2020年度未踏ジュニア提案書

提案するプロジェクトのタイトル

Visible ― 自動修正可能なWebアクセシビリティーテストツール

提案者に関する事項

メインクリエータ（代表者）の氏名	五十嵐 涼
グループの場合、メンバーの氏名	

以下の入力欄は必要に応じてサイズを変更できます。図や表の挿入も推奨します。

1. 提案するプロジェクトの簡単な説明（200字以内）
3. の提案内容に記載される内容を、簡潔にまとめてください。

例) 「暗記クッキー」は、海外に住む日本人の子どもや、日本語を学習したい外国人をターゲットにした、WEBベースの漢字クイズで頑張って覚えた漢字がクッキーになって届くことでモチベーション継続を支援するシステムです。漢字クイズの結果を元に、レーザーカッターを利用してクッキーに漢字を刻印し、魔法のようにユーザーに届けることが出来ます。

「Visible」はNode.jsで開発されるオープンソースのWebアクセシビリティーテストツールです。WebサイトのURLやソースコードからアクセシビリティー上の問題点を検出するほか、アルゴリズムや機械学習プラットフォームを有効活用して修正を提案します。

2. 開発費の使用計画

開発に関わる費用が合計 50 万円まで補助されます。現時点で未定の項目があっても構いません。支出項目については採択後PMとの相談により決定します。
例:
・××デバイスの購入 8,000 円、◇◇ 機能の実装のため
・△△ソフトウエア 12,000 円、〇〇 を作成するため
・☆☆デジタルデータ 20,000 円、□□で使用するため
・使途未定（開発で必要が生じた場合の予備）: 240,000円

- Google Cloud Platform Vision AI 従量課金
  プロバイダー（後述）に利用します
- Google Cloud Platform Cloud Natural Language 従量課金
  プロバイダー（後述）に利用します
- VPS契約費 20,000円
  Web版APIサーバーのデプロイに利用します。
- S3契約費 従量課金
  Web版で画像をアップロードするために利用します
- ZEIT now契約費 従量課金
  Web版デプロイに利用します。
- ドメイン契約費 3,000円
  Web版デプロイに利用します。
- 使途未定（要計算）

3. 提案内容

[提案内容を書くときのアドバイス]
フォーマットは自由です。図表や画像も使用できます。ページ数にも制限はありません。以下の項目は一例です。

どんなもの？
・おじいちゃん、おばあちゃんに説明すると想定してまとめてみましょう
・完成したときのイメージを図や画像で表現してみましょう

誰のどんな問題を解決するもの？　これができると誰がうれしい？
・誰がどんなシーンで使いますか
・必ずしもたくさんの人の役に立つ必要はありません
・役には立たないかもしれないけれど、おもしろいというのもありです

