RMP エンジニア・ブートキャンプ 2017

ようこそ
webフロントエンドの世界へ

- Welcome to the world of the Web frontend -

Naoki YAMADA
@wakamsha

2017.05.30

Introduce myself

- 自己紹介 -

俺の名前を言ってみろ

- Introduce myself -

Naoki YAMADA 山田 直樹

@wakamsha

株式会社リクルートマーケティングパートナーズ

web フロントエンドエンジニア

NET BIZ DIV. TECH BLOG

https://tech.recruit-mp.co.jp

1. 混沌と狂気に満ちた JavaScript の世界
2. この美しくも醜い CSS の世界
3. web フロントエンドのトレンドを追いかけるコツ
4. これから web フロントエンドを本格的に学ぶなら？
5. 課題のお知らせ

Agenda

本日お持ち帰りいただきたいこと

1. web フロントエンドの各種バズワードの整理整頓
    1. 界隈でよく言われている疑問の解消
    2. よくある勘違いの解消
2. 簡単な web フロントエンド開発環境の作り方

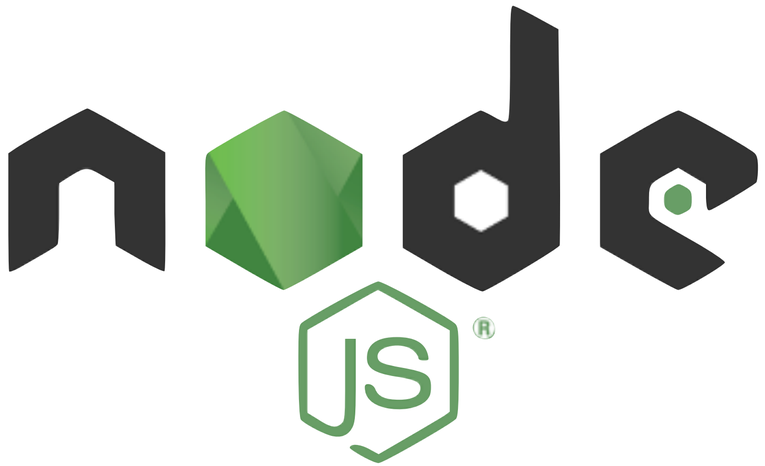
Points to learn

本日お話しないこと

1. JavaScript のテスト関連全般
2. JavaScript フレームワークに関する (宗教戦争的な) アレコレ
3. CSS の仕様やコーディングのお作法とか
4. HTML 関連全般

- 混沌と狂気に満ちた JavaScript の世界 -

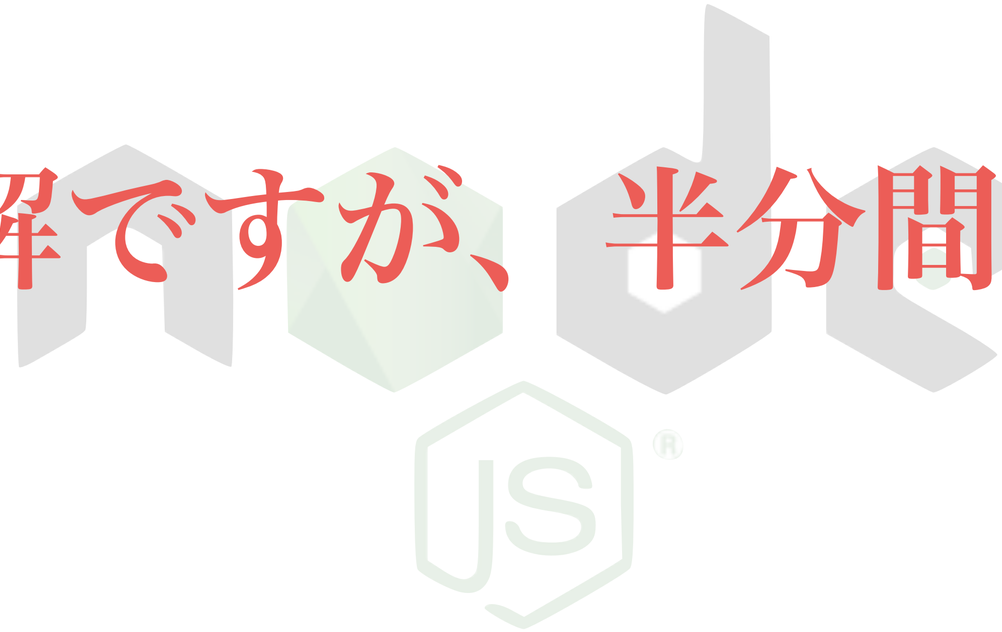
- Crazy and chaotic JavaScript world -



- Node.js とはナニモノ？ -

サーバーサイド JavaScript のことでしょ？

半分正解ですが、半分間違いです



- Node.js とはナニモノ？ -

- Node.js とは JavaScript の『**実行環境**』である
- 元来 web ブラウザ上で動かすことが前提となっていた JavaScript を『ブラウザ以外の環境でも動かしたい』というニーズが生まれる
  - じゃぁ JS を実行する環境を別途作って、どこにでもインストール出来るようにすればいいんじゃね？
- **Node.js 爆誕**

