Sea.js 与
前端技术平台梦

2013.6.1

关于我

= {

姓名: **王保平**

工作: 支付宝 · 杭州

微博: 玉伯也叫射雕

}

大纲

一、前端模块化简史

二、Sea.js 的发展与现状

三、前端技术平台

一、前端模块化简史

模块系统的基本问题

- 模块是什么?
- 模块之间如何交互?

命名空间时代

```
var org = {};
org.CoolSite = {};
org.CoolSite.Utils = {};

org.CoolSite.Utils.each = function (arr) {
  // 实现代码
};

org.CoolSite.Utils.log = function (str) {
  // 实现代码
};
```

向 YUI 致敬

```
YUI().use("io-base", "mod-a", "mod-b", function(Y) {

  // Y.on 是 mod-a 添加的、还是 mod-b 添加的?
  // 如果都往 Y 上添加 on 方法，会出现什么问题?
  Y.on(...);

  // io-base 在 Y 上添加了 io 方法。
  // 是怎么知道的呢? 查文档? 为什么不是 "Y.IO" ?
  var request = Y.io(...);

});
```

Modules/1.x (CommonJS)

increment.js

```
var add = require('math').add;
exports.increment = function(val) {
    return add(val, 1);
};
```

http://wiki.commonjs.org/wiki/Modules/1.1.1

AMD (Asynchronous Module Definition)

```
define(["alpha"], function (alpha) {
    return {
      verb: function(){
        return alpha.verb() + 2;
      }
    };
});
```

https://github.com/amdjs/amdjs-api/wiki/AMD

Modules/2.0 (CommonJS)

Sample Module: increment.js

```
module.declare(['math'], function(require, exports, module) {

var add = require('math').add;

exports.increment = function(val) {

return add(val, 1);

};

})
```

http://www.page.ca/~wes/CommonJS/modules-2.0-7/

CMD (Common Module Definition)

```
// 所有模块都通过 define 来定义
define(function(require, exports, module) {

  // 通过 require 引入依赖
  var $ = require('jquery');
  var Spinning = require('./spinning');

  // 通过 exports 对外提供接口
  exports.doSomething = ...

  // 或者通过 module.exports 提供整个接口
  module.exports = ...

});
```

前端模块系统

- 模块是一段 JavaScript 代码，具有统一的书写格式。
- 模块之间通过基本交互规则，能彼此引用、协同工作。

模块定义规范

Module Definition Specification

分支演化

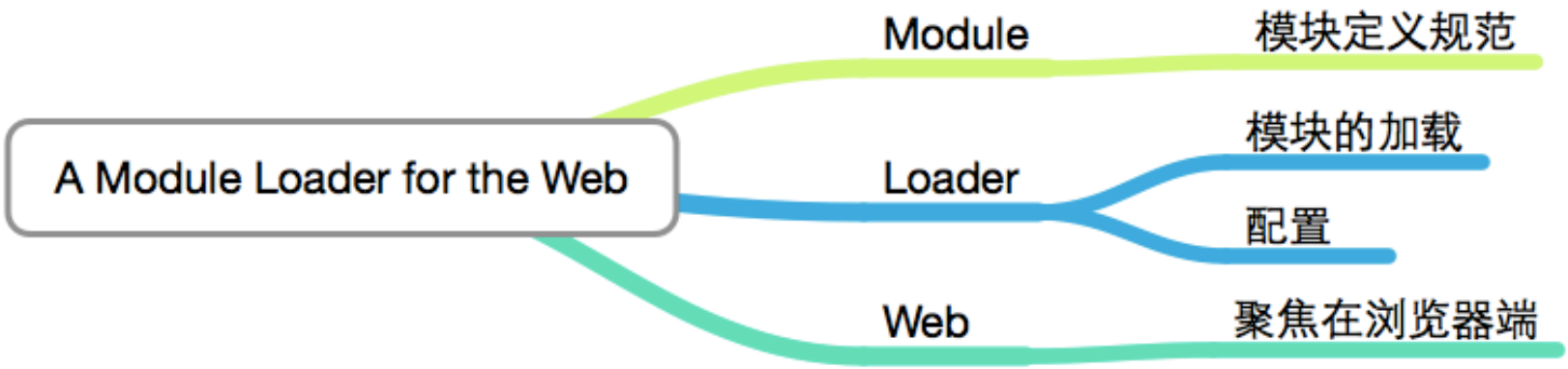
规范	服务端	浏览器端
Modules/1.x	Node.js	component
Modules/Async	r.js	RequireJS
Modules/Wrappings	Sea.js	Sea.js

如何选择模块化方案

- 团队成员的协作方式
- 产品的复杂度
- 工具、CDN 等基础建设

二、Sea.js 的发展与现状

Sea.js 是什么



http://seajs.org/

CMD 规范

```
// 通过 define 定义一个模块
define(function(require, exports, module) {
    // 通过 require 引入模块
    var $ = require('jquery');

    // 通过 exports 向外提供接口
    exports.doSomething = function() {
    };
});
```

