

textlint editor - ブラウザでも動く Privacy Firstの文章校正ツールを作る話

自己紹介

- Name : **azu**
- Twitter : @azu_re
- Website: [Web scratch](#), [JSer.info](#)



テーマ

- **textlint**
- Privacy Firstな校正ツールを作る
- サーバにデータを送らずに、ローカルで文章のチェックをする

textlintとは?

textlint

- `textlint` JavaScriptで書かれた文章のLintツール
 - ESLintの文章版
- Markdown、Re:View、HTMLなど文章構造をパースしてからチェックする
 - 一般的なスペルチェッカーは構造を見ないので誤検知する
- 200弱ぐらいのルールがある
 - [Collection of textlint rule](#)・[textlint/textlint Wiki](#)
 - 日本語、英語、言語に依存しないルールなど
 - [textlint入門 \(全11回\)](#) - プログラミングならドットインストール

1 | ✓ # textlintのデモ {#textlint-demo}

2

3 | 寿司は食べられる！

4

5 | お刺身が食べれない。

⊗ README.md 2 of 3 problems

ら抜き言葉を使用しています。 (ja-technical-writing/no-dropping-the-ra) textlint(ja-technical-writing/no-dropping-the-ra)

6

7 | これは問題はある文章だ。

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL

Filter. E.g.: text, **/*.ts, !**/node_modules/**

✓ ⓘ README.md source/basic/object 3

⊗ Disallow to use " ! ". (ja-technical-writing/no-exclamation-question-mark) textlint(ja-technical-writing/no-exclamation-question-mark) [3, 9]

⊗ ら抜き言葉を使用しています。 (ja-technical-writing/no-dropping-the-ra) textlint(ja-technical-writing/no-dropping-the-ra) [5, 7]

⊗ 一文に二回以上利用されている助詞 "は" が見つかりました。 (ja-technical-writing/no-doubled-joshi) textlint(ja-technical-writing/no-doubled-joshi) [7, 6]

textlintのユースケース

- Angular、React、Vue、Nuxt.js, Next.js、Gatsbyの公式ドキュメントの翻訳など
- 書籍: JavaScript Primer、Pythonクローリング&スクレイピング、RustPrimer、SurviveJS - Webpack
- VuePress、Cypress、日本マイクロソフト Azure Identity、OWASP Cheat Sheet Series

文章に対するCI

- 文章に対するContinuous Integration(CI)を持ち込む
 - Continuous Writingするためのツール
 - [asciidwango/js-primer: JavaScript Primer - 迷わないための入門書](#)とかで使ってる
 - 長期間書いていると人間のほうが変わってるので質が一定に保てない
 - jsprimerは5年ぐらい書いている
- そのためどちらかというと技術的な用途がメイン

