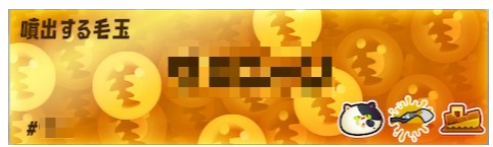
令和7年版
あなたが使ってよいフロントエンド機能とは

2025/1/18 北陸三県.rb Lightning Talks in Kanazawa / @mugi_uno

自己紹介

- Toyama.rb 主催者
- サイボウズ株式会社
  フロントエンドエキスパートチーム
- X: @mugi_uno
- Splatoon3 Salmon Run が好きです
  (全ステカンスト/ビッグラン等上位1%)

Toyama.rb



Webフロントエンドの進化

Interop

- ブラウザの相互運用性を向上するためのプロジェクト
- 毎年重点項目が列挙され、ダッシュボードでカバー率を確認できる
- ↓ は Interop 2024 の 2025/01/14 時点でのステータス

使えるようになった技術の例 / async,await

- Promiseを用いた非同期の取り扱いを容易にするSyntax
- Babelとかで変換するケースも多かったが、いまはブラウザネイティブで動く

```
async function foo() {
  const response = await asyncAPI();
  return response.body;
}
```

使えるようになった技術の例 / CSS Nesting

- CSSでネスト記法が使えるように
- SassやPostCSSなどのCSSプリプロセッサ経由で実現するケースも多かった

```
.button {
  span: {
    color: gray;
  }

  &:hover {
    font-weight: bold;
  }
}
```

新しい技術が増えたのはわかったけど…

「で、結局これって俺たちはもう使っていいの？」

「使えるようになった」とはなにか

- すべてのメジャーブラウザの最新版でサポートされた
- すべてのメジャーブラウザで使えるし、時間が経過して安定した
- 俺たちのサービスがサポートするブラウザで使えるようになった
- 世の中で使われるようになった気配を感じ始めた
- その界隈で有名そうな〇〇って人が使うと良いって言ってた
- ブラウザでは使えないけどトランスパイルすれば…
- Chromeで使えればオッケーでしょ
- …

コンテキストによって「使える」はそれぞれ

ある機能を「使ってよいか」はどう判断すればいい？

従来多く見られた方法
→ ブラウザとサポートバージョンを決め打ち

- 対象のブラウザを絞り、動作保証バージョンも絞る
- このバージョンを元に、個々の機能ごとに利用可否を判断

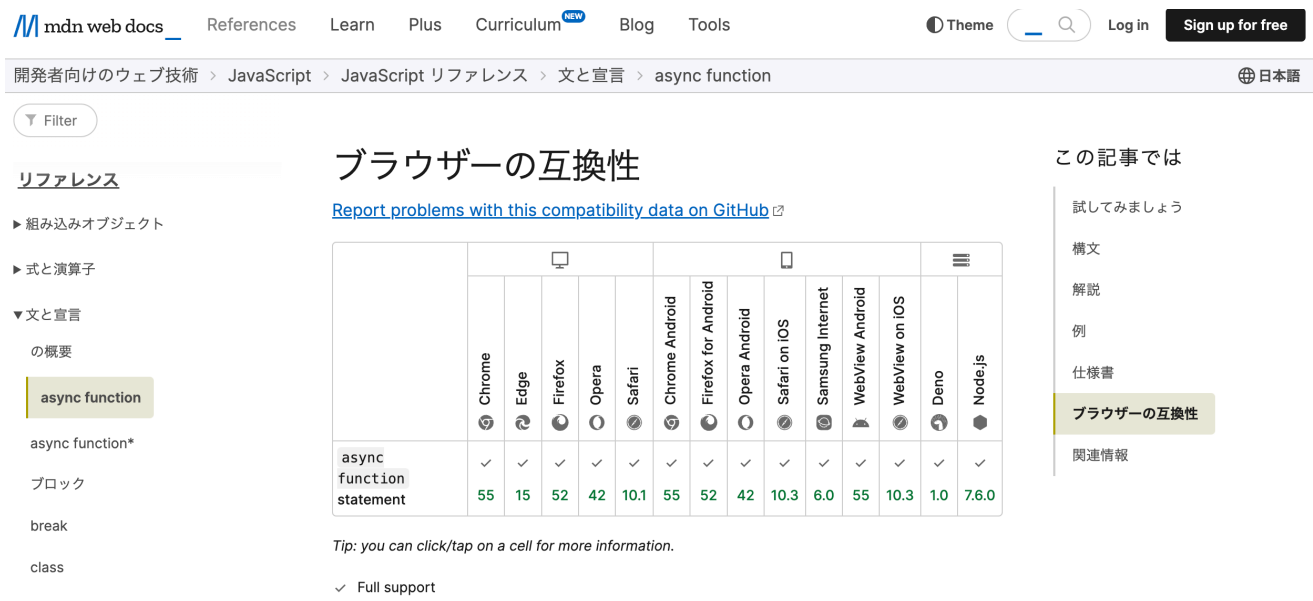
(例)