既存のもの（似ている製品やサービスなど）は何がある？
・可能であれば海外の事例も調べてみましょう。日本語よりたくさんの情報が見つかります

あなた独自のアイデアは？
・既存のものと比べて、提案のユニークなところはどこでしょう
・改良した点を書いてみましょう

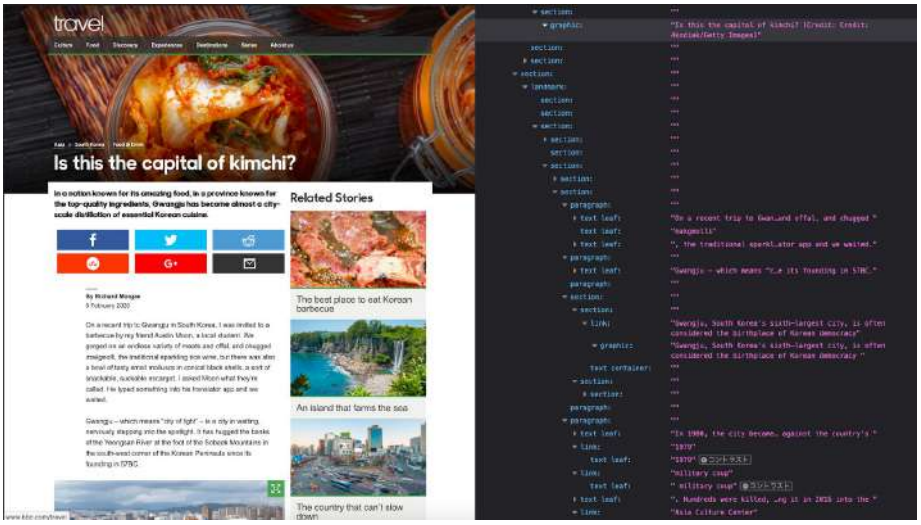
Webの課題

Web開発は「見た目」が重視されがちですが、本来Webは誰でも自由に文書を公開できるプラットフォームとして設計されていました。言い換えれば「見た目」はさほど重要ではなく、障害の有無に関わらず万人が問題なくアクセスできること、すなわち**アクセシビリティー**の方が尊重されるべきであるということです。

アクセシビリティーの対応が必要なWeb利用者の一例として、スクリーンリーダーの利用者が挙げられます。スクリーンリーダーとは画面上に表示される文字を読むことが困難な人が利用するソフトウェアで、文字の代わりに音声で内容を聴きます。

スクリーンリーダーの利用者でも問題なく利用できるWebサイトを開発するには、アクセシビリティーの観点で慎重な対応が必要です。

例えば、私たちはWebサイトをマークアップしていて、何かを強調したいと思ったとき、文字を大きくしたり、色を変えたりといった装飾で閲覧者が興味を引くように仕向けるしれませんが、スクリーンリーダーでは当然文字の大きさや色は伝えられません。

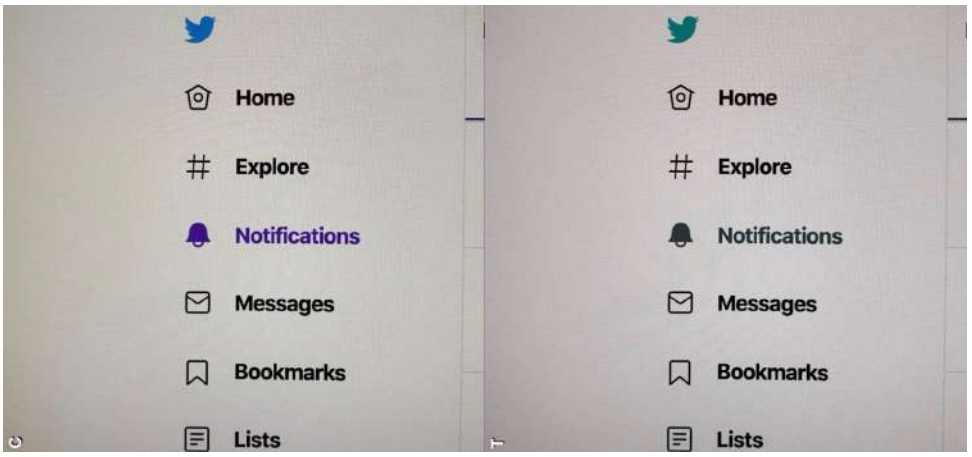


私たちが見るWebコンテンツ（左）と、スクリーンリーダーが読み上げるWebコンテンツ（右）の比較

そこで、そのような用途のためにWeb標準が提供しているセマンティック要素を利用します。例えば、何かを強調したいと思ったら*em*、ある文字列がその文章の見出しであるなら*h1*～*h6*といったコードを追加して、装飾がなくても同じだけの情報を利用者に伝えられるようする必要があります。

さらに、一部のWebコンテンツはそのままではスクリーンリーダーに伝えるのが不可能な場合があるので、その場合は代替となる情報を提示します。例えば画像は音声として伝えるのは難しいので、キャプションを指定することでこれを解決します。

Webを利用する上で問題になり得る障害は視覚障害だけではありません。別の例として色覚特性―色の見え方が普通と異なること―が挙げられます。



テーマカラーを紫に設定した状態のTwitter（左）と、赤の色弱の人が見る状態を再現した画像（右）。
コントラストが十分でないため、どれが選択されているのかがわかりません。