Easy的な使い方

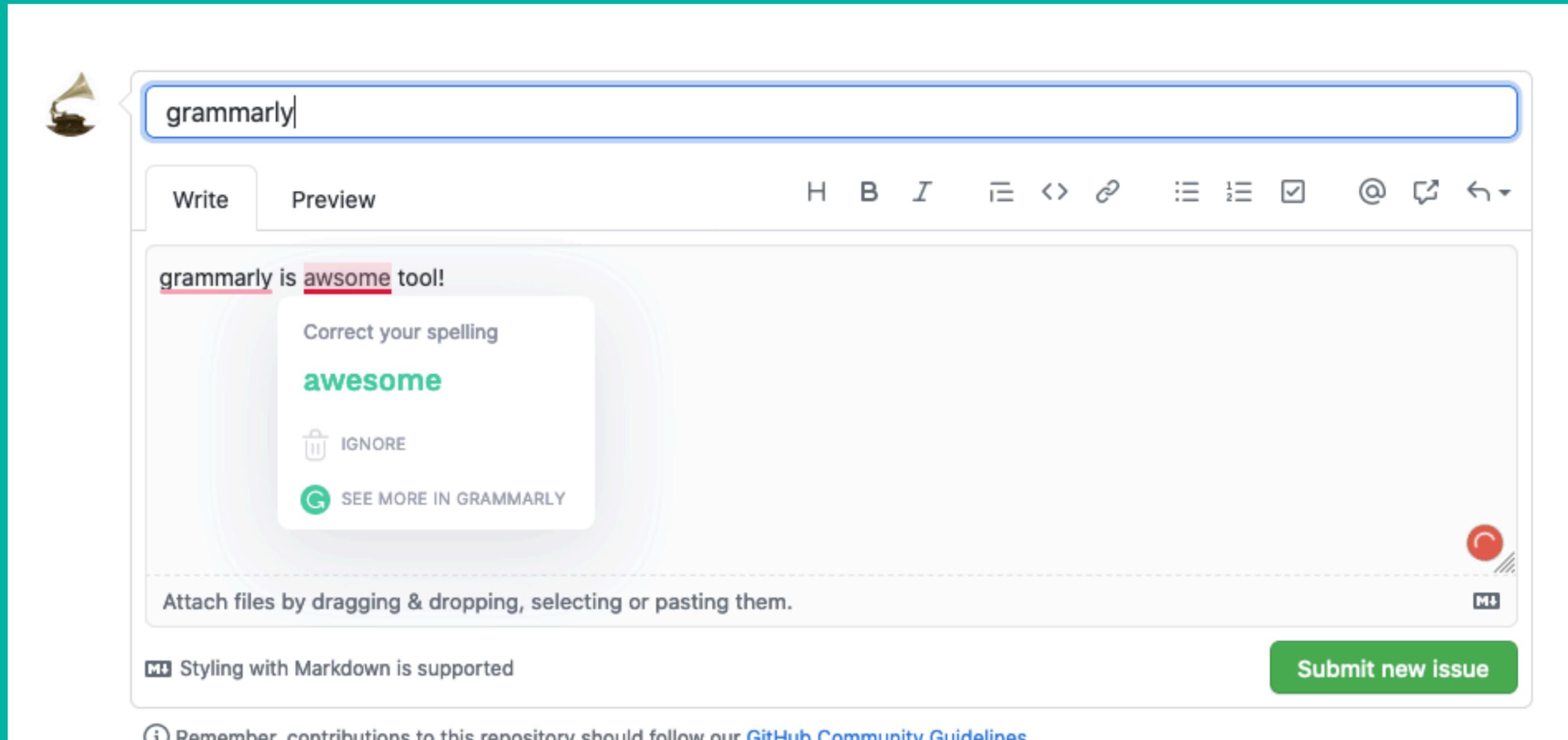
- 今までtextlintをラップしたものはあった
- 文書校正ツール textlint の Chrome 拡張を作った - もなでいっく
- テキスト校正くん - Visual Studio Marketplace

textlint/editor

目的

- Privacy Firstなgrammarlyを作る
- grammarlyのブラウザ拡張が便利だった
- → 今回作ったもの: [textlint/editor: textlint editor project](https://github.com/textlint/textlint-editor).

grammarly



The screenshot shows a GitHub issue creation interface. At the top left is the GitHub logo. A search bar contains the text "grammarly". Below the search bar are two tabs: "Write" (active) and "Preview". A rich text editor toolbar includes icons for heading (H), bold (B), italic (I), list (≡), code (<>), link (🔗), table (📊), table (📋), checkmark (☑), mention (@), share (🗨), and undo (↶). The main text area contains "grammarly is awsome tool!". A Grammarly tooltip is displayed over the word "awsome", showing "Correct your spelling", the correction "awesome" in green, and options to "IGNORE" or "SEE MORE IN GRAMMARLY". Below the text area is a dashed line and the instruction "Attach files by dragging & dropping, selecting or pasting them." with a file upload icon (M+). At the bottom left, a note says "M+ Styling with Markdown is supported". At the bottom right is a green "Submit new issue" button. At the very bottom, a footer note reads "Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)."

grammarlyの問題

- textareaに入力した文字が全てサーバに送られる仕組み

The image shows two side-by-side screenshots illustrating a security issue. The left screenshot is from Burp Suite Community Edition v2020.7, showing a list of intercepted HTTP messages. The right screenshot is from a browser window showing a text area editor with the text 'テスト文章' and a Grammarly notification.