このサービスの動作保証環境は以下の通りです。

- Microsoft Edge 102 以上
- Google Chrome 102 以上
- Firefox 101.0 以上
- Safari 15.0 以上

@mdn/browser-compat-data

- ブラウザ機能の互換性情報を集約しているOSSリポジトリ
- `browser` とあるが Node.js や Deno も含まれる
- MDN Web Docs で表示されているデータの大本



サポートバージョン決め打ちの弊害

- サービス提供側視点では「動作確認してるのはコレです」と言いやすいが…
- ブラウザ更新はセキュリティ向上の側面も大きい
  - バージョン固定 = 更新しない理由になってしまう
  - 特別な理由が無ければ、サービス側がそれを妨げないほうが望ましい
- 開発者は「サポートバージョン」に縛られる
  - 更改しない限り、ネイティブで利用可能な機能が増えることはない
  - ビジネス的な事情が絡み頻繁には変更できないケースも多い
- 代表的な例としてよく言われていたのがIEサポートの有無

Transpile および Polyfill という解決策

- さまざまな事情によりバージョンを上げられない
  またはレガシーブラウザを切り捨てられない悩める人類が編み出した手法
- 開発時はモダンな機能を利用してコードを書くが、
  レガシーブラウザでも動作する形に変換(Transpile)する、
  あるいは不足しているAPIを追加で定義(Polyfill)する
- Babel が有名。TypeScript も target 指定が可能
- 「TypeScript はとりあえず target: "es5" ってしとけばええんやろ？」

ES5互換を実現できるなら全部解決するのでは？

本質的なレガシー互換の必要性を考える

"The State of ES5 on the Web"
https://philipwalton.com/articles/the-state-of-es5-on-the-web/

本質的なレガシー互換の必要性を考える

内容を一部抜粋すると…

- `console.log([1, 2, 3].at(-1));` というコードをES5に対応させると、
  31 Byte → 11,217 Byte まで増える
- CrUX (Chrome UX レポート) Pupularity Ranking の Top 10,000 サイトを調べた結果、
  68% のサイトで ES6 以降のコードと
  ES6 以降のコードを ES5 に Transpile したコードが混在している
  (具体的には async/await のコードで調査した模様)
- 世の中の多くのアクセスを占めるサイトにおいてこの状況だが、
  これによって「動かない！」といった大きな問題にはなっていないのが実情
- あなたの企業も同様にES5をサポートしなくても影響を受けない可能性が高い

Transpile や Polyfill は万能ではない

- サイズが肥大化しやすく、結果としてパフォーマンスに影響も与えうる
- そもそも Transpile や Polyfill で実現できない機能もある
  - WebAuthn
  - WebRTC
  - File System API
  - …
- サービスの性質にもよるが、多くのケースにおいて
  サポートバージョンを無視し続けることは難しい