ES6 とか ES2015 とかあるけど、何なの？

JS の元ネタになってるプログラミング言語でしょ？

元ネタと言えなくもないですが、言語ではありません

ES6 - ES6 とか ES2015 とかあるけど、何なの？ -

- ES とは **ECMAScript** の略称
- 派生言語を実装するうえで指標となる『**標準**』ないし『**規格**』である
  - このようなプログラミング言語は存在しない
  - JavaScript はこの ECMAScript に基づいて設計・実装されている
  - ActionScript3 も ECMAScript 準拠の言語 (ES3 相当)

ES6 - ES6 とか ES2015 とかあるけど、何なの？ -

- ES6 は **ECMAScript 6th Edition** を意味する
  - IE11 を除く現行のブラウザ (Node.js 含む) はほぼ対応済み
  - ES2015 は ES6 の別称で、途中からこれに呼び名が変わった
  - 『2015』 は仕様が確定して正式公開された年を意味する
    - **主要ブラウザの実装が完了された次期ではない**ことに注意

TypeScript？CoffeeScript？Dart？
JavaScript と何が違うの？

全て JavaScript を生成するための『中間言語』です
これらは『AltJS』もしくは『JS プリプロセッサ』と
呼ばれています

JavaScript の中間言語 (AltJS) たち

- 全て最終的に JavaScript となる
- **ブラウザも Node.js も原則として実行できるのは JavaScript のみ**
  - TypeScript, CoffeeScript, Dart を直接実行することはできない

“JavaScript はもはやブラウザのためのアセンブラ言語である”

誰かが言った格言

JavaScript の中間言語 (AltJS) たち

- JavaScript はブラウザにおけるアセンブラ言語のようなもの
  - TypeScript, CoffeeScript, Dart はJVM における Java, Scala, Groovy みたいな位置づけだと思ってもらえればだいたい OK

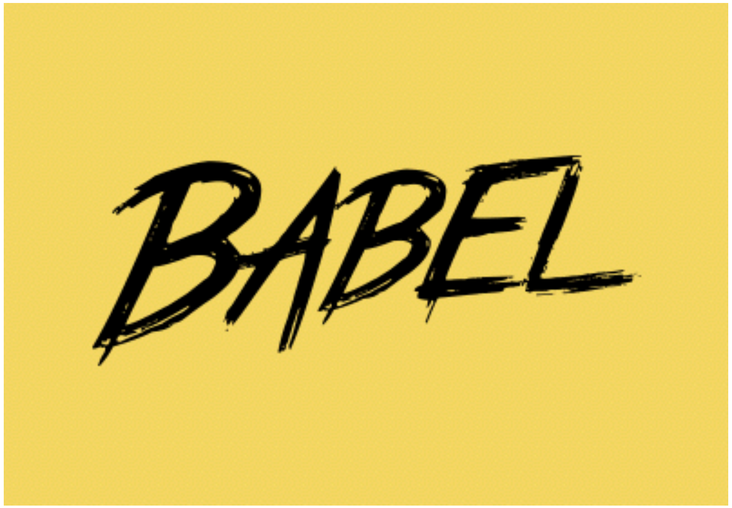
最近 flowType というのが流行ってるみたいだけど、
これも AltJS の一種？
TypeScript とよく比較されてるよね？