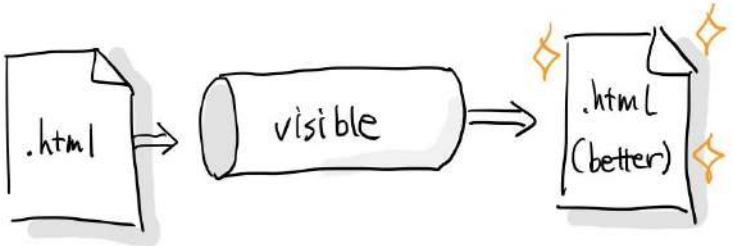
アクセシビリティーに対応するためには、色を配色する際にこういった利用者を想定し、カラーコントラストを計算しなければなりません。私たちにとって美しいデザインだとしても、色覚特性のある利用者にはただの読みづらいコンテンツになってしまうからです。

これらの懸念点をWeb開発者が一つ一つ把握できれば良いのですが、残念ながらこれらの利用者のユースケースを考慮しながらWebサイトを開発するのは非常に難しく、私を含むWeb開発者の多くがセマンティックを忘れたり、キャプションを入れなかったり、コントラストが足りなかったりして、意図せずアクセシビリティーを欠いたWebサイトを開発してしまうことがあります。

もちろん、障害があるというだけの理由で特定の利用者を排除することはあってはなりません。

私が提案する解決策

上記の問題を解決するために私が提案するプロジェクトがVisibleです。VisibleはNode.jsで開発されるオープンソースのWebアクセシビリティーテストツールです。WebサイトのURLやソースコードからアクセシビリティー上の問題点を検出するほか、アルゴリズムや機械学習プラットフォームを有効活用して修正を提案します。



Visibleに入れると魔法のようにアクセシブルなWebサイトになる...予定！

Visibleを利用するのはWeb開発者です。上に述べたようなアクセシビリティー対応に多くの時間と労力を割いている開発者に、正しい「ガイド」を提供することで、Webをより多くの人に開かれた、良い場所にすることができるのではないかと考えています。

Visibleは迷えるWeb開発者の水先案内人

Visibleでは、アクセシビリティーに取り組むステップを2つに分けます。1つは**「検出」**すなわち問題を特定することで、2つは**「修正」**すなわち全ての利用者にとって利用しやすい代替を提案することです。

検出と修正の例

- **見出しや強調といったセマンティクスが無い要素**
  セマンティック要素が利用されていないのにも関わらずCSSがオーバーロードされている箇所を検出、スタイルに基づいて適切なセマンティック要素―太字であれば*em*, 文字サイズが大きければ*h*―に置き換えることで修正
- **スクリーンリーダーが読み上げられない画像**
  alt属性が無いimg要素を検出、画像からテキスト情報を作成してalt属性として与えることで修正
- **背景色によって文字が読みづらくなっている要素**
  colorとbackground-colorのコントラスト比をWCAG G18に基づいて計算し、4.5:1より少ないものを検出、background-colorの明るさを下げるアルゴリズムで修正

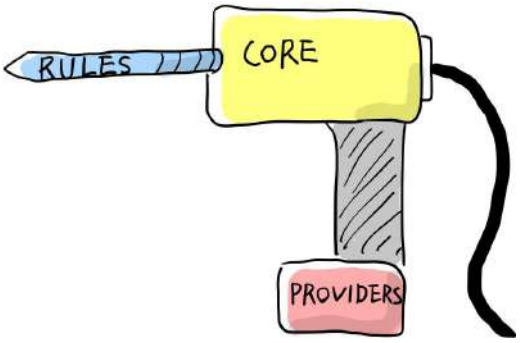
その他、WCAGやWAI-ARIAなどのガイドラインに準拠した100以上のルールの検出と修正を実装予定です。

仕組み

Visibleは以下の従属するコンポーネントから構成されます

1. **コア** ― ルールと設定からヘッドレスブラウザー（Puppeteerなど）でテストを実行するフレームワーク。

2. **ルール** — WCAGやWAI-ARIAなどのガイドラインに準拠する拡張可能なルール。問題を検出するコードと修正アルゴリズムはここに属します。
3. **プロバイダー** — ルールで検出した問題のうち、アルゴリズムで一意に解決できない問題に修正を提案する拡張可能なパッケージ。例えば「画像にキャプションが無い」という問題が報告された時、画像から文字列に変換するプロバイダーが代替テキストを作成し、それが修正として開発者に提案されます。Google Cloud Platformなどの機械学習プラットフォームを利用します。
4. **Webクライアント、CLI** — Visibleは上の3つのパッケージから構成されるNode.jsライブラリですが、インターフェイスとしてWebページやCLIから利用できるようにします。



