Webフロントエンドと
アーキテクチャ事情の持論を喋る

～ 我々は JavaScript エンジニアではないという強い気持ちを添えて

@ahomu（Ayumu Sato）| TechFeed Experts Night vol.4　2022.09.21

@ahomu

- TechFeed Official Expert No.25
- Web Frontend Engineer, VPoE
  - Offers | overflow, inc.
- Author of #超速本
  - w/@1000ch

前回のおさらいと今回のこと

TechFeed Conference 2022

- SPA vs MPA の構図はもう無理
  - CSR と SSR の局所使い分けが実状
  - フレームワークサポートも多様化
  - もはや設計具体ではなく偏った思想
- 設計のキモはどこにあるのか
  - HTML、キャッシュ、JavaScript
  - 色んなアプローチがあるよね
  - 今日はこっちを主に再編してお届け

https://techfeed.io/entries/62a8534290ee1f2ca98b1a2a

今回のアジェンダ

- フロントエンドとは
- フロントエンドアーキテクチャとは
- アーキテクチャ設計の要点
- 実例を交えたシミュレーション
- まとめ

ではやっていきましょう

Frontend ?

ユーザーとサービスを UI で繋ぐために
HTML 生成処理を整える技能領域

OR/AND

システムとデザインの境界で責任を持ち
ユーザーに届ける品質を司る職能

Decays without doing anything

- 通信環境、デバイス、ユースケース、デザイントレンド etc
- すべて時間の進み、技術の歩みと共に変化し続けている
- クライアントサイドはその変化を直接その身に受ける
- なにもしなければユーザー体験も開発体験も朽ちていく
- だからこそ、フロントエンド技術は How をアップデートし続けている

Transitions of Frontend

Node.js の台頭、フルスタックフレームワーク普及
いわば JavaScript エンジニアを経て
Webアプリケーションエンジニアへの変化

Client Side

JavaScript a.k.a jQuery

CSS Styling

HTML Template

Server Side

Server Framework

Database

UI Components

Cache Strategy - Edge Computing

Web Application - CI/CD

API Gateway

Truly Backend Services

バックエンドエンジニアが提供する価値の中心も
Webアプリケーションの後方にシフトしている
モバイルアプリの存在も API 前後の線引きに影響？

Web Frontend Architecture

- ソフトウェアにとって性能（速さ）は価値そのもの
- **アーキテクチャは性能と向き合っている**
- 一方で、開発体験/効率の名の下に保守性や生産性とも向き合っている
- 性能と開発体験の天秤を揺らし続けているのが近年のフロントエンド

Modern History（記憶が定かでないので順不同）

- 各種ライブラリの台頭により**コンポーネント志向開発**が本格的に普及
- Node.js を中心とした**ビルドプロセスとエコシステムが重層的に発達**
- **1**つのコードベースで Server / Client に対応する実装（SSR + CSR）
- Server → Client で**状態を引き継ぐため Hydration** が誕生
- JavaScript ですべてを解決する開発者体験の御旗（e.g. CSS in JS）
- サバクラ共通化されたコードベースに **npm でライブラリがマシマシ**

Seriously bloated JavaScript

- 肥大化の一途を辿ったクライアントサイドで実行される JS
  - ファイルサイズによる Networking コストの問題は当然ある
  - しかしディスクキャッシュ経由でも生じる Computing コストのほうが深刻
- JS を Web の CO2 と喩えられるのも言い得て妙
  - 開発者体験を支えるため JavaScript に依存しがちな Web 開発の現状
- Hydration 課題はきっと Yosuke Furukawa or/and りぃ-san が喋る

設計の要点

どこで HTML を生成するか
どのようにキャッシュ戦略を描くか
どれくらい JavaScript を載せるか

どこでHTMLを生成するか

- サーバーサイド
  - 特に静的なコンテンツならサーバーサイド + CDN が最速で HTML を返すのが最強
- クライアントサイド
  - とはいえ、インタラクティブなサービスはクライアントサイドでも生成が必要になる
- エッジサイド
  - 近年、CDN〜クラウド業者を中心にエッジで諸々を処理できる環境が整いつつある

どのようにキャッシュ戦略を描くか

- キャッシュ自体はあらゆる箇所に差し込む余地がある
  - サーバーアプリケーション（データベースとかオンメモリとか）
  - ミドルウェア（nginx proxy cache とか、Varnish とか）
  - CDN（キャッシュだいすきコンテンツデリバリーネットワークさん）
  - ブラウザ（ローカルキャッシュ、ServiceWorker CacheStorage とか）
- 鮮度 & 整合性と戦いつつつも HTML は CDN に食わせたいのが本命
  - CDN は転送距離の課題も解決するので頭ひとつ抜けている
  - ストリーミング動画ファイルもゲームのパッチファイルも CDN が大好きサ

どれくらい JavaScript を載せるか

- JavaScript のバンドルサイズは、クライアントサイドの仕事量と比例
  - 極端な話、JavaScript ゼロで HTML 最速返却しまくるだけで優勝可能
  - 現実にはそうもいかないのでバンドル分割や減量に励むわけである
- ヘビーになりうるパラメータ
  - インタラクションが必要な UI の有無、およびその複雑性
  - CSR の有無
  - フルスタック SPA フレームワークの(予防的)導入の有無
  - Hydration の有無

