

2020年9月9日

株式会社インプレスR&D

<https://nextpublishing.jp/>

Web ブラウザで作る本格インタラクティブ！

『入門 GUI』発行

技術の泉シリーズ、9月の新刊

インプレスグループで電子出版事業を手がける株式会社インプレス R&D は、『入門 GUI』(著者:hashrock、daiiz、miyaoka、f_subal)を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」や「技術書同人誌博覧会」をはじめとした各種即売会や、勉強会・LT 会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『入門GUI』

<https://nextpublishing.jp/isbn/9784844379034>



著者:hashrock、daiiz、miyaoka、f_subal

小売希望価格:電子書籍版 1800円(税別)／印刷書籍版 2000円(税別)

電子書籍版フォーマット:EPUB3／Kindle Format8

印刷書籍版仕様:B5判／カラー／本文92ページ

ISBN:978-4-8443-7903-4

発行:インプレス R&D

<<発行主旨・内容紹介>>

本書は、Web ブラウザにおける本格的な GUI の入門書です。

Web ブラウザに期待される役割は年々増えています。ネイティブアプリ顔負けのデザインツールや、ドラッグアンドドロップでの操作が必要なフォームを作りたくなったとして、我々は何に気をつけるべきなのでしょう？

これまで HTML / CSS / JavaScript で実装するのが大変だった UI をどう実装するべきなのでしょう？

この点を本書は重点的に扱います。本書では定番のセレクトボックスやヘッダーレイアウトの他、バウンディングボックスやツリーといったドラッグを伴う UI なども実装していきます。苦しくも楽しい、Web ブラウザの複雑な GUI の世

界に、エディタとブラウザを開きながら足を踏み入れてみましょう。
 (本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。)

イラストを交えてフルスクラッチ GUI を解説

図1.2: 結局作らないのが一番という結論になる

1.2 select要素を自作してみよう

ここから実装を試みていきますが、実力に自信のある方は、すぐここから先を読まないことをおすすめします。

あくまでここからは、筆者 hashrock の実装の一案にすぎません。自分のやり方、アイデアで実装してみたくはないですか？

React でも Vue でも Angular でも構いません。まずは自分で実装を試みて、それからここから先を読んだほうが、10倍楽しめることでしょう。自分のアイデアをブラウザというキャンバスで試してみませんか？

1.3 トレースもとを観察しよう

まず、トレースもとのUIを配置します。

```
<select>
  <option>Option 01</option>
  <option>Option 02</option>
  <option>Option 03</option>
  <option>Option 04</option>
  <option>Option 05</option>
</select>
```

これをレンダリングすると、下記の図のようになります。

図1.3: 普通のSelect要素

ひととおり操作を行い、どんな遷移があるか洗い出しましょう。遷移を表示すると、下記ようになります。

図1.4: マウス操作による遷移

1.4 モックしよう

開発順序は人によるかと思いますが、私は書きながら設計を考えたいので、まずモックを選択します。一番状況が整いそうなViewを先に作ってしまいます。

select要素は、ボタンとリストで構成されたコンポーネントとみなすことができます。

図1.5: そんなに難しくなさそう (油断)

select + 要素は

て、できています

まずは細かい挙動はおいといて、開いた状態のコンポーネントをモックします。v-ifなどで非表示にすることで、これらの挙動を再現できます。

今回の野はフロートするoptionsで、position: absoluteが必要になりそうです。

8 | 第1章 フルスクラッチGUI入門
第1章 フルスクラッチGUI入門 | 9

バウンディングボックスの移動、拡張、回転を解説

図2.1: いろいろ変形

2.2 レイヤーについて

バウンディングボックスを持つツールには、おそらく「レイヤー」という概念があるでしょう。バウンディングボックスは、本質的にはレイヤーというモデル (あるいはそれに相当する何か) のビューであり、レイヤーの設計を考えることは、バウンディングボックスの設計に影響します。レイヤーの対象には画像、図形、文字などがありますが、ここでは単純な矩形で考えます。

図2.2: いろいろなレイヤー

大きさ200x100の長方形 (rects) があります。500x500のキャンバス (svg id="canvas") の左上、クリックも動きません。これを移動・拡大・縮小・回転できるようにするのが目標です。

```
<svg id="canvas" viewBox="0 0 500 500" width="500" height="500">
  <rect fill="orange" width="200" height="100" x="0" y="0" />
</svg>
```

