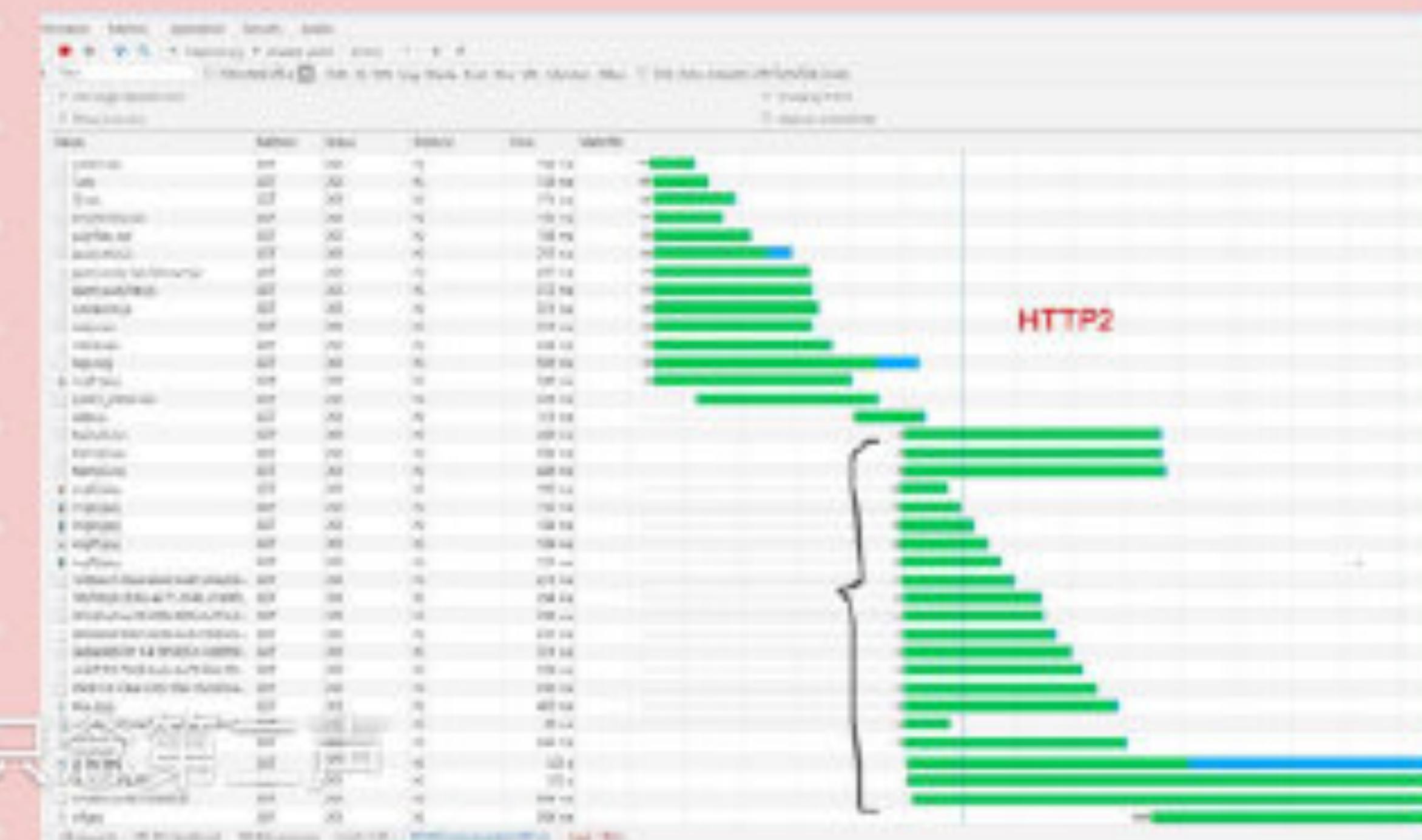