Visibleはカスタマイズ可能で*future-proof*

2, 3に関してはプラグインのようになっていて、設定ファイルから利用者が独自に拡張することも可能です。3では例としてGCPを取り上げていますが、AWSなどの別の方法を用いることも可能です。

類似プロジェクトとの比較

	Visible	axe	ESLint
動的解析	✔	✔	❌
自動修正	✔	❌	✔

axe

類似する機能を持つプロジェクトとして、Deque Systemsが5年以上前から開発しているaxeがあります。Googleが開発している有名なWebスコアリングツールのLighthouseやChromeの開発者ツールも内部でaxeを使っています。axeは優れたフレームワークですが、いくつか問題点があります。

1. 問題を検出するのみで、修正を提案できない。
2. ブラウザをターゲットにコンパイルされたJSバンドルしか提供されておらず、ブラウザで別途実行する必要がある。
3. 最近のWebサイトはReactやVueなどのコンポーネントベースなフレームワークで開発されているが、これに対応していない。

特に1については、開発者は axe が問題を検出したらその問題について別途調査し、自分でパッチを当てなくてはなりません。また、3についてもモダンなWeb開発をする上では致命的です。Visibleは上の問題を次のように改善します。

1. **問題を検出し、かつ修正も提案**する
2. ヘッドレスブラウザを内蔵し、**URLと設定を指定するだけで実行**できる
3. ReactやVueも**コンポーネント単位でテスト可能**にする

他にも、axeよりもモダンなコードベースで保守性を高めたいと思っています。

ESLint

他にも、ESLint があります。ESLintはJavaScriptのリンター―コードの保守性を上げるためにコードの書き方にルールを設定するツール―で、その拡張性の高さからアクセシビリティーの問題点を検出できる eslint-plugin-jsx-a11y などのプラグインが開発されています。

しかし、ESLintはJavaScriptの構文から問題を検出している（すなわち静的解析）ため、例えば「ボタンの色が見づらい」などの問題はHTMLとスタイルの対応付けができず、検出できません。

Visibleはフレームワーク上で本物のブラウザーを動作させるため、実際に画面に表示される内容から問題点を検出することが出来ます。

4. あなたが自分の貴重な時間を使ってこのプロジェクトを実現したい理由 (任意)
なぜ、世界中の誰かではなく「あなた」がこのプロジェクトに時間を使うべきなのでしょうか？何か、原体験や自分にしかない強み、プロジェクトに対する思いがあれば書いてください。

初めて「アクセシビリティー」というコンセプトを知ったのは分散型ソーシャルネットワークプラットフォームMastodonのソースコードを読んでいた際、aria-labelというHTML属性を見たときです。それまで「障害のある人はパソコンなんて使わないだろう」と考えていた私にとって、Mastodonの開発者が尊重していた「スクリーンリーダーなどでWebを『見える』ようにする」という考えは衝撃的でした。

しかし、同時に多くのWebサイトがMastodonほどアクセシビリティーに関心がなく、せっかく使いやすいサイトであっても障害がある人には正しく情報が読み上げられないときもあると分かった時、この状況は改善しなくてはならないと感じました。

Webの策定機関であるW3Cによれば、全てのWebサイトは以下のように設計されていました。

> *Webはハードウェア、ソフトウェア、言語、文化、所在地、物理的/精神的能力にかかわらず、基本的にすべての人に向けて設計されています。*

本来、国家や出版社による検閲の妨害を受けることなく、誰でも自由に文書を公開できるプラットフォームとして設計されていたWebの開発者が、障害があるという理由だけで特定の人々を排除してはならないはずです。

後述しますが、私は中学1年生の冬から様々な原因が重なって学校に登校できていませんでした。中学、高校と年齢を重ねていくうちに「不登校に限らず、社会に蔑ろにされてきた同じ境遇の人たちを助けられないか？」と考えるようになりました。

運良く身体的障害が無く生まれ、経済的不利もなく、コードが書ける私が、障害を理由にWebから虐げられてきた人々にできる手助けの一つは、Visibleの開発だと思います。私は、私がこの問題を解決するふさわしい人だという自信を持っています。