Filter: Showing all items	Direction	Edited	Length	Comment	TLS	Time	Listener port	Web
ammarly.com/freews	← To client		90		✓	14:43:17 28...	8080	1
ammarly.com/freews	← To client		144		✓	14:43:17 28...	8080	1
ammarly.com/freews	← To client		144		✓	14:43:17 28...	8080	1
ammarly.com/freews	← To client		200		✓	14:43:17 28...	8080	1
ammarly.com/freews	← To client		39		✓	14:43:17 28...	8080	1
ammarly.com/freews	← To client		50		✓	14:43:18 28...	8080	1
ammarly.com/freews	← To client		144		✓	14:43:18 28...	8080	1
ammarly.com/freews	← To client		200		✓	14:43:18 28...	8080	1
ammarly.com/freews	→ To server		93		✓	14:43:30 28...	8080	1
ammarly.com/freews	→ To server		147		✓	14:43:30 28...	8080	1
ammarly.com/freews	← To client		90		✓	14:43:30 28...	8080	1
ammarly.com/freews	← To client		39		✓	14:43:30 28...	8080	1
ammarly.com/freews	← To client		146		✓	14:43:30 28...	8080	1
ammarly.com/freews	← To client		50		✓	14:43:31 28...	8080	1

Message

Raw Hex

```
1 {"ch":["+0:0:テスト文章:0"],"rev":21,"action":"submit_ot","id":23,"timer":{"client_clock":165671,"id":"..."}}
```

Textarea - Froala

Grammarly for Chrom

Grammarly

テスト文章

Textarea Editor

Try it yourself:

A: ¶: +:

テスト文章

Powered by Froala

Grammarly is Now Active

Want to see how it works?

TAKE A QUICK TOUR NO, THANKS

Edit in JSFiddle

HTML

```
<textarea id="froala-editor">Initialize the Froala WYSIWYG HTML Editor on a tex
```

JAVASCRIPT

類似するツール

- [grammarly](#)
- [Microsoft エディター](#)
- どちらもオフラインでは動かない、サーバに通信してその結果を表示する

サーバに入力文字を送ることによって起きる問題

- DeepLでNetflix?の字幕を学習してる問題
- Baisu IME ログ情報送信問題

DeepLの学習

- -Now, を翻訳するとNetflixっぽい字幕がでてくる
- DeepL利用規約(無料版)は入力データは全てDeepLのものとなる
 - 無料版のデータは学習に使うため保持することが記載されていた
- DeepL翻訳で機密は保持される？セキュリティは？ | ブログ | 株式会社ヒューマンサイエンス



Privacy Firstな校正ツールを作る

- **textlint**はJavaScriptで書かれている
- 基本的にルールもJavaScriptで書かれている
- そのため、オフラインでも動作する
- サーバにデータを送る必要がない

textlintをブラウザで動かす

- 目標: `grammarly`のtextlint版を作る
- textlintは基本的にはNode.jsで動かしている
- ブラウザで動かすには依存を色々解決するものが立ち