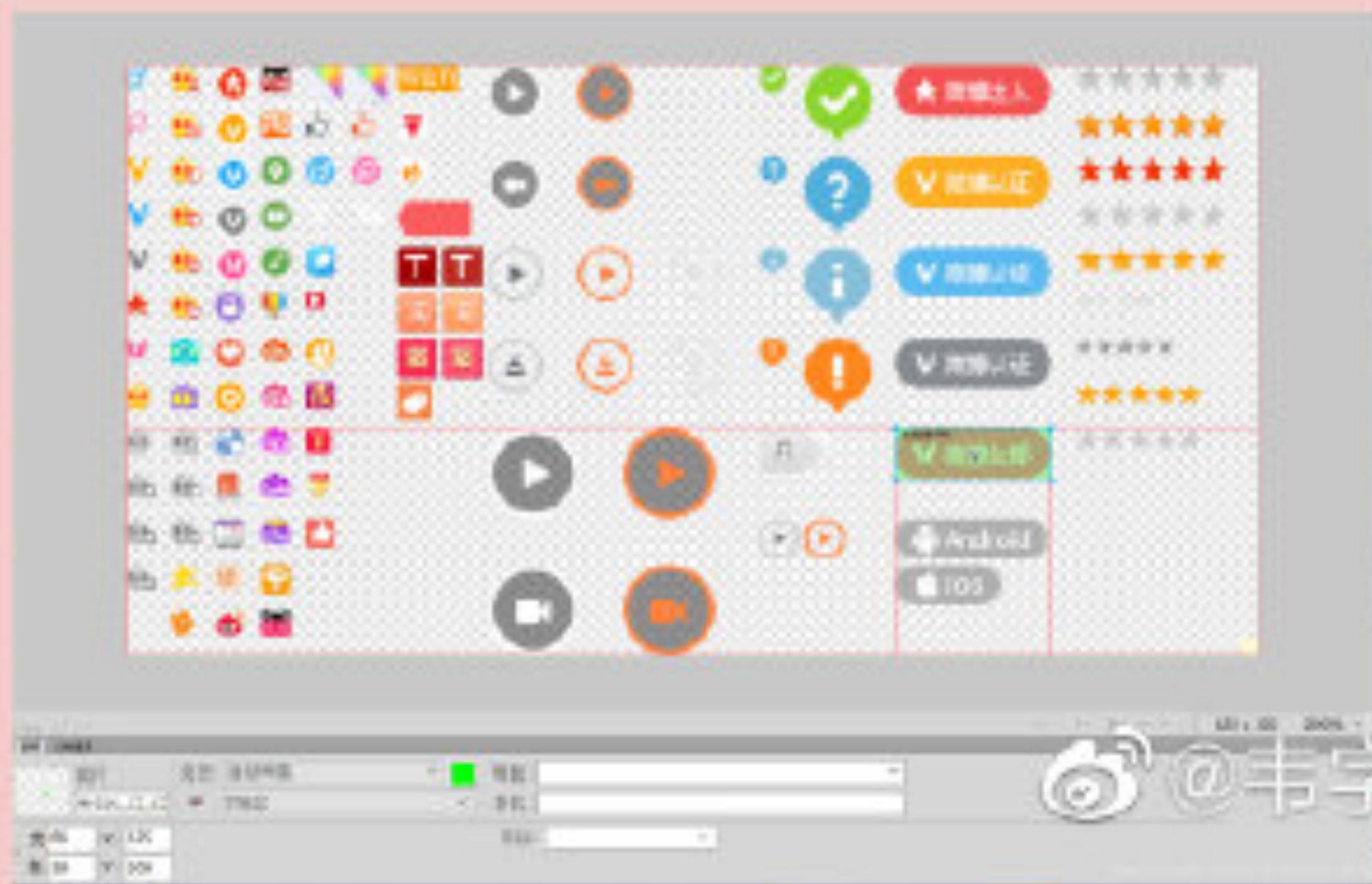