模块的加载

```
// 通过 use 方法加载初始模块
seajs.use(["a", "b"], function(a, b) {
  // 模块的依赖会自动加载好
  // 理论上不再需要全局变量
})
```

Sea.js 的设计理念

① 保持简单

② 职责清晰

③ 性能优先

④ 适度完备

保持简单

- API 要简单
- 内部实现也要简单

https://github.com/seajs/seajs/issues/266

https://github.com/seajs/seajs/tree/master/src

职责清晰

- Web 端的模块加载器
- SoC（关注度分离）

性能优先

- 模块数量与依赖往往比想象复杂
- Transport 规范与构建工具

https://github.com/seajs/seajs/issues/226

tests/speed/thousand-modules/

适度完备

- 考虑开发时的便利性
- 可扩展机制
- 不断完善的测试用例

https://github.com/seajs/seajs/issues/264

tests/runner.html

https://travis-ci.org/seajs/seajs

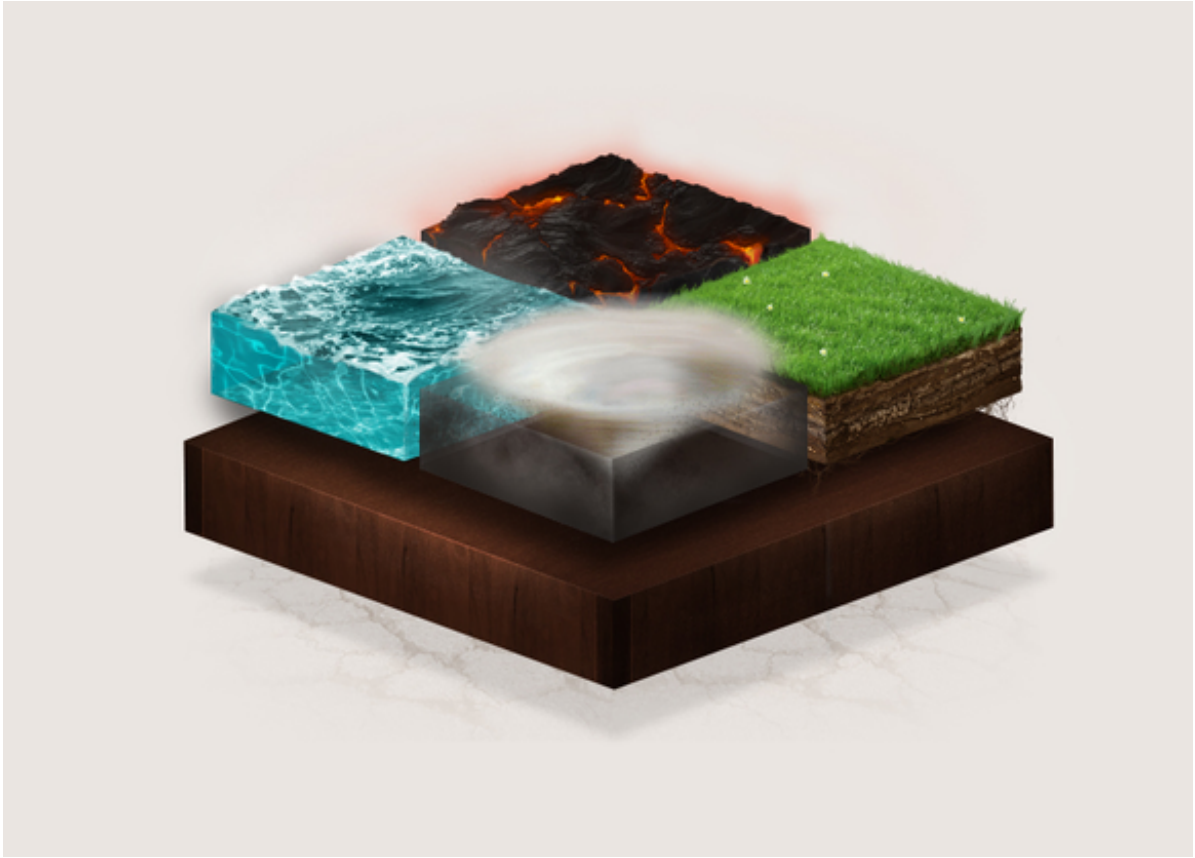
Sea.js 现状

纯开源方式运作

Sea.js 2.1.0 即将发布

https://github.com/seajs/seajs/issues/755