5. このプロジェクトについて現在までに取り組んだこと (任意)
類似品の調査や、実験、プロトタイプの開発など、今までに取り組んだことがあれば書いてください。なにがどこまでできていて、どういったことがこれから難しそうかを詳しく書いてくれたら、面談でのやりとりがスムーズになります

既にコア部分、ルール、CLI、Webクライアントの一部を開発しており、ライブデモをデプロイしているほか、GitHubでソースコードもご覧いただけます。

- Web版ライブデモ
- CLI版ライブデモ
- GitHubリポジトリ
- タスク一覧
- ドキュメント

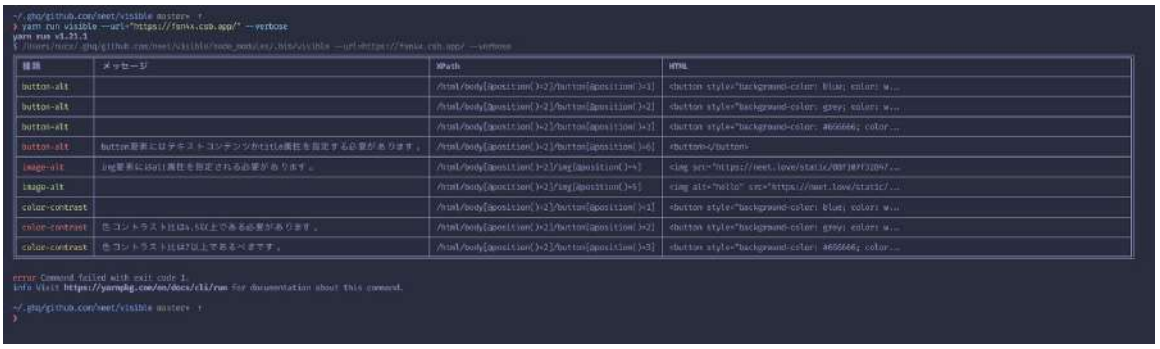
進捗状況はissuesやPRにコメントとともにメモしておりますので、ご覧いただければと思います。