flowType は JavaScript に型情報を補填するだけ

どちらかと言うと『Lint』の方が近いです

flowType は型情報を後から補填するツール

- AltJS のようなプログラミング言語にあらず
- 既存コードにも後から部分的に適用が可能なので、導入障壁が低め
- JS に型をもたらすということで TypeScript と比較されがち
  - TypeScript は『JS のスーパーセット』というプログラミング言語なので、そもそものコンセプトが異なる
- 同じ Facebook 製である React と併用される事例が多い



じゃぁ BABEL ってのは？

JavaScript に後方互換性を持たせた状態に
トランスパイルするツールです

JavaScript に後方互換性を持たせる

- ES2015 や ES2016 の仕様で書かれたコードを ES5 実装の環境でも動くコードにトランスパイルする
  - const や let は全て var に置き換えられる
  - Promise 等はそれぞれの Polyfill メソッドに置き換えられる
- IE11 など ES2015 非対応のブラウザをサポートするためには必須のツール

Grunt？Gulp？なんですかそれ？

web フロントエンドにおける『タスクランナー』です。
Java で言うところの Gradle みたいなものだと思ってください

タスクランナー？ビルドツールとは違うの？

違います。ビルドツールを『動かす』ための
『タスクランナー』です

web フロントエンドにおけるタスクランナー

- 出来ることは『ビルド』に限定されない
  - ex) ローカルサーバの起動、ファイル / ディレクトリのコピー、etc…

web フロントエンドにおけるタスクランナー

- Grunt の登場で web フロントエンド界隈でタスクランナーというものが認知されはじめる
- 後発の **Gulp 登場で爆発的に普及**
  - Grunt はハッシュ形式でタスクを定義し実行する
  - Gulp はチェーンメソッド風にタスクを定義し実行する
  - Gulp の方が処理の流れが追いやすく記述も統一されやすいので人気

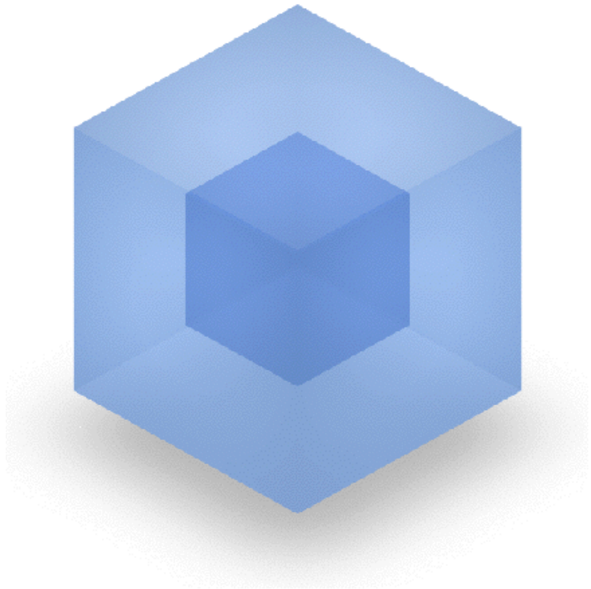
browserify はまた違うもの？

browserify はビルドツールの一種です

ビルドツール？ Gulp とかとは違うの？

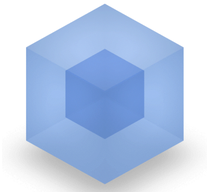
JavaScript の依存関係を解決するツール

- JavaScript の依存関係を解決し、最終的に単一のファイルとして吐き出す
  - れっきとしたビルドツールの一種



じゃぁ webpack というのは？

より多機能な『ビルドツール』です。



browserify よりも多機能なビルドツール

- browserify の後発といった立ち位置
- JS だけでなく CSS や画像ファイルの依存関係も解決できる
  - ※ browserify が扱えるのは JS のみ

なんか npm script ってのもあるみたいだけど？

よりプリミティブな『タスクランナー』です

よりプリミティブなタスクランナー

- npm script で実行できるのは単なる『コマンドライン』にすぎない
  - CircleCI の circle.yml みたいなものだと思ってもらえれば OK
- Gulp ほど複雑なタスク定義には向かないが、ある程度はこれでもカバー出来る