支付宝



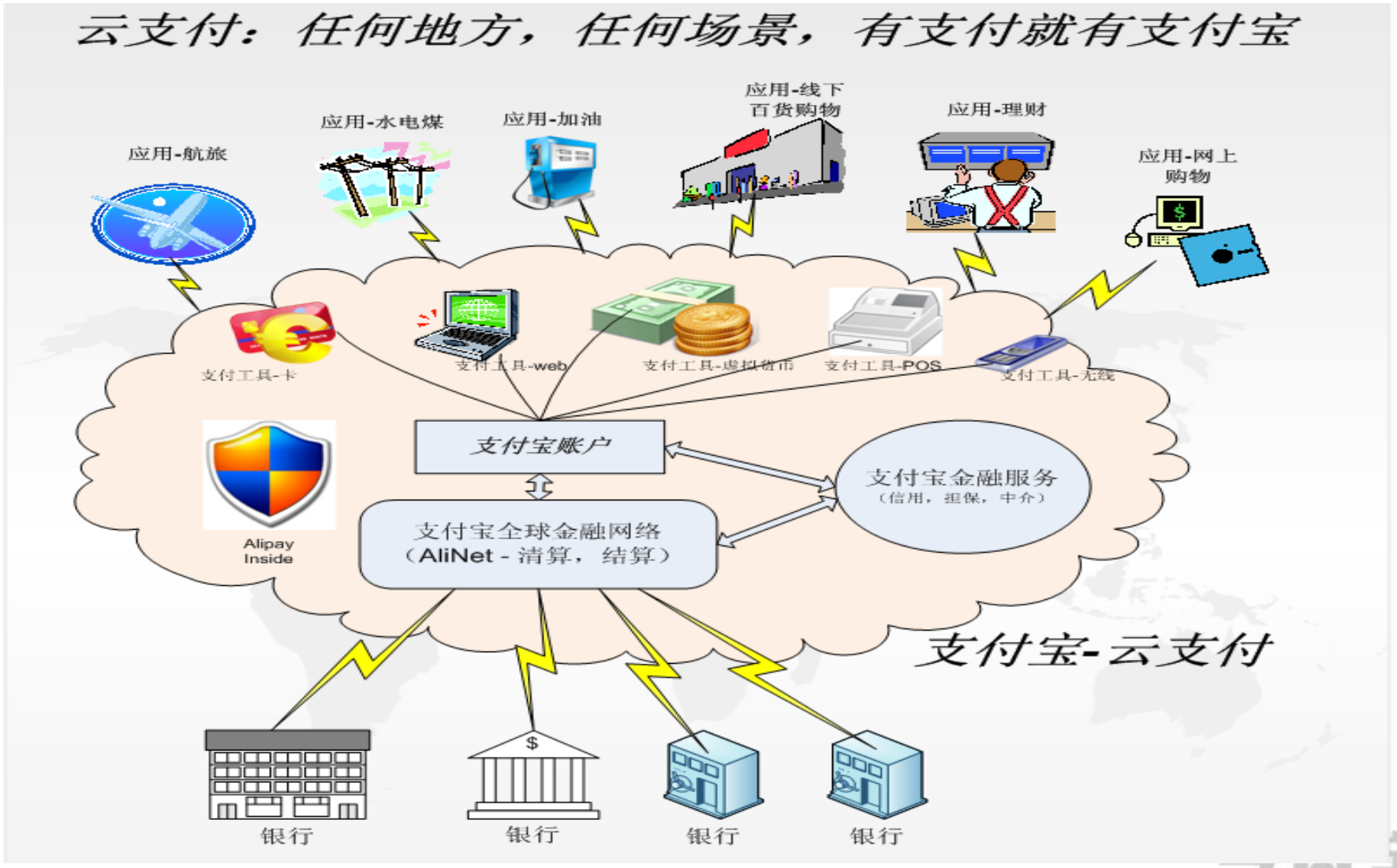
三、前端技术平台

支付宝

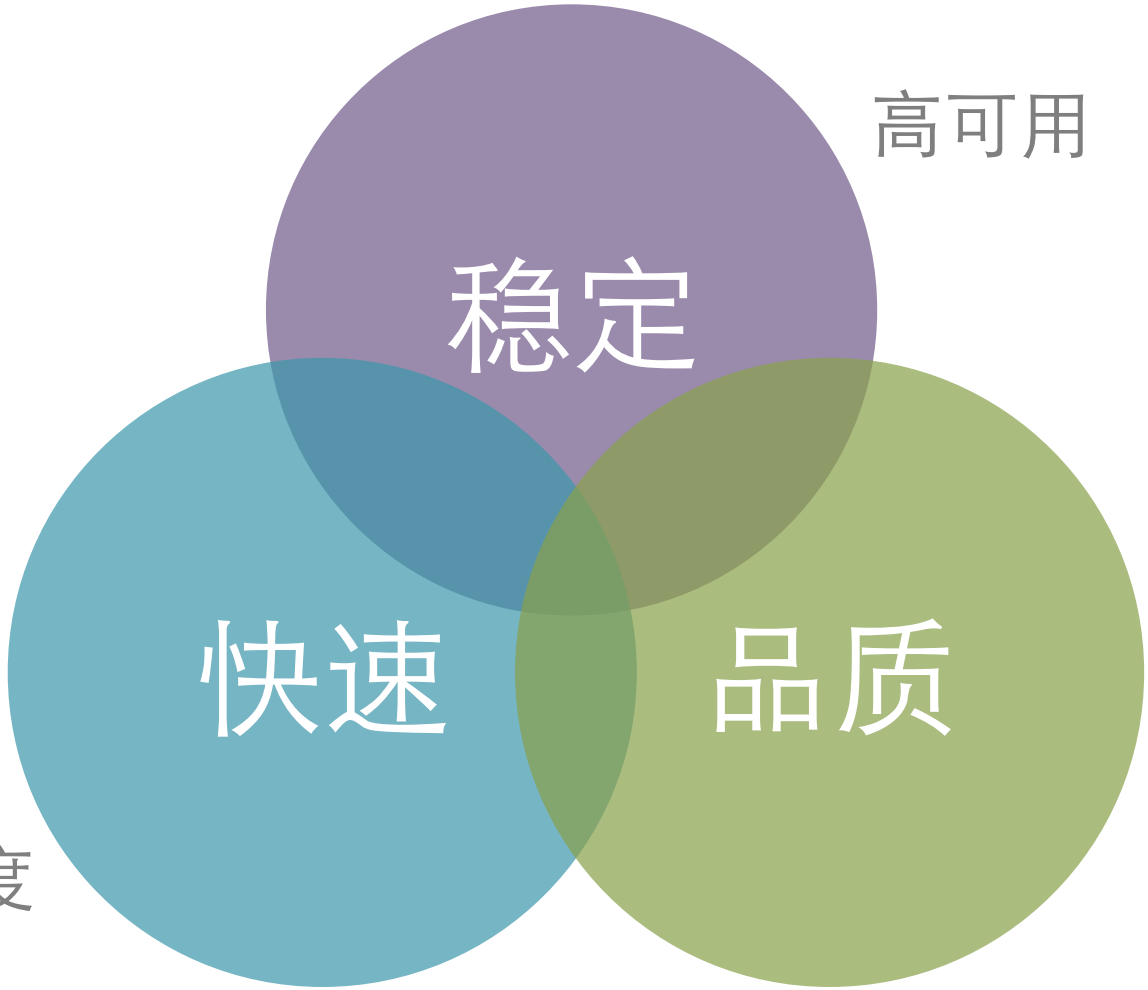
互联网

金融

业务愿景


云支付：任何地方，任何场景，有支付就有支付宝
应用-航旅
应用-水电煤
应用-加油
应用-线下
百货购物
应用-理财
应用-网上
购物
支付工具-卡
支付工具-web
支付工具-虚拟货币
支付工具-POS
支付工具-无线
Alipay
Inside
支付宝账户
支付宝金融服务
（信用，担保，中介）
支付宝全球金融网络
（AliNet - 清算，结算）
支付宝-云支付
银行
银行
银行
银行