提供物の理想状態にあわせて
これらの要点を総合的に判断する

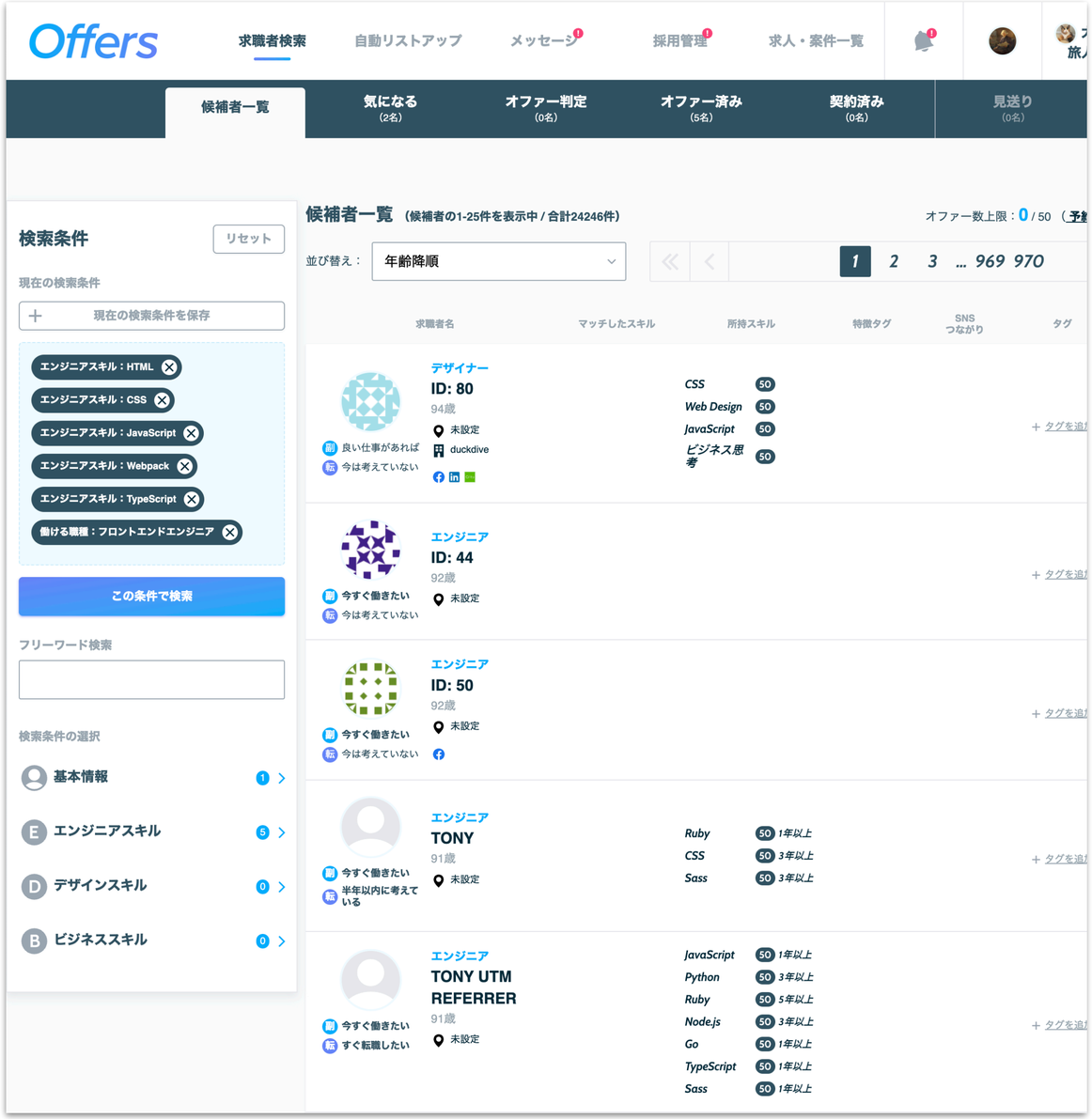
シミュレーション

Offers Magazine

- 静的なオウンドメディア
  - リアルタイム性が乏しい
- インタラクティブ性は極低
  - ログイン依存処理もほとんどない
- どういうHowがあるだろうか
  - HTML は SSR？ 事前静的生成？
  - キャッシュは CDN に置けそう？
  - CSR + Hydration は本当に必要？

Offers 候補者検索

- ログイン前提のサービス
  - HTMLはキャッシュしようがない
- 検索など動的な UI が多い
  - SPA っぽくする余地がある
- どういうHowがあるだろうか
  - HTML のキャッシュは無理そう？
  - 割りきって SPA っぽく構成する？
  - 肥大化しうるJSをどう最適化する？



まとめ

サマリ

- フロントエンドは Web アプリケーション全域が守備範囲になりうる
- 性能と開発体験の間で揺られるなかで設計上の課題も生まれている
- 設計のポイントは、HTML生成・キャッシュ戦略・JavaScript 積載量
- アプリケーションの要件にあわせた設計を！

When solving a problem of interest, do not solve a more general problem as an intermediate step.

Statistical learning theory by Vapnik, V. N, 1998

ありがとう
ございました！

複業転職マッチングプラットフォーム Offers もよろしくね！