サポートバージョンを最新のみにすれば全部解決する？

利用推奨環境を「メジャーブラウザの最新バージョン」としているケースも多い

e.g.) サイボウズのクラウドサービスの動作環境

→ 最新バージョンでOKなら、全ブラウザがサポートすれば全部使える！？

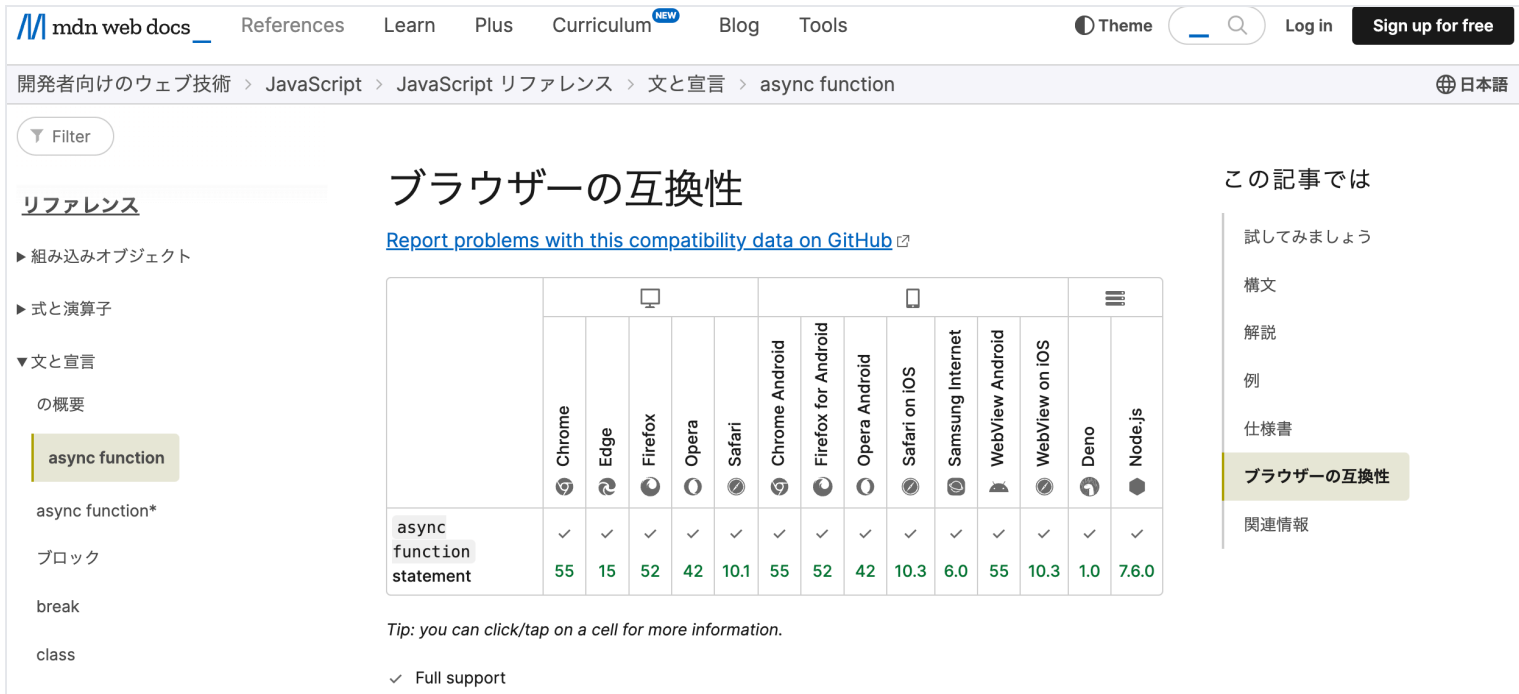
サポートバージョンを最新のみにすれば全部解決する？

残念ながら「使ってよい技術」の判断は不要にならない

- サポートが最新バージョンであっても、
  すべてのユーザーがすぐに最新化するわけではない
- ブラウザに新たに追加された機能をすぐに使ってリリースすると
  「動かない」という声が出る可能性はもちろんある
- 現実としては、全ブラウザの最新バージョンで利用可能となった機能でも、
  実際に利用して問題ないと判断するための許容期間は決めなければいけない