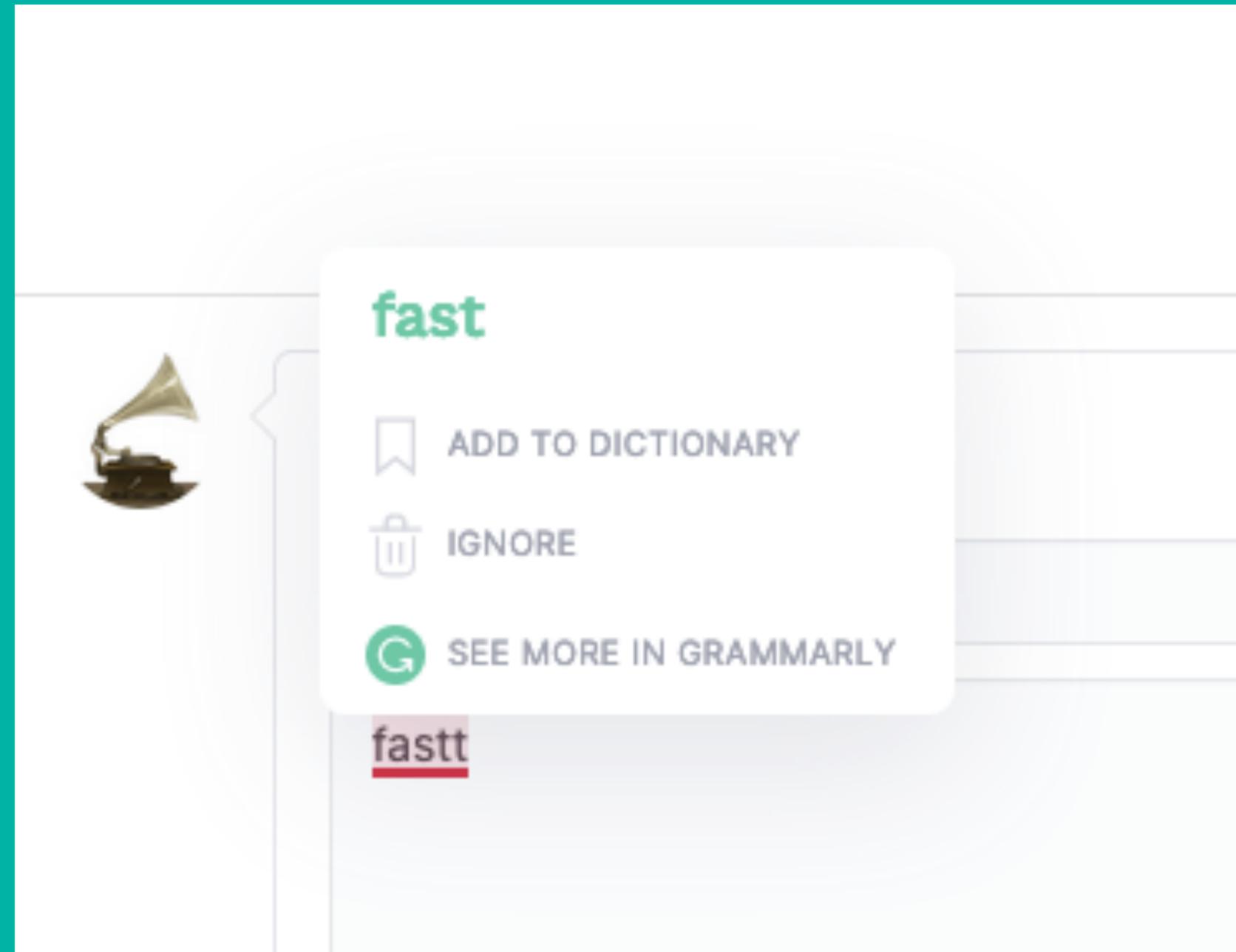
5日でプロトタイピング

- ここから本題 20min
- 5日でtextlint editorを作る
- ブラウザでtextlintを動かす
- ブラウザ拡張であらゆるサイトでtextlintでのチェックを使えるようにする

1日目

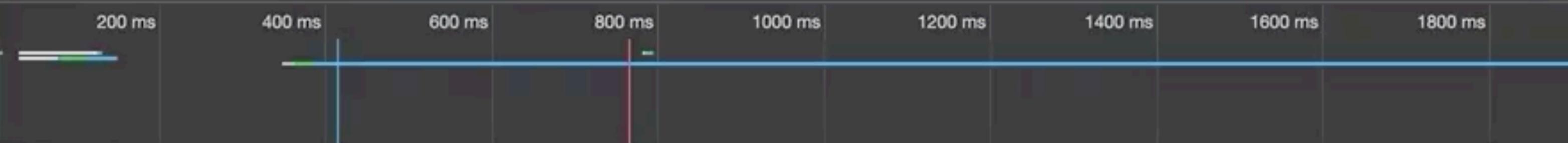
1日目

- grammery的なUIを作るプロトタイピング
- あらゆるサイトで動かないと行けないのでWeb Componentsでコンポーネント作成
- → textareaに重ねるようにして要素を置くことで、一部に下線が出るようにするコンポーネント作る



DEMO: [https://twitter.com/azu_re/
status/1286309148758482945](https://twitter.com/azu_re/status/1286309148758482945)

0123456789!
ABCDEFGH? 日本語もいけ
るかな?



Name	Method	Status	Protocol	Domain	Type	Initiator
hid-dot-on	GET	200	h2	cdn.jsdelivr.net	xhr	VM42226:1

form お刺身が食べれない! 寿司は食べられる。東京特許許可局は難しい漢字だ。

linter

要素 コンソール ソース ネットワーク パフォーマンス メモリ アプリケーション セキュリティ 監査

```
... <body> == $0
  <main class="main">
    <text-checker-element>...</text-checker-element>
    <textarea class="textarea">form お刺身が食べれない! 寿司は食べられる。東京特許許可局は難しい漢字だ。</textarea>
  </main>
  <aside class="side">...</aside>
  <script src="/public.8e0568f4.js"></script>
  <textchecker-popup-element>...</textchecker-popup-element>
  <div id="input-textarea-caret-position-mirror-div" style="white-space: pre-wrap:
html body main.main text-checker-element
```