对技术的要求



高可用

质量
性能
创新

研发速度

前端的职责

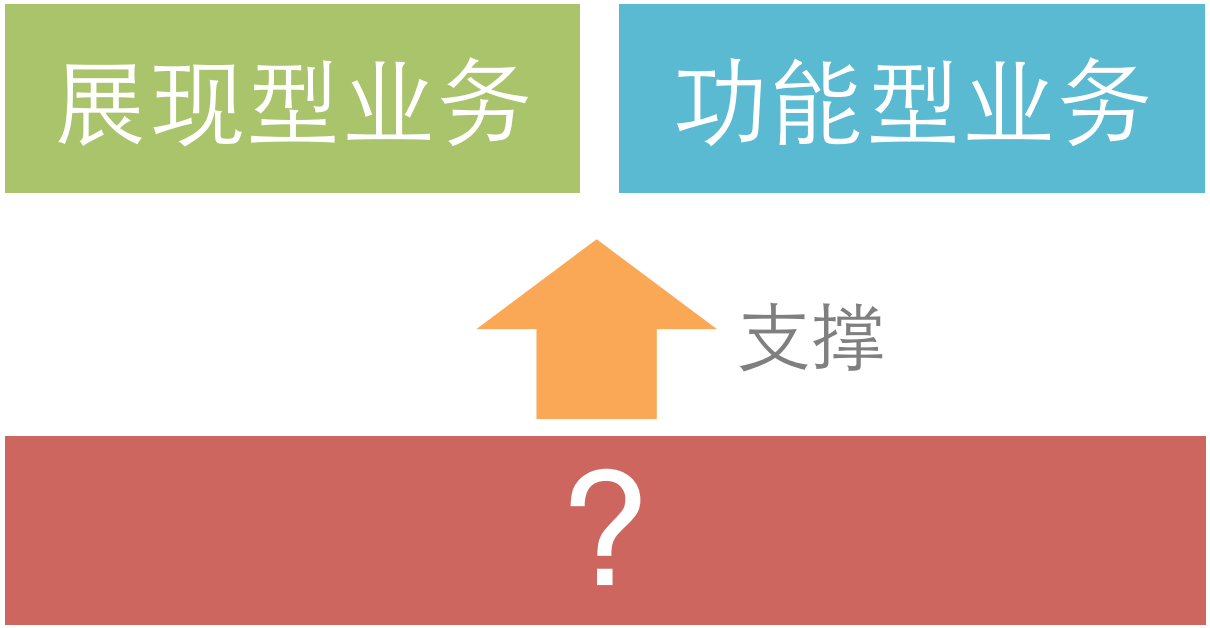
① 实现界面交互

② 提升用户体验

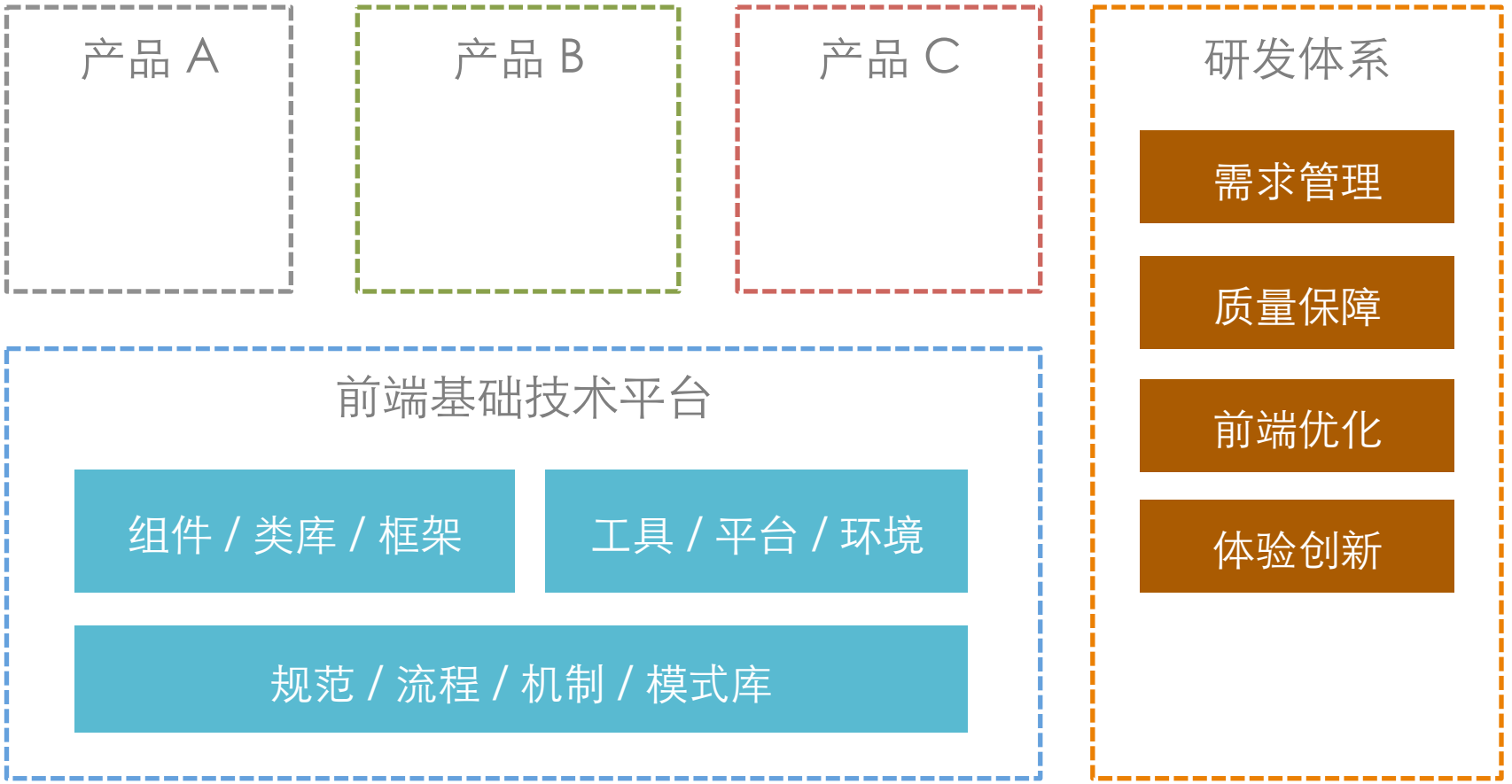
前端业务分类

给用户看的

给用户用的