browser-compat-data から「期間」の判断は大変

- サポートバージョンは列挙されているが、
  「世に出てからどのぐらい経ったか」は自分で調べる必要がある



どうする？

Baseline

- Google I/O 2023 で発表された「ブラウザサポート状況の考え方」
- 現在は W3C WebDX Community Group によって定義されている
- 個々の機能を 3 つのステージに分類し互換性レベルを定義する
- 次のブラウザを対象とする
  - Chrome (PC, Android)
  - Edge
  - Firefox (PC, Android)
  - Safari (macOS, iOS)
- 厳密な定義は以下に存在する

  https://github.com/web-platform-dx/web-features/blob/main/docs/baseline.md

Limited availability

- 対象ブラウザのうち一部でのみ利用可能な機能
- experimentalフラグが必要なケースなども含む

Newly available

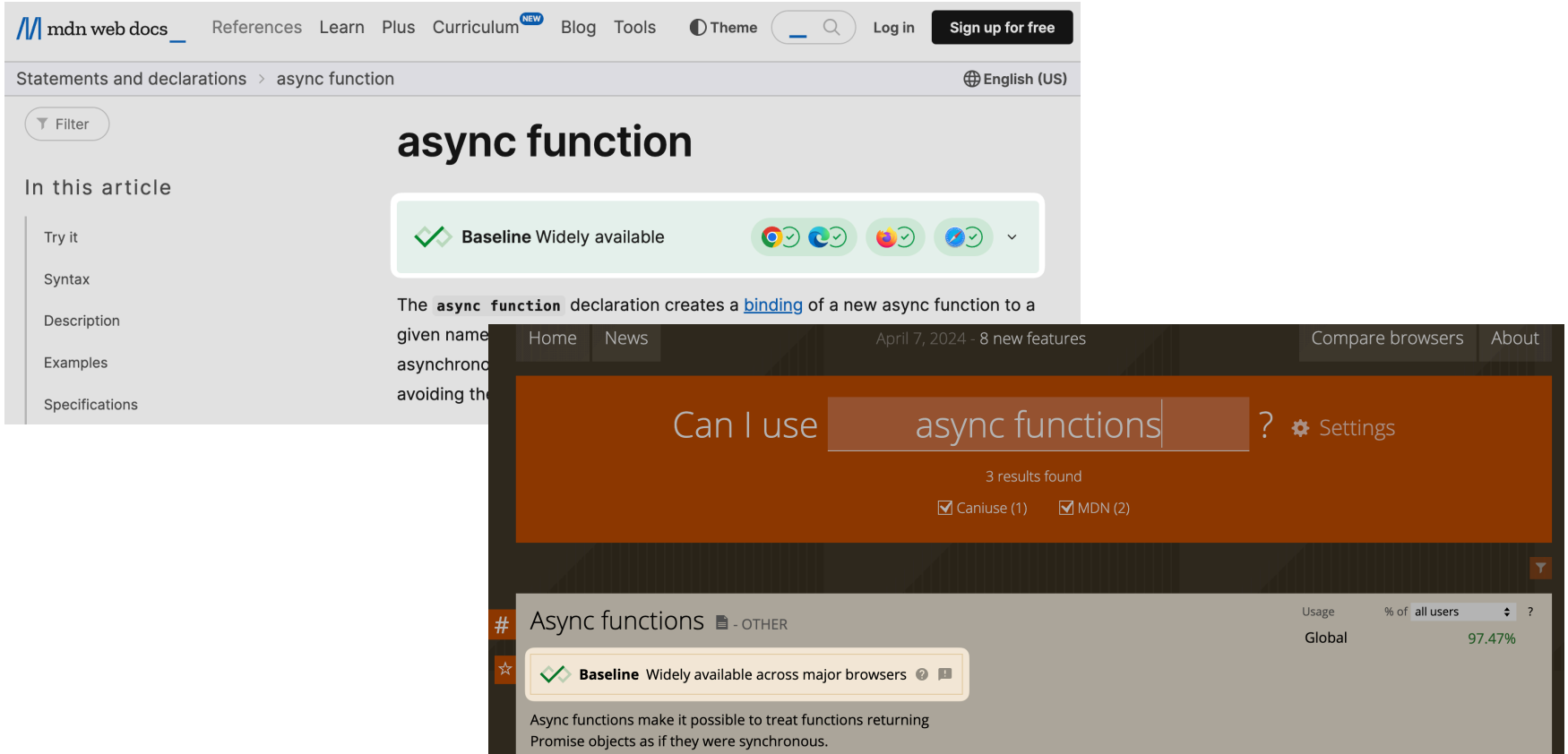
- 対象ブラウザすべての最新安定版で利用可能となった機能
- experimentalフラグが必要なケースなども含む