架构的演化与软件的未来



必须要合成一张图片。我这一次加载全部，能省很多流量，减少请求数！



20年前



现在

图片肯定要分开加载才得，
图片资源更好维护。

```
<html>
  <head>
    <style>
      .title {
        color: red
      }
    </style>
  </head>
  <body>
    <p class="title">Hello World</p>
    <script>
      console.log("Hello");
    </script>
  </body>
</html>
```

```
<style>
  img {width: 100px; height: 100px}
</style>

<template>
  
  <HelloWorld msg="Hello Vue 3.0 + Vite" />
</template>

<script>
import HelloWorld from './components/HelloWorld.vue';

export default {
  name: "App",
  components: {
    HelloWorld,
  },
}
</script>
```

20年前
他们告诉我写一起不好维护



现在
他们告诉我写在一起更好维护



软件的本质

- 软件：完成某项任务的工具或某项服务的载体。
- 行业软件：面向特定行业人群，承载行业一组服务或帮助从业者快速或高质量的完成工作任务。

软件的架构

1. 系统架构(基础架构): 保证软件高效稳定运行 -> 云计算 (服务能力)
2. 软件架构: 工程化, 可扩展, 可复用, 可定制, 可维护 -> 框架 (生产能力)

主题

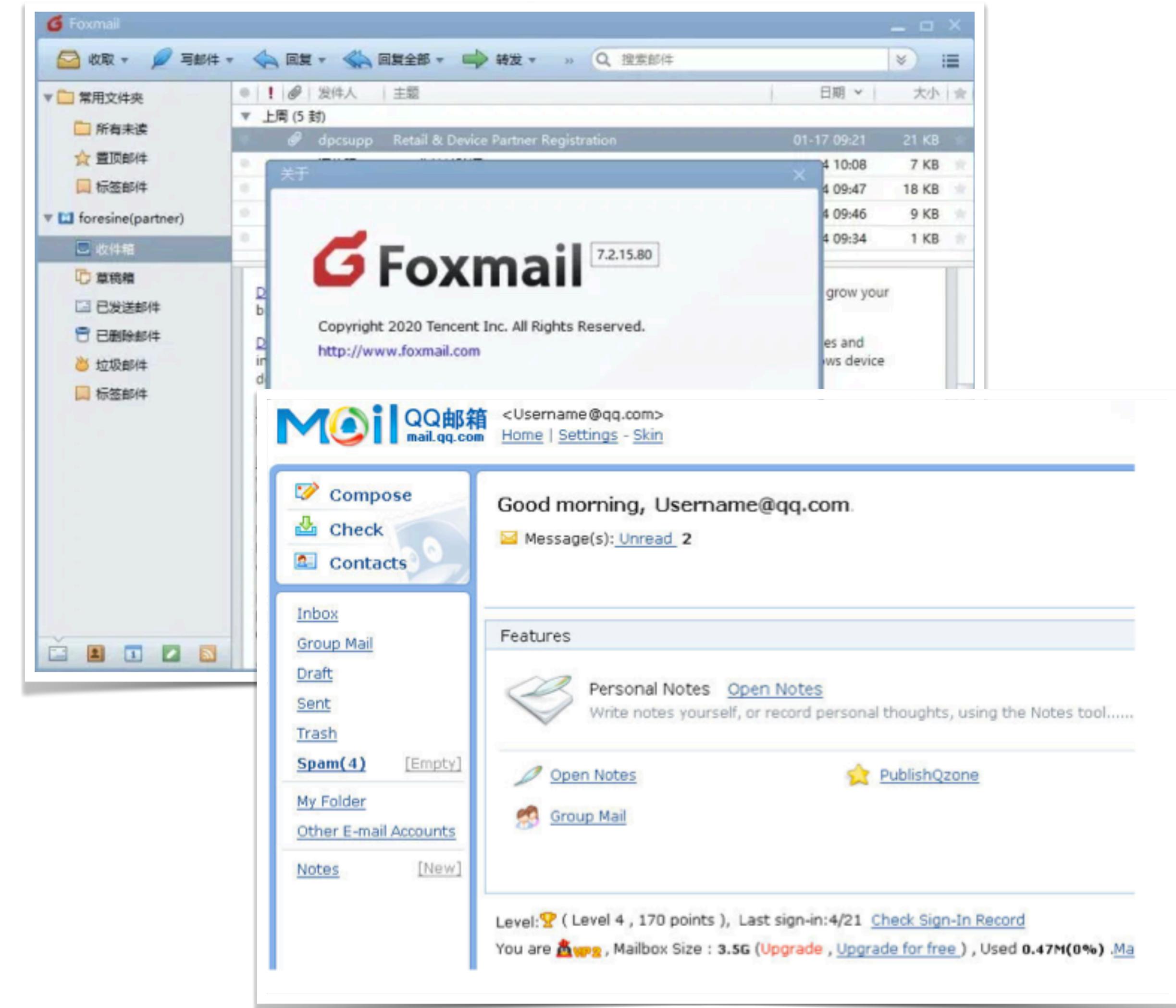
- 回顾软件架构的演变过程
- 展望软件以及行业的未来

第一部分：架构的演进

上古：C/S还是B/S

C/S到B/S的架构演进

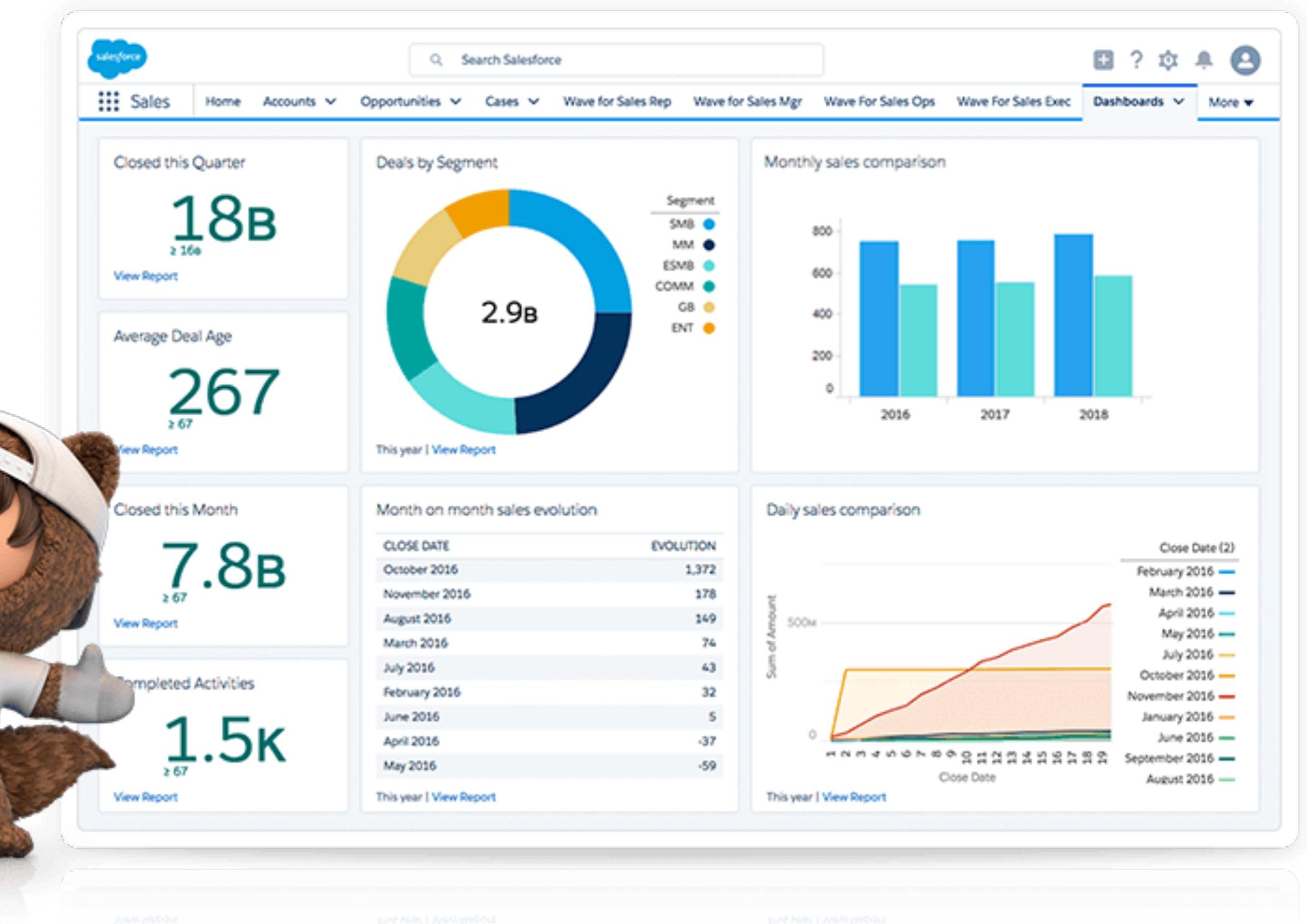
- C/S 操控体验更好，省带宽 (相对于B/S)
- B/S 系统升级更方便，使敏捷开发成为可能
- 从自由软件 Foxmail 到 Gmail 、QQ Mail
- 说服甲方使用B/S架构