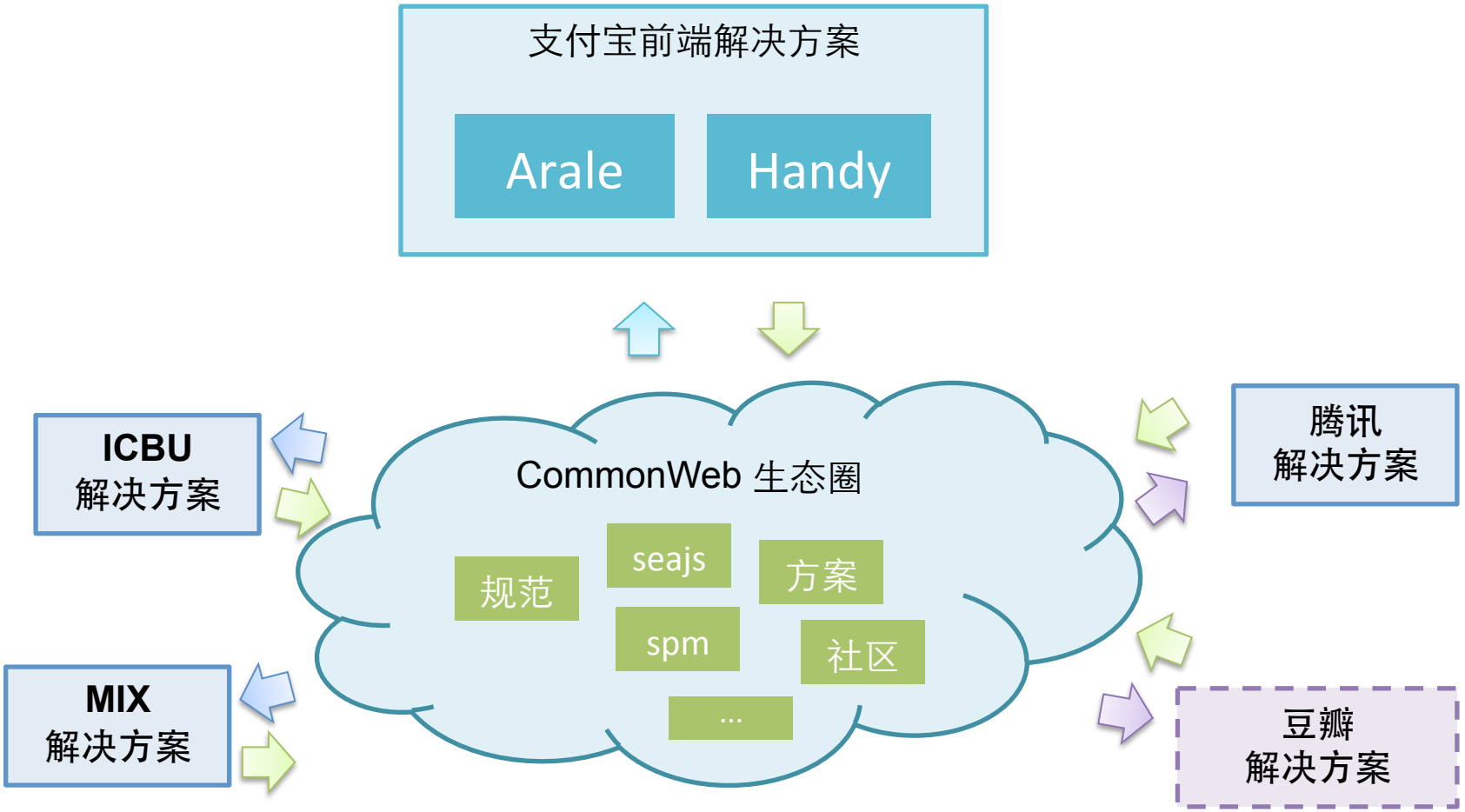

展现型业务
功能型业务
支撑
?

前端技术平台


产品 A
产品 B
产品 C
研发体系
需求管理
质量保障
前端优化
体验创新
前端基础技术平台
组件 / 类库 / 框架
工具 / 平台 / 环境
规范 / 流程 / 机制 / 模式库