Widely available

- Newly available になってから**30ヶ月経過した**機能
- 「ほとんどの機能において、30ヶ月未満で95%以上のユーザーが利用可能となる」という調査結果に基づく

Baseline ステータスの簡単な確認方法

MDN Web Docs や Can I Use などでは Baseline のバッジでステータスを確認できる



web-features

https://github.com/web-platform-dx/web-features

- 正確な Baseline の対象機能とステータスは
  web-platform-dx/web-features リポジトリで管理されている
- ブラウザを含む Web Platform の機能の共有カタログを構築する取り組み

誕生の経緯

https://github.com/web-platform-dx/web-features/blob/main/2022-backgrounder.md

- Web Platform の機能リストの情報源がさまざまな箇所に散ってしまっている
  - MDN Web Docs, Can I Use, State of JS, 各種プロポーザル, Polyfill.io, Bugzilla, …
- それぞれで粒度も異なり、情報源を組み合わせて把握することが難しくなっている
- ウェブコミュニティ全体で正しく同じことを話し合い、
  相互運用性の共通理解を高めるために共通の機能リストに価値があり必要

Web Platform Status

- https://webstatus.dev/
- Baseline に関連する情報で検索/一覧表示できるダッシュボード
- 「いま Newly Available な機能って何があるんだろ？」とかを一気に確認できる

将来的に Baseline で検討されるかもしれないもの

将来的に新たなステータスがカバーされる可能性があり、
以下が検討対象候補として挙げられている
https://github.com/web-platform-dx/web-features/blob/main/docs/baseline.md

- Upcoming / まだ全ブラウザで利用不可だが、近日出荷予定
- Progressive enhancement safe / プログレッシブエンハンスメントのサポート
- Developer feedback requested / 開発者のフィードバックを募集中
- Buggy / バグが含まれる
- Support in assistive technology that is not built into browsers. / 支援技術サポート
- Obsolete/deprecated/legacy/etc. / レガシー, 廃止, 非推奨など
- Having high-quality polyfills available / 高品質の Polyfill の有無

共通言語としての「Baseline」

- Baseline という言葉だけで説明できるのは大きな利点
- web-features の存在により、対象機能の認識ズレも起こりづらい
- Widely available が安心して使える一つのラインとして考えられる
- 現実的な話としては、個人が考えた利用可能かの根拠ではなく
  「すでに世の中にある概念」として引用できるため、説得力を持たせやすい
- 使っていきましょう！