図2.3: Hello world! SVG

今この長方形は、なんのレイヤーも表現していません。モデルとなるレイヤーをどこかに持つ必要があります。すべてのレイヤーは、少なくとも「大きさ」「位置」「回転角」を持つでしょうから、次のような形で定義します。

```
type Pixel = number
type Radian = number

interface Layer {
  id: number
  width: Pixel
  height: Pixel
  positionX: Pixel
  positionY: Pixel
  rotate: Radian
}
```

ちゃんとしたレイヤーは「重なり順」なども持つでしょうが、今回は複数レイヤーの並び替えは

3. 「名前」「位置」「回転」の3つをそれぞれ「Transform」で呼ぶことがあります。実際によっては、バウンディングボックスのコンポーネントはLayer型のpropsで返さず、Transform型のpropsを受け取るほうが、より汎用的な実装になるかもしれません。

26 | 第2章 バウンディングボックスことはじめ——移動・拡張・回転
第2章 バウンディングボックスことはじめ——移動・拡張・回転 | 27

エンジニアによる座談会も収録

miyaoka他にも、「メルカリのようなサービスをNoCodeで作る」みたいな記事もありましたね。実際に見ると、けっこう難しそう感じましたけど。

keima今までだと、GUIでアプリ作れる系のサービスはけっこう難しかったですね。

miyaoka最近だと、開発よりもOpsにコストがかかるので、NoCodeで作ったものを結局メンテナンスには……みたいな懸念はあるんですね。そうすると、そこはコーディングが必要になる。

f_subatまあそこはさっきの階層の話と一緒に、難しい問題が起きたら一段階低レイヤーなことをするって話じゃないかなと。サーバーが話まったら、アプリケーションレイヤーじゃないところで頑張りたいな。

miyaoka長い歴史では、そこを克服して高機能化していくはずなんですけどね。車だって100年後には自動運転かもしれないし……。

f_subatでも車の場合も、自動運転で通りにくいところは、そこだけ手で運転する人が出たりして……。

5.3 「モーターで走る馬」を作らない

keimaSTUDIOだとよく馬が車になって……みたいな話をしますよね。

miyaokaさっきの話も見方を変えると、馬が車で代替されて、馬の時代にあった痛みと車の時代にあった痛みが別、みたいな言い方もできますよね。

keimaコードの痛みは馬の痛みで、GUIの痛みは車の痛みみたいなね。今の時代は馬が選択されていると。

keimaSTUDIO作ってるのとよく、「このままだと我々は「モーターで走る馬」ができてしまう」と言ったりすることがあるんですけど。

一問 (笑)

f_subat「モーターで走る馬」って、どういう文脈の言葉なんですか？

keimaSTUDIOは、今までコードで書かれていたものを全く別の概念にする、という話で「車を作るんだ」と言うことがあるんです。たとえば、STUDIOではCSSのクラスを入れる機能がないんですよ。クラスってコードでは必須の概念だけど、STUDIOのGUI環境では、必ずしもいる概念じゃないよねって。前の時代と同じものが正しいわけじゃないと。

f_subatああ、じゃあ仮にCSSのクラスみたいな、すごい柔軟なものを感じて入力する何か(?)を作ってしまうと、それは「モーターで走る馬」みたいなものだよ。

keimaそうですね。モーターは車の中にいたほうが良いもので、馬という動力でやることは違う。何かをGUIにするとともに、前やっていた機能をそのまま全部GUIに持ち返らなくていい、うまくいかないんですよ。

一問なるほどー

miyaoka似たようなツールだとクラスが入れられるけど、STUDIOはあえてそうしてないですね。

3https://hashrock.com/_static/_next/00227f96d

4https://twitter.com/miyaoka/status/11882220208827244 参照: STUDIO Code の開発者、@C - JavaScriptのコードを書く機能をフロントエンドに実装している。

5.4 コード vs GUI

f_subatちょっと話がそれるんですけど、CSSのクラスは入れないにしても、今STUDIOの中にコードが書ける機能は入れようとしてるって聞いたんですけど。さっきの話がいろいろ、それでもSTUDIO Codeの方はやろうと思ったのは、どういう理由からなんですか？

keima……良い質問ですね (笑)。STUDIO Codeは元をたどると、「STUDIOではスタティックなページしかできないという問題をどう解決するか」という話から出てきたんですよ。いずれデータにもとづいて、動的なページを作りたいって話があったんですが、何度もボツになって。最初はフィルターみたいなUIを作って、データをGUIで加工するもの考えたんですけど、いうてコードじゃないと追いつかない場面が出てきたんですよ。