よりプリミティブなタスクランナー

- Gulp よりもシンプルな分、処理速度が速い
  - Gulp 用の依存モジュールがまるごと無くなるので、アップデートしたらタスクが動かなくなったという事態が発生しにくくなる
- Gulp のようにイベントストリームに乗せてシーケンシャルなタスクを実行する必要が無いのなら、こっちで充分

なんかいろいろとモノが多すぎてカオス過ぎません？

ブラウザで動くアプリケーションを作ろうというのが

そもそも狂気の沙汰なのです

混沌と狂気に満ちた JavaScript の世界

- Vanilla JS をダイレクトに記述するだけではカバーしきれない領域に僕らは置かれている
- かつては JS, CSS, HTML をそのまま書くのが当たり前だったが、その代わりそこまで複雑な要件ではなかったはず

ブラウザ・アプリケーションとは複雑怪奇なもの

- JavaScript, CSS, HTML は、もともとアプリケーションを作るためのテクノロジーではない
- にも関わらずそれで無理やりアプリを作ろうとしているのだから、カオスになるに決まってるじゃん
- というか web ページって**元を辿れば『ドキュメント (文書)』**ですよ？

jQuery ってオワコンなんですか？

そんなことはありません

要件によっては今でも全然使えます

web サイト 用途なら今でも全然使える

- もともと jQuery とは各種ブラウザ間の挙動の違いを吸収するために生まれた『**DOM 操作 API**』である
- 登場して間もなく Ajax (XMLHttpRequest) 通信のラッパー API が実装されたことで、アプリっぽい用途にも使われるようになったが、本質はあくまで DOM 操作 API である

web サイト 用途なら今でも全然使える

- web サイト内の要素をグリグリ動かすなど、用法用量さえ間違えなければこれほど便利なライブラリは他にない (現在進行形)

- オワコンと言っているのは、より複雑な web アプリを作っているごく一部の『**酔狂な人々**』であり、彼らの多くは件の web サイト制作のことは一切視野に入れていない

1. 混沌と狂気に満ちた JavaScript の世界
2. **この美しくも醜い CSS の世界**
3. web フロントエンドのトレンドを追いかけるコツ
4. これから web フロントエンドを本格的に学ぶなら？
5. 課題のお知らせ

Agenda

- この美しくも醜い CSS の世界 -

- This beautiful & ugly CSS world -

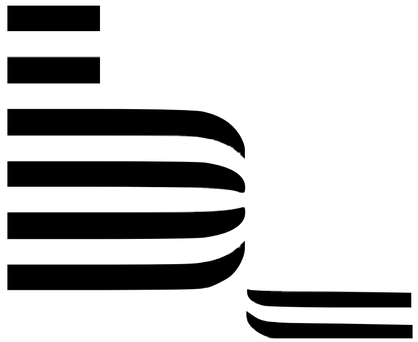
思った通りに反映されない！

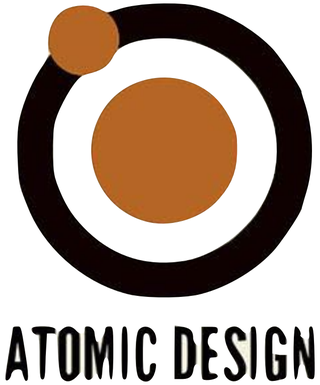
直感的じゃない！

欠陥仕様だろこれ！

CSS が直感的じゃないと思われがちな件について

- そもそもアプリのような複雑な用途で使うことを想定していない
  - 適用対象が HTML (文書) なんだから仕方ないね
  - float: left; を使って段組みレイアウトを実現するってのは『**ハック**』ですからね、みなさん
  - 最近は flex というレイアウトに特化した機能が実装されたのでだいぶ楽になった (※ IE11 も対応済)