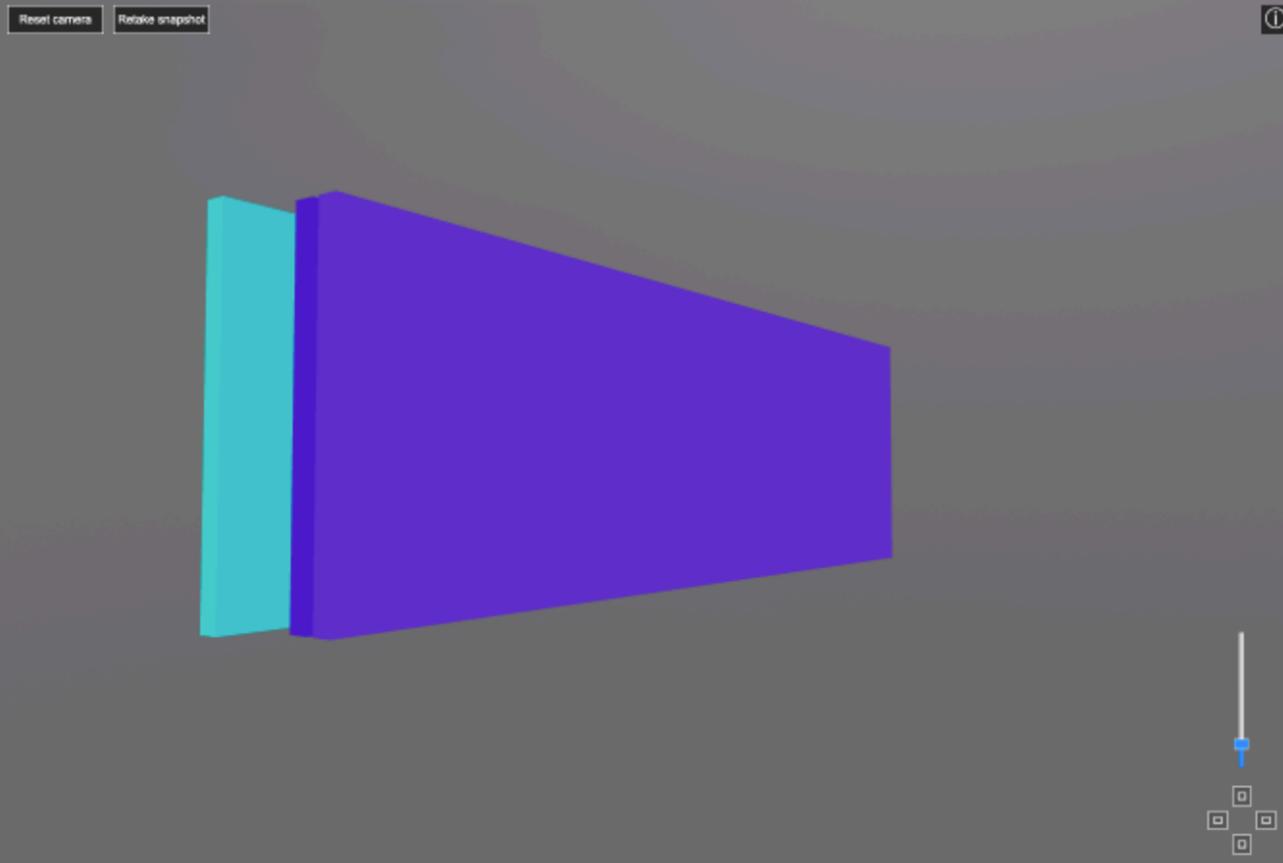
スタイル 計算済み イベント リスナー >>

フィルター :hov .cls +

```
element.style {
}
body {
  display: block;
  margin: 8px;
}
```

3D ビュー x コンソール 新着情報

- Z インデックス DOM
- すべて表示
 - スタック コンテキストのみを表示
 - ラベルの使用
 - 選択した要素の分離
 - 親を保持
 - 色の種類
 - 紫から白
 - 背景色
 - 親と同じ描画順序の要素を非表示にします。



2日目

2日目

- 作ったUIにtextlintを組み込んでまずは実証
- 既存のtextlint-browser-runnerを使ってintegration
- → あらゆるtextlintルールをどうbundleをどうするか考える
- → 動いた

textlintのbrowser版はなんでビルドが必要?

- textlintはすべてがpluggable
- textlintのルールを個人に合わせる仕組みが必要
- 同じ人でもルールは文章によって違う

2日目: textlintのbundleをどうするか?