支付宝前端基础技术平台

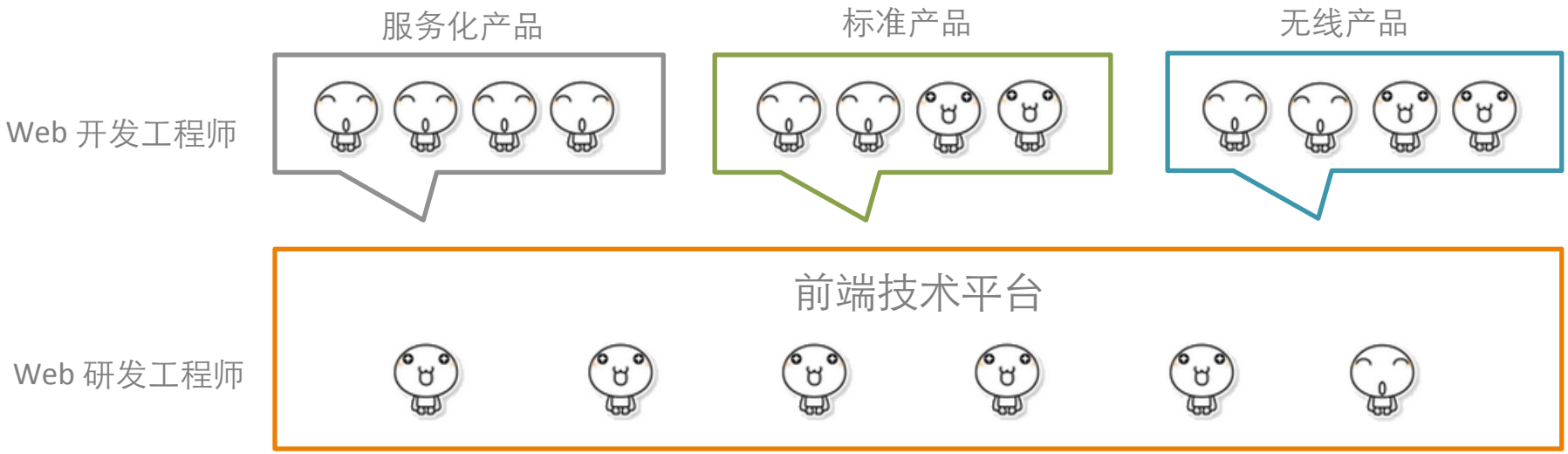
CommonWeb 体系



还要考虑人的需求

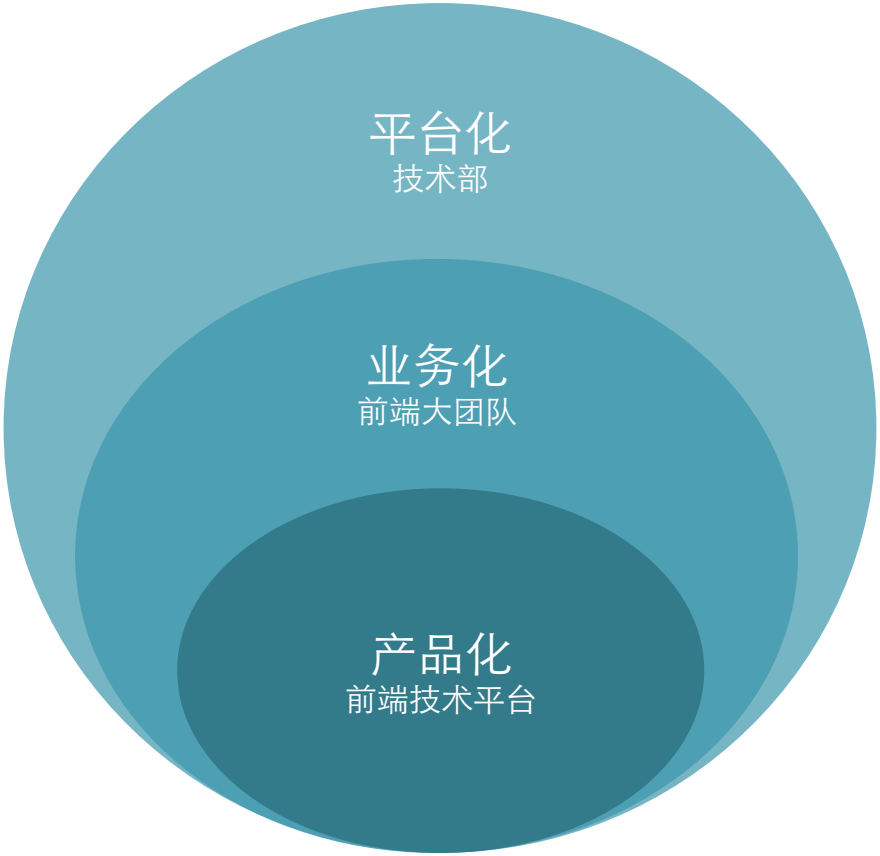
- 成就感
- 发展空间

研发模式



由资源型团队转换成服务型、创新型团队

建设思路