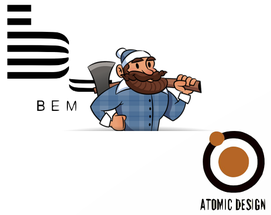




BEM とか SMACSS とか ATOMIC DESIGN って何？

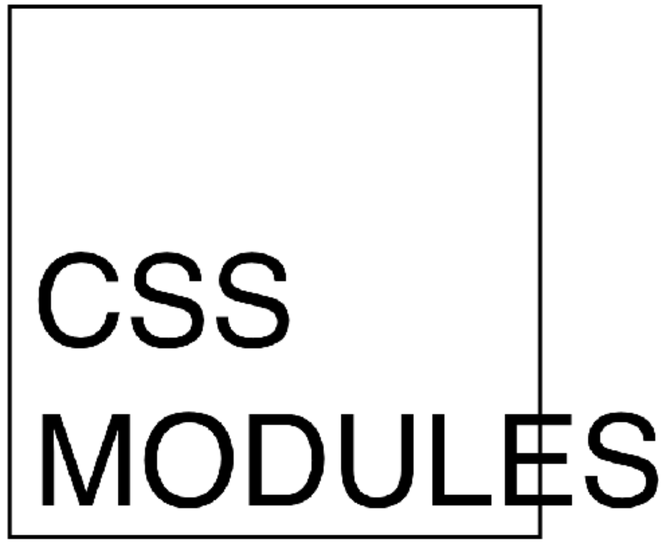
全て CSS を綺麗に書くための『設計思想』です

ライブラリではありません



さまざまな CSS の設計思想

- どれも一長一短であり、現時点で決定打は存在しない
  - BEM が比較的シンプルな規約なので導入しやすい
  - 後発の ATOMIC DESIGN がここ最近は注目されている



最近 CSS MODULES なんてのを聞くんだけど？

CSS
MODULES

アプリケーションに特化した CSS 設計思想

- 文字通り CSS を『モジュール』として断片的に各コンポーネント内に取り込むという設計思想
  - React, Angular, Vue といったコンポーネント志向の JS フレームワークにおいて使われることが前提
  - いい線行ってるけどまだまだ課題も多い印象

SCSS ? Sass ? Stylus ?

CSS と何が違うの？

全て CSS を生成するための『中間言語』です
これらは『AltCSS』もしくは『CSS プリプロセッサ』と
呼ばれています

CSS の中間言語 (AltCSS) たち

- 全て最終的に CSS となる
- **ブラウザが原則として実行できるのは CSS のみ**
  - SCSS / Sass や Stylus を直接実行することはできない
  - CSS もまた複雑化の一途を辿っており、アセンブラ言語のようになっている
  - ならば JS 同様もっと高機能で書きやすい言語でコーディングしてから CSS にトランスパイラすればいいじゃん

最近 PostCSS というのが流行ってるみたいだけど、
これも AltJS の一種？

AltCSS ではなく『CSS フレームワーク』です

PostCSS

CSS の新しい機能を先取り

- 『**変数機能**』や『**ネスト機能**』などさまざまな『**モジュール**』を自由に組み合わせて自分だけのトランスパイル環境を作る
- 次期 CSS の仕様を全て先行実装したモジュールが『**cssnext**』として提供されている
  - 未来の CSS 標準仕様に則ることが出来るというメリット
  - SCSS や Stylus と比較すると機能は数段劣るため、よほどのもの好きでない限りオススメはしない

1. 混沌と狂気に満ちた JavaScript の世界
2. この美しくも醜い CSS の世界
3. **web フロントエンドのトレンドを追いかけるコツ**
4. これから web フロントエンドを本格的に学ぶなら？
5. 課題のお知らせ

Agenda

web フロントエンドのトレンドを追いかけるコツ

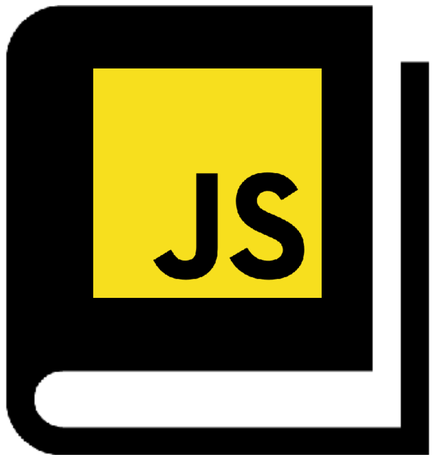
web フロントエンドのトレンドを追いかけるコツ

- 他の世界と比較して変化が激しい（※ 特に JavaScript 関連）ので、なるべく新しい情報だけを参考にする
  - ググる時は『**期間指定 = 1 年以内**』に設定するくらいでちょうどいい
- 『XXX はオワコン』『まだ XXX で消耗しているの？』といった**オラついた発言にいちいち耳を傾けない**
  - 自分が贔屓にしている技術を持ち上げるために他を貶す発言をする人ほど生産性が低く視野が狭かったりする