古代：软件即服务(SaaS)

为什么 SalesForce 能够成功？

- 随着软件规模扩大，如何干好“工程”是“软件工厂”需要解决的重要问题之一。
- 软件制作：工程化按需制作(组件模块构建)
- 软件交付：零开发快速交付
- 中国的最佳实践：烟草行业某软件公司



近代·云计算

为什么云计算会是一场革命?

- “进入系统部的第一个考验是每天能搬多少台2U服务器”
- IT基础资源交付：从几天到开通即用
- 免运维：共享运维资源，降低业务维护成本
- SLA更高：大厂专业的运维团队，服务稳定性更有保障



近代: 移动互联网

C/S 架构回归

- 原生应用客户端 (iOS,Android...)
- 混合应用客户端 (PhoneGap, ReactNative, ionic, 各种H5打包器...)
- 全平台客户端 (Web + electron, flutter...)
- 流量平台的小程序 (微信, 支付宝, 抖音,百度...)
- 后端即服务 (BaaS)
- BaaS服务的实践



现代：新B/S应用

WEB技术栈将继续引领时代

- React, VUE, 小程序框架流行
- TS,JS 占据大部分市场份额， WEB技术栈仍是主流
- 前端组件化、一次布局多端呈现(跨平台)、SSR
- 后端服务化、低代码、业务无关（GraphQL、云开发、云函数）

# Ranking	Programming Language	Percentage (Change)	Trend
1	JavaScript	20.147% (-0.161%)	
2	Python	15.873% (-1.195%)	
3	Java	11.404% (+1.038%)	
4	Go	8.814% (+0.005%)	
5	TypeScript	7.488% (+1.479%)	^
6	C++	6.934% (-0.117%)	▼
7	Ruby	6.185% (+0.316%)	
8	PHP	5.030% (-0.034%)	
9	C#	3.716% (-0.092%)	
10	C	2.890% (-0.305%)	
11	Shell	1.863% (-0.196%)	
12	Scala	1.544% (-0.255%)	
13	Dart	1.113% (+0.346%)	^

未来?的软件架构

理想中的软件架构

- 运行时: 后端渲染, 前后端一体
- 开发时: 前后端分离, 前端搞定一切业务逻辑, 真正的持续集成, 保存即成品。
- 组件: UI组件前后端一体, 一次开发通过编译全端适配。有完善的设计和制作工具。
- 共享: 有更强的应用/组件hub, 快速获取, 组合成应用, 并且能够再发布。
- 复用: 应用可独立使用, 亦可拆解为组件/部件被其他应用使用, 改进后可在发布。
- 无代码/低代码: 90%的功能是通过搭积木的方式构建。终极目标: 产品经理可以不看研发小哥哥/小姐姐脸色, 自己搭建功能并交付给客户。
- 探索与实践 Yao Framework

Yao is an application framework that separates front-end and back-end, zero back-end code and UI components can be used in web、desktop and mobile app.

