统或者服务端逻辑，根据用户信息或者用户喜好推荐的数据。

搭建系统本身是个产品，我们针对数据这个实体，要设计增、删、改、查的能力，根据我们以上的分析，搭建系统的数据部分有两个难点。

第一个难点是数据的手工编辑能力，现在一般的数据都会采用 JSON 格式，JSON 格式中有数字、字符串、数组、对象、布尔等数据类型，我们需要根据数据的格式定义为每一种类型设计编辑器。

但是仅仅是基本类型还不够，我们实际开发中，还需要跟实际业务结合来设计编辑器，下面，我就把我在之前的工作中设计的数据编辑器列一下。

- 整数：整数编辑器，可用 HTML 原生输入框`<input type=number min=1 max=100/>`实现。
- 数字：数字编辑器，可用`<input type=number min=1.0 max=100.0/>`实现
- 字符串：字符串编辑器，可用`<input />`实现。
- URL：URL 编辑器，可用`<input />`配合格式校验。
- 图片：图片编辑器，需要自研图片上传功能。
- 固定字段对象：对象和字段编辑器，可用多个`<input />`和`<label>`实现。
- 布尔型：开关，可用`<select>`或者自研组件实现。
- 自由字段对象：需要自研 KV 输入组件。
- 数组：需要自研列表组件实现。
- 对象数组：需要自研表格组件或者列表组件实现。
- 矩形区域：需要自研区域选择组件。

这里要注意 JSON 是一个级联的格式，所以对象、数组中很可能需要插入各种不同的数据类型的编辑器，这部分技术上有一定挑战。此外，实践中，对象数组很多时候都来自 Excel 数据，Excel 导入也是非常重要的。

第二个难点则是跟服务端 API 的对接，对于服务端系统统一性较好的公司，这不是什么难事，对服务端系统比较奔放的公司，如果服务端 API 调用方式不统一，就非常麻烦了。这一块只能根据实际情况见招拆招，我这里没办法详细介绍，

模板

模板可以简单得理解成挖了许多坑的页面，它一般是由前端工程师来生产的一种实体。与数据之间的连接是数据的格式，对 JSON 格式来说，JSON Schema 是社区接受度较高的一个方案。

最简单的模板可以用字符串模板来设计，复杂一点的模板则可以由 JavaScript 进行渲染，通过约定全局变量名称或者约定调用函数入口做到把数据传递给模板，你可以根据实际需求复杂程度选择合适的方案。

需要注意，在产品设计上，模板可不是“增、删、改、查”那么简单，考虑到实际工程需要，模板必须是版本化的，也就是说，前端每发布一个模板，都需要永久性存储一条记录，并且产品设计上必须保持可以回滚，这样，一旦线上发现问题，可以迅速回滚到一个可工作的版本，有效降低不可用时长。

此外，模板设计还有批量更新的需求，一些运营活动可能包含数百个页面，它们使用同一套模板，产品设计上必须要注意提供批量更新机制。

模块

模块跟模板非常相似，但是从产品的角度，模块是可组合的。跟模板相似的部分如数据连接、版本化发布、批量更新等，这里就不再赘述。

模块化搭建有额外的技术难点，就是可拖拽的模块编辑器，移动端搭建布局相对简单，可以通过简单的自上而下布局和拖拽改变位置来实现。

桌面的模块拖拽比较复杂，一般都会采用一些变通的思路简化设计，如提供几种固定的布局模板，提供布局容器，或者采用纯绝对定位布局。

在一些产品设计中，会先用模块拼成模板，再指定数据源，这种模式中的”模块“，我们认为是一种开发模板的技术方案，跟我们此处讲的产品上的模块概念不同。因为在我们的认知中，模板应该是由前端工程师产生的，具有复用性的一种实体。

页面

不论是模板搭建还是模块搭建，我们的最终生产的目标都是页面。页面同样需要版本化发布，便于回滚。

页面部分实现的难点是跟发布系统的结合，在我们前面讲的所有产品实体中，模板、模块、数据都是存储在搭建系统本身的，但是页面不一样，页面必须要提供线上服务，所以页面是要发布到线上生产环境的。

如我们上一课讲的，假设前端持续集成系统有校验规则，页面也必须经过这个过程。

在我之前的工作中，是通过自建静态 Web 服务器 +CDN 回源的方式来支撑搭建系统的线上应用的。

因为服务器上只发布静态内容，并且有 CDN 挡住用户流量，所以只需要少量几台线上机器即可。

搭建系统的实施

在我工作的实践中，搭建系统的实施可以说是所有系统中最容易的了，对多数公司来说搭建系统是一种刚性需求，只要完成了产品开发，立刻会有大量的用户。

所以只要正确识别了需求，搭建系统的推行几乎完全不需要担心。

搭建系统的监控

作为一个工具型技术产品，搭建系统同样会产生大量有价值的数据，搭建系统的用户访问和生产页面数量是衡量自身的重要指标。

总结

本课我为你讲解了搭建系统，搭建系统是为了应对大量简单页面的生产需求而设计的一种工具型产品，它的目标非常明确，就是快速生产大量的页面。

方案上，它重点和难点在于几个产品实体的设计，数据部分重点在于编辑器和跟服务端 API 的对接，模板部分则主要是版本化和数据的格式定义，模块除了模板的重点，还有拖拽系统，最终产生的页面主要的难点是跟生产环境的对接。

搭建系统的实施主要是把产品在做出来，一般来讲推广是非常自然的事情，最后，搭建系统产生的数据监控关键的指标是用户访问数和生产页面数。

本课的思考问题是，请你分析一下你们公司是否有搭建系统的需求，尝试用本课的知识来设计或者改进一下你们的搭建系统。

bd2star

-
  2019-05-16
-

  2019-05-16