- 方法
 - 全部入りのbundleを作る
 - 全部オンラインロード
 - ルールだけをパッケージするか

2日目: bundle prototype

- 全部入りのbundleを作る
 - @zeit/nccでbundle
 - → うごいた
 - webpackでも動くのは過去からわかってる
- 全部オンラインロード
 - PikaCDN, Skypackを使ってルールを読み込む
 - → ローカルでのビルドが不要というのが大きなメリット

textlintのbrowser版はなんでビルドが必要?

- Skypackとの格闘
- CDNでのビルド済みファイルの利用
 - mizchi/unirollでのビルド
 - importしたモジュールをCDNからロードするようにしてビルドする
 - Rename Pika to skypack by azu · Pull Request #10 · mizchi/uniroll
 - Support "module" field by azu · Pull Request #690 · textlint/textlint
- いくつか対応してunirollでtextlintが動いた！
 - https://twitter.com/azu_re/status/1286657627032678400

3日目

npm CDNの問題

- skypackでは`require("pkg/file")`がtranspileされない仕様
 - pkgモジュールの相対パスの`file.js`を読み込む仕組み
- Lookup a Package File
 - そのため direct requireしてるライブラリがどこかにあると動かない
 - 回避するためにはライブラリガワがbundleファイルを配布してくれないとできない
 - feat: use microbundle for distribution by azu · Pull Request #15 · textlint/textlint-util-to-string
- → 現実的にあらゆるルールで担保するのは難しい
 - 断念

webpack as a service

- nccやwebpackでtextlintはビルドできる
- → webpack as a serviceを考える
 - codesandboxを使えば、webpackでビルドしたファイルをブラウザだけで取得でき
そうな予感がした
 - どちらにしてもだいたい人は配布されたファイルを使うだけでいい
 - 高度な使い方をする人は自分のtextlintrcを持っているはず → node環境がある
 - → つまり textlint + textlintrcからbundleを作成できるコンパイラーを書けば解決できそう

@textlint/compiler

- textlintのルールは仕組み的にdynamic requireで解決している
(textlintが依存を知らないため)
 - ESLintなど多くのルールをもつツールはだいたい同じ仕組み
 - dllみたいなもの = ルール
- @textlint/compilerを作成する
 - webpackに優しいコードを出力する

@textlint/compiler

- このルールをcompilerがコンパイルしやすいように、静的な形に整形するlinkerを書く
- .textlintrc(設定ファイル) → staticなrequireに変換するlinker
 - 実体はconfig loader + code generator
 - https://twitter.com/azu_re/status/1286952957754273793
- コンパイルの処理自体はwebpackに生成したコードを投げる

4日目

リアルタイムLint

- ブラウザで動くなら入力ごとのLintしたい
- web workerの対応
 - 入力文字における処理はWebWorkerなど別スレッドで行うのが基本
 - コメント入力はレイテンシーに弱い 許されるのは 5ms以内ぐらい
 - webworkerに対応したtextlintを作成する
 - @textlint/kernel(コア)はpure jsで書かれているので、postmessageのラッパーを書くだけで解決
 - target: selfで解決

kuromoji.jsの辞書解決

- textlintのルールでは一部のルールがkuromoji.jsでの形態素解析を使ってる
- この辞書が圧縮して20mbぐらいある(無圧縮は100mb)
- この辞書を毎回ロードしててる大変
- worker内でキャッシュ、同時に取得した時に1回の取得にまとめる必要がある

kuromoji.jsの辞書ハック

```
// InMemory Cache
const dictionaryDeferred = new Deferred();
const urlMap = new Map();
BrowserDictionaryLoader.prototype.loadArrayBuffer = async function (url, callback) {
  // https://github.com/takuyaa/kuromoji.js/issues/37
  const fixedURL = url.replace("https://", "https://");
  const cachedDictBuffer = await dictionaryStorage.get(fixedURL);
  if (cachedDictBuffer) {
    // console.log("return cache", cachedDictBuffer);
    return callback(null, cachedDictBuffer);
  }
  // Suppress multiple request to same url at same time
  if (urlMap.has(fixedURL)) {
    return urlMap.get(fixedURL).promise.then(result => {
      callback(null, result);
    }).catch(error => {
      callback(error);
    });
  }
  const deferred = new Deferred();
  urlMap.set(fixedURL, deferred);
  fetch(fixedURL).then(function (response) {
    if (!response.ok){
      return callback(response.statusText, null);
    }
  }
  response.arrayBuffer().then(function (arraybuffer) {
    var gz = new zlib.Zlib.Gunzip(new Uint8Array(arraybuffer));
    var typed_array = gz.decompress();
    return dictionaryStorage.set(fixedURL, typed_array.buffer).then(() => {
      // console.log("cached", fixedURL);
      deferred.resolve(typed_array.buffer);
      callback(null, typed_array.buffer);
    });
  });
}).catch(function (exception) {
  deferred.reject(exception);
  callback(exception, null);
});
};
```