1. 混沌と狂気に満ちた JavaScript の世界
2. この美しくも醜い CSS の世界
3. web フロントエンドのトレンドを追いかけるコツ
4. **これから web フロントエンドを本格的に学ぶなら？**
5. 課題のお知らせ

Agenda

これからweb フロントエンドを本格的に学ぶなら



JavaScript の場合

JavaScript の場合

- 実行環境は web ブラウザよりも Node.js の方がオススメ
  - モジュール機能がデフォルトで備わっているのでファイル管理・依存関係解決が比較的楽ちん
  - browserify とかそんなもんは後だ後！
  - DOM 操作が一切無い世界なので、ビジネスロジックに専念して言語仕様を学べる
  - DOM 操作を学ぶなら当然 web ブラウザ一択 (※ 基本的に Chrome のみで OK)

JavaScript の場合

- 早い段階で flowType を導入する
  - 型情報はとても大事
  - **型はコードに最も密接したテストであり、規約である**
  - flowType 部分的に導入することが出来るので、敷居が低め ＆ 捨てるのも容易
  - JS 自体の文法は素のままなので、ピュアな JS を型安全が保証された状態で学べる
  - TypeScript は機能が豊富過ぎるので、ある程度ピュアな JS に慣れてからのほうが吉

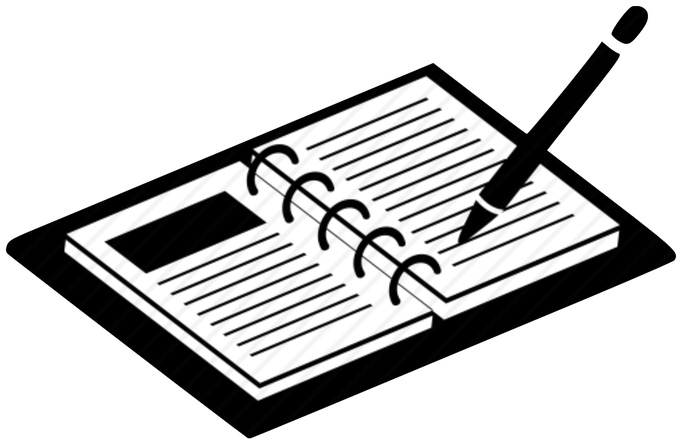
CSS の場合

CSS の場合

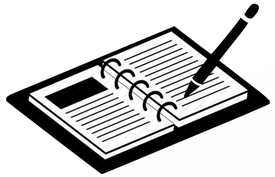
- デザインに対するモチベーションがある程度ないと途中でくじけがち
  - 『**web サイトを作ってみよう**』みたいな書籍を写経するとモチベーションが持続しやすいかも
- **早い段階で AltCSS を導入する**のが吉
  - 日本語の知見が最も多いのは SCSS
- BootStrap のソースコードを眺めるだけでもかなりの学びがある

1. 混沌と狂気に満ちた JavaScript の世界
2. この美しくも醜い CSS の世界
3. web フロントエンドのトレンドを追いかけるコツ
4. これから web フロントエンドを本格的に学ぶなら？
5. **課題のお知らせ**

Agenda



課題をお伝えします



課題をお伝えします

- 実際に web フロントエンド開発環境を自分で作ってみよう
  - JavaScript, CSS いずれも何かしらのプリプロセッサを使用すること (※ HTML は任意)
  - ローカルサーバの起動
  - ファイルを変更したら自動的にビルド処理が実行
  - コンテンツは『Hello, world』で OK
  - 自信がある人は 何かしらのフレームワークも導入してみよう

Thank you 😃