The name **YAO** comes from the Chinese word **爻**. In Chinese culture, The eight trigrams are made up of YAO. As the base symbol, Yao has two states, YIN, and YANG, Just like 0 and 1. Yao state changed, The eight trigrams changed, then things changed.

```
Max, a year ago | + author (Max)
1 <header @import="/components/header" ></header>
2 <nav @import="/components/nav" ></nav>
3 <section class="pet-list">
4   <div class="head">
5     L[[Recommend Today]]
6   </div>
7   <div class="body cards">
8     <card
9       yao:for="{{ pet in pets }}"
10      class="pet-card"
11      title="{{ pet.name }}"
12      cover="{{ pet.cover.url }}"
13      price="{{ pet.price }}L[$]"
14      description="{{ pet.description }}"
15      url="/detail/{{ pet.id_pet }}.html"
16    />
17  </div> Max, a year ago • + petshop demo
18  <div class="foot">
19    <button @click="next" class="btn btn-primary">L[[More]]</button>
20    <a href="/assets/images/cat-1285634_640.png" >L[[Read]]</a>
21    <a href="/assets/images/cat-1285634_640.png" >L[[Read]]</a>
22    <a target="_blank" href='/assets/images/cat-1285634_640.png'>L[[Read]]</a>
23  </div>
24 </section>
25 <footer @import="/components/footer"></footer>
26
27 <!-- Page setting-->
28 <yao-setting type="json">
29 {
30   "entries": [
31     { "router":"/", "ttl":3600 },
32     { "router":"/index.html", "ttl":3600 },
33     { "router":"/pets", "ttl":0 },
34     { "router":"/pets.html", "ttl":0 }
35   ],
36   "imports": {
37     "card":"/components/card",
38     "button": "@yao/material-ui/button",
39     "notification": "@yao/material-ui/notification"
40   },
41   "data" : {
42     "title": "Recommend pets",
43     "pets": {
44       "api" : "/yao-cli/unit-test-api/petshop/pet/search",
45       "query": {
46         "orderby": "recommend",
47         "lang": "en"
48       }
49     }
50   }
51 }
```