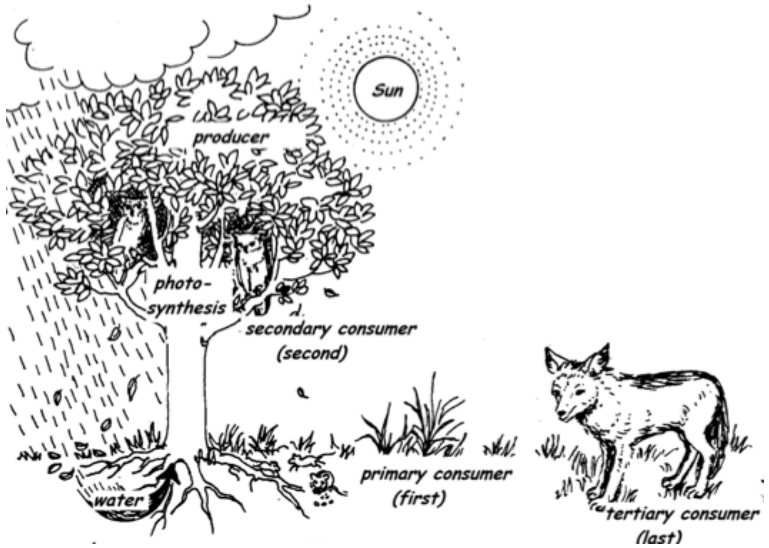


建设原则

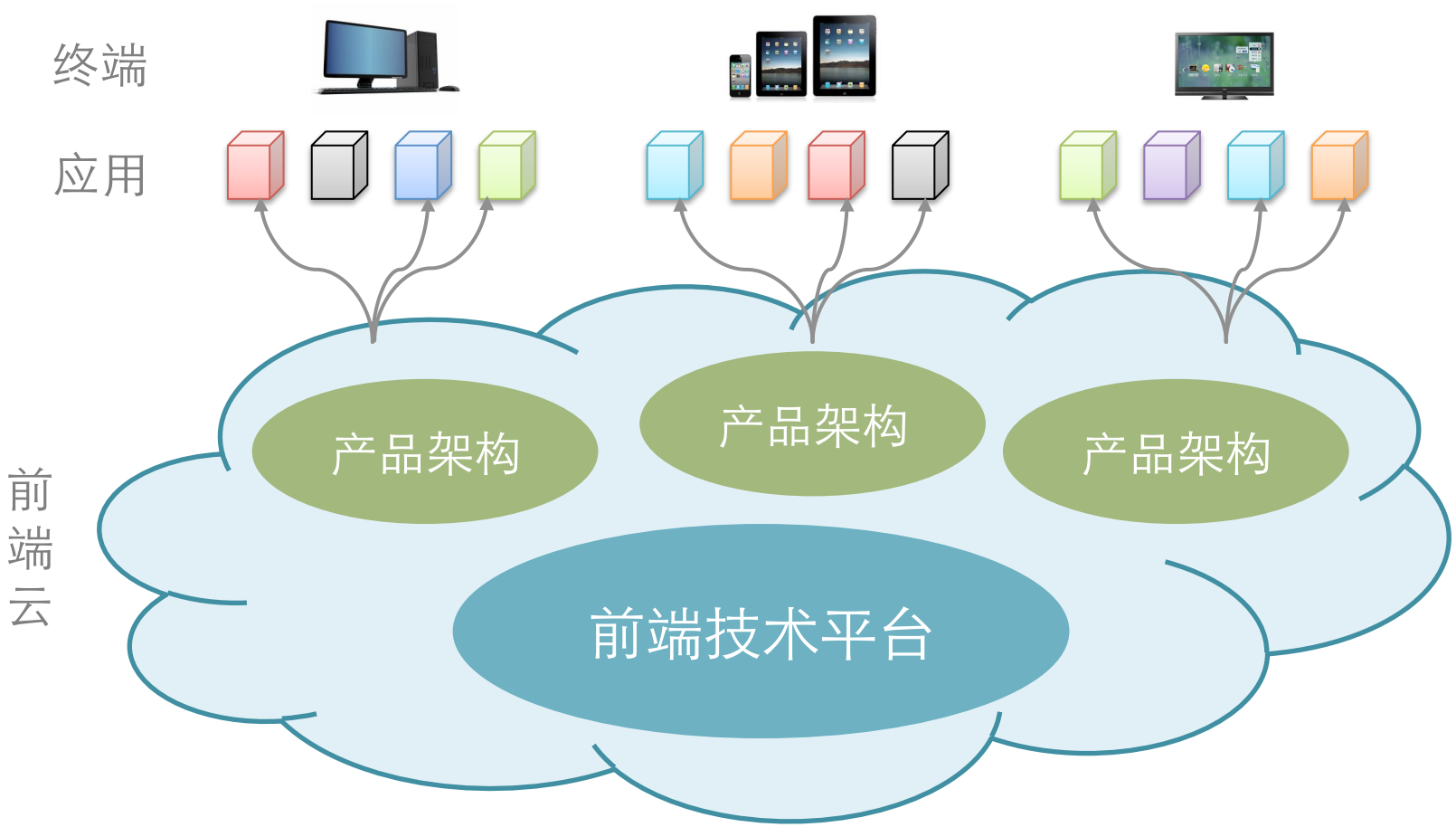
- 保障业务的正常进行
- 兼容现有架构，支持并行一段时间
- 3年内建设完成，持续交付

未来展望

- 『小而美』的模块生态圈
- 技术驱动产品创新



前端的云时代



seajs.log(“儿童节快乐”)