辞書ハック

- prototype hackをしてる
- ちゃんと解決するにはkuromoji.js自体をイジる必要がありそう
- dictionary loaderを外から使えるようにしないとダメそう
- forkしかないのかも

ここまで

- ブラウザでIntegrations
 - 1日目に作ったUI
 - 2日目は仮説検証 → 断念
 - 3-4日目に@textlint/compilerで生成したtextlint bundleを読み込む
- 統合して動かすデモ

お刺身が食べれない!寿司は食べられる。東京特許許可局は難しい漢字だ。

ら抜き言葉を使用しています。

linted

5日目

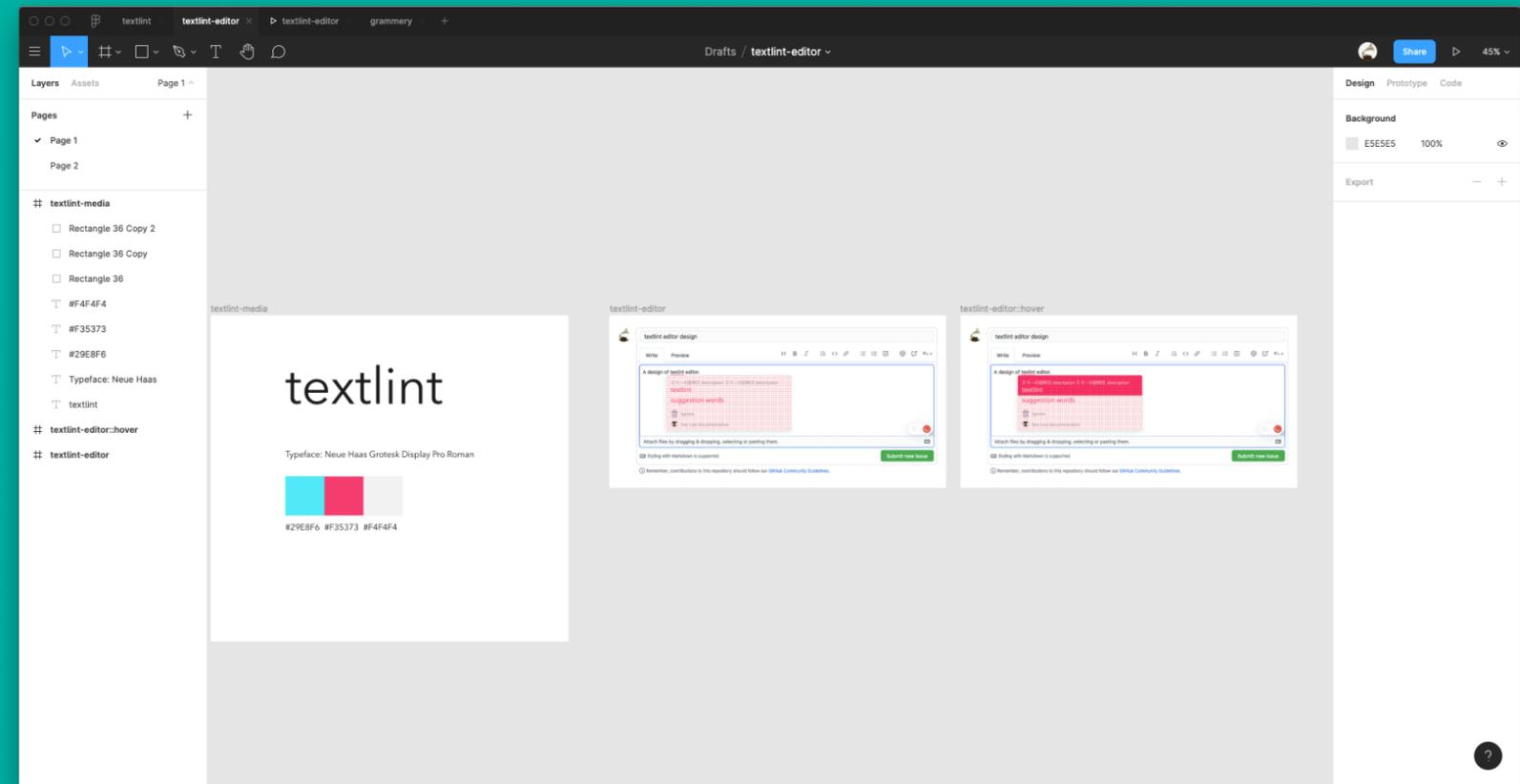
- ブラウザ拡張を作る
- web extension firefox + chromeとかに対応する
- WebExtension Toolboxを採用
- メンテにちょっと不安感あるけど、configを減らして運用でカバー ejectable