总结

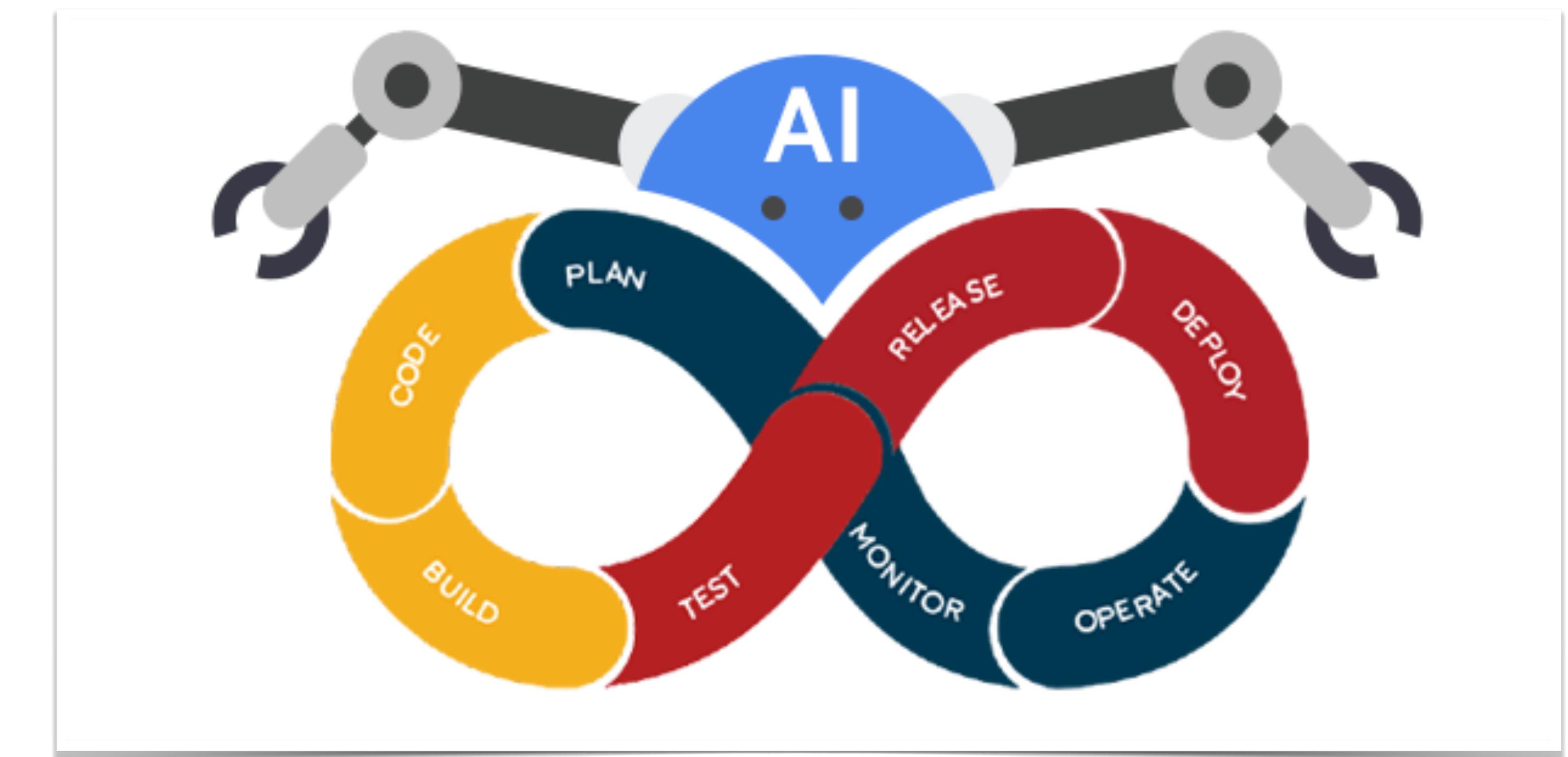
- 软件架构和软件形态随着硬件环境、网络环境升级而革新。
- 软件架构也会随着协同方式的变革而改变，比如云计算。
- 软件开发趋近以无代码或低代码方式进行。
- WEB技术栈随着网络环境升级将愈加流行。
- TypeScript 是最好的语言。

第二部分：软件的未来

自动化 & AI

自动化 & AI 大势所趋

- DevOps 已有很多最佳实践
- 无代码 & 低代码元年
- “除了艺术人工智能搞定一切”
- 一个“智能”研发的场景
- 我们离这个场景有多远？



VR/AR 的行业应用

当软件行业遇上VR/AR

- 3D 建模的行业软件
- PS的VR版本
- 未来软件交互的方式?



未来的工作模式

分布式办公或成为主流

- 软件行业早已具备分布式协同工作的条件
- 疫情加速分布式办公成为主流的进程
- FB 微软等大公司引领
- VR/AR持续改进远程交流的体验
- 有机会雇佣全球优秀人才
- 未来也许会进化出超越“公司”的组织形式
- 协作模式改变也必将改变软件行业业态

未来的软件行业

回归软件的本质, 关于行业的思考和观点

- 最好的软件是没有软件。
- 谁能生产出最适用的软件?
- 数字化-人类社会在网络世界的全息投影。
- 通过投影数据可以精确的预测未来。
- 不能玩转数据的传统公司，将被淘汰，新一轮行业洗牌正在进行。
- 随着软件制作门槛逐步降低，软件将与行业融为一体。
- 未来能力领军行业的公司一定是玩转数据的“先知”



谢谢