keima今STUDIOにはRSSの機能があって、日付のフォーマットが選べるんです。これも表向きにはコードを隠してるんですけど、内部的にはJavaScriptのコードを保存するような箇所もあります。

keima色々悩んでるうちにふと、これって「コンポーネント」を作る機能なんじゃないかと思って。「STUDIO Code」と言いつつも、フロントエンドエンジニアが知っている「コンポーネント」をGUI化するって発想でやることにしたんです。要は、今までただ「ボックス」として作っていたものに、ReactやVueのような、[state] や [method] という概念を持たせようと思った。

keimastateとかdataはまあ、GUIで表現できなくもないんですけど、処理 (getterやmethod) についてはコードが必要になると。最近だと、何でしたっけ、なんかブロックを繋げてコードを書くやつとかありますけど……。

f_subatScratchとか。

keimaそう。でもあれが作りたいかというところ、違うと思うちゃって。将来複雑なものを作るのに、みんながブロックをビコビコ組み立ててるのが想像できなかった。でもだからといって、今のようにファイルをつけて、それをimportするという世界観のままが理想ではないとも思っています。Wixにもコードっぽい機能があるんですけど、ちょっとjQuery世代って感じがあって、今のコンポーネント志向の時代にあった形でそれができたら良いな、という感じですね。

5.5 処理そのものはGUIにならない

miyaokaScratchのアプローチが何故合わないかは、もうちょっと言語化したいですね。ジャンル的には「ビジュアルプログラミング」っていうし、その点ではScratchもSTUDIOも近そうに見えるけど、いかに違う。

f_subatやっぱり、「処理」はGUI表現と相性が悪いんじゃないですかね。ボタンとかは「モノ」なので絵にかけるし、GUIにも落とし込めそうな気がするんですけど。

keima処理ってそもそも命令だし、命令はそもそも言語に近い概念なので。たとえば「走れ」というコマンドをGUIにするのは違う気がする。

f_subat名詞で表現するやつと、動詞で表現するやつ？みたいな……。

84 | 第5章 座談会——STUDIO CPO とする GUI の話

第5章 座談会——STUDIO CPO とする GUI の話 | 85

<<目次>>

- 第1章 フルスクラッチ GUI 入門
- 第2章 バウンディングボックスことはじめ——移動・拡張・回転
- 第3章 スクロールに連動して高さが変わるヘッダビューを作る
- 第4章 ドラッグ&ドロップ並び替え入門
- 第5章 座談会——STUDIO CPO とする GUI の話

<<著者紹介>>

hashrock (第1章 フルスクラッチ GUI 入門)

普段はCを扱う受託系プログラマー。「anydown」や「mangadown」など、Markdownを図やその他諸々に変換するプロダクトを作成している。

Twitter: @hashedrock / GitHub: @hashrock

daiiz (第3章 スクロールに連動して高さが変わるヘッダビューを作る)

仕事ではScrapboxとHelpfeelに携わる。昨年、『探究 SVG とスクリーンショット』を執筆。今年は時系列データを扱うGUIの研究を行う。

Twitter: @daiizplus / GitHub: @daiiz

miyaoka (第4章 ドラッグ&ドロップ並び替え入門)

Web フロント系エンジニア。STUDIO のデザインツールを開発。「JAMstack 完全入門」を執筆。ポッドキャストsoussune。

Twitter: @miyaoka / GitHub: @miyaoka

f_subal (第2章 バウンディングボックスことはじめ——移動・拡張・回転)

ピクシブ株式会社で、グッズ制作と同人誌入稿のサービス「pixivFACTORY (factory.pixiv.net)」のフロントエンドを担当。創作や編集にまつわる UI、TypeScript の型システムを得意とする。本書では、まえがきと表紙イラストも担当。

Twitter: @f_subal / GitHub: @fsubal

<< 販売ストア >>

電子書籍:

Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER

印刷書籍:

Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※ 各ストアでの販売は準備が整いしだい開始されます。

※ 全国の一般書店からもご注文いただけます。

【インプレス R&D】 <https://nextpublishing.jp/>

株式会社インプレスR&D(本社:東京都千代田区、代表取締役社長:井芹昌信)は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。

※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム(またはメソッド)の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド(POD)による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 <https://www.impressholdings.com/>

株式会社インプレスホールディングス(本社:東京都千代田区、代表取締役:松本大輔、証券コード:東証1部9479)を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「モバイルサービス」「学術・理工学」「旅・鉄道」を主要テーマに専門性の高いメディア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター

TEL 03-6837-4820

電子メール: np-info@impress.co.jp