進捗

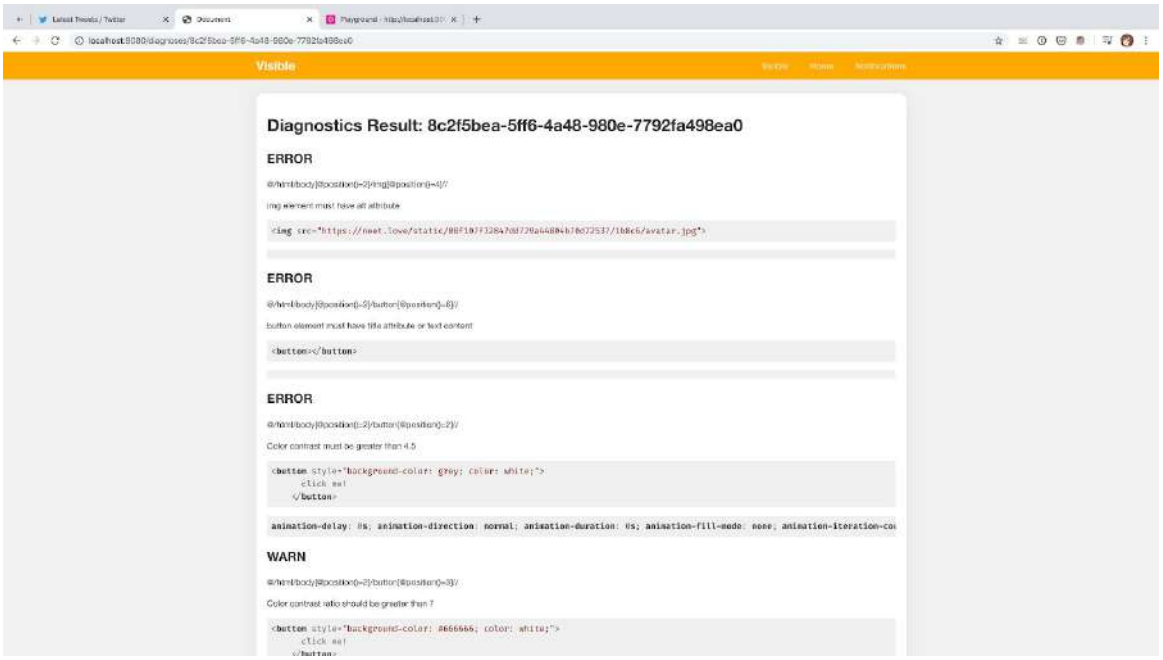
完了

- コアのプラグイン読み込みの仕組み、コンフィグ指定の仕組み
- コアのルール実行、レポート収集の仕組み
- CLI, Webクライアントのプロトタイプ
- 一部ルール実装
    - 画像にキャプションがあるか
    - WCAGに準拠した色コントラスト比か
    - ボタンにテキストがあるか

CLI版



Web版



ロゴデザイン

visible

未完了

- **ルール実装**
  アイデアは無限にあるので未踏ジュニア期間中に作れるだけ作ろうと思います
- **プロバイダー実装**
  GCPを契約してからでないと進められないのでまだ手をつけていません。上のライブデモで出来るのは問題検出のみで、修正の実装は未完了です。
- **React / Vueインタプリタ #23**

axeの節で言及したものです。具体的にはStroybookのようにコンポーネント単位でコンテナ内にマウントしてテストを実行する感じになるかなと思います。

- **テストの進捗状況を表示する仕組み**
  コアでrxを使って、バックエンドにjob queueを実装します
- **アクセシビリティー仕様との対応付け #21**
  例えばWCAG, EPUB, BBC HTML Accessibility Standardsなどの標準に対応するルールをルールセットとして整理します
- **W3C ACT対応 #20**
  W3Cがこの手のツールのフォーマットを標準化しているので、これに準拠します

開発を進めていくにあたって新たなタスクが判明する可能性もあるので目安として参考にしていただければと思います。

6. 提案者がこれまで制作したソフトウェアまたはハードウェア

- 自分が、このプロジェクトを進めるにあたり十分な能力があることを、アピールしてください。（特に、5.でまだプロトタイプなどの開発をやっていないと解答した方は、ご自身の能力を強くアピールしてください。）
- これまでの活動実績が載っているホームページや、GitHubのアカウント、YouTubeチャネル等がある場合も、こちらでアピールしてください。
- フォーマットは自由です。図表や画像も使用できます。ページ数も制限はありません。複数人で開発した場合は、どの部分を担当したのか、明確に記述してください。

Togetter

2019年9月よりアルバイトとして開発に参加しているWebサイトです。Twitterのツイートから記事を作成するというアイデアの基に設計され、今年で10周年を迎えました。

Masto.js

分散型ソーシャルネットワークプラットフォームMastodonのTypeScript版APIクライアントです。

2018年から2年間継続してメンテナンスを行っているほか、開発中に発見した問題を基にMastodonにコントリビュートしたり、Mastodon本体のアクセシビリティーを修正したり、日本語翻訳に参加するなどして分散SNSコミュニティーに貢献しています。

Refined Itsukara.link

バーチャルユーチューバーの配信時間がチェックできるWebサイト「いつから.link」をスタイリッシュなデザインと柔軟な機能で非公式に再設計したWebサイトです。GraphQLやServer-side Renderingなどの先進的な技術を多く導入しました。*リリース時のツイート*

VSCode Qiita

プログラマー向けブログサイトQiitaをテキストエディターVisual Studio Codeから編集できるようにする拡張機能です。*リリース時のツイート*

その他、GitHubアカウントやWebサイトもご覧いただければと思います。

7. 週あたりの作業時間の目安

作業時間:
学期中 ×時間
夏休み中 ◇時間

一日3時間*週3日で9時間ほどかなと考えています。開発の進捗に依っては柔軟に変えることも可能です。

普段のプログラミングにおける作業時間の統計をWakaTimeで公開しておりますので参考にしていただければと思います。

- https://wakatime.com/@Ryo

8. 自己アピール

その他、まだアピールしきれていない、得意なことや、ほかの人にはないような経験があればアピールしてください。必ずしも提案内容と関連している必要はありません。

（註: 諸事情で削りましたが、普段しているプログラミングのことやバックグラウンド、持っている資格について箇条書きしましたものを提出しました。）