Chrome拡張での検証

- [x] WebWorkerをBackground Pageで動かす
- [x] Content Scriptsで1日目のUIをあらゆるサイトにInject
 - Content ScriptsでWeb Componentがうごかない！！！！
 - これ絶対Chromeおかしい！！！！
 - [390807 - Content scripts can't define custom elements - chromium](#)
 - `import '@webcomponents/custom-elements'`を使うことで回避
 - WebComponentをWebComponentのpolyfillで動かすことで回避
- UI → Content Scripts → Background Pages → WebWorker(textlit)
 - Lintに成功

デザインを書く

- デザインを描いていく
- figmaでいくかデザインのパターンを書く
- <https://www.figma.com/file/9kRm0Cr869zbdACytRE74R/textlint-editor?node-id=0%3A1>





textlint editor design

Write

Preview

H B I ≡ <> 🔗 ☰ ☷ ☑ @ 🗨️ ↩️

A design of textlint editor.

エラーの説明文 description エラーの説明文 description

✎ Fix it!

🗑️ Ignore

📖 See rule documentation

Attach files by dragging & dropping, selecting or pasting them.



Styling with Markdown is supported

Submit new issue

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

デザインを実装する

- Figmaで書いたデザインを実装する
- `textchecker-element`
- ひとまず完成！

textlint editor

<https://textlint-editor.netlify.app/>

デモ: [https://twitter.com/azu_re/
status/1288038759192174593](https://twitter.com/azu_re/status/1288038759192174593)

お刺身が食べれない!

今すぐ行動することができる。

まとめ

- UIはWebComponents
- コアは最初からPure JavaScriptで書いておく
 - ちょっとしたNodeライブラリならwebpackでなんとかできる
- skypackのCDNはRollup系なのできれいなビルドができる
 - キレイじゃないライブラリは使えない
 - Node.js exportsとかがこれからくるので、きれいな方法がまだ定義されていない
- textlint + ルールを一つにbundleしてオフライン校正ツールが動いた!

これから

- Collaboratorを募集！
 - Repository: [textlint/editor: textlint editor project.](#)
 - Gitter: <https://gitter.im/textlint-ja/textlint-ja>
- ブラウザで動けば大体のところ動く
 - サイトに組み込みしやすい、サーバ側に遅れたデータを管理しなくていい単語
 - → 同じ仕組みでウェブサイトにも組み込みができる

これから

- frontend
 - suggest対応
 - ignore対応
 - see document対応
- compiler
 - codesandboxでコンパイルしたものを配布するテンプレート
- performance
 - 今はlintが10msぐらい、文字数に応じて線形的に処理が増える
 - ちゃんと考えるなら差分処理が必要だけど、文章にはコンテキストがあるので制限がある
 - WebWorkerのおかげでUIにはlatencyは少ないので、1sぐらいならOK
 - 現実的にtextareで10kbレベルの文章を書く人は少なく、ファイルを分けるからなんとかなる
 - DOMの最適化が必要。表示してるエリア飲み情報を表示するなどの最適化が必須
- 拡張
 - Chrome拡張にtextlint bundleを更新する仕組みを作る
 - textlint.jsをダウンロードしてChrome拡張で使うものを切り替え → Workerを再起動
 - storeの公開がめんどくさいのでどうする?
- ルール: まだ動かないルールもあるはず

おわり

- 現在はprototypeができた段階、ちゃんと使えるものにする必要がある
- 興味がある人はCollaboratorとして参加してください。
- Summary Issue: <https://github.com/textlint/editor/issues/4>
- [textlint/editor](https://github.com/textlint/editor): textlint editor project.