Svelte5でのevent受け渡し

Svelte Japan Offline Meetup #2

2024/02/21 (水)

株式会社 Nextbeat

富永孝彦

自己紹介

- 名前: 富永 孝彦 (とみなが たかひこ)
  - X: @takapi327
  - Github: https://github.com/takapi327
- 所属: 株式会社 Nextbeat
- 趣味
  - ScalaでOSS開発
  - プログラミングに関する情報収集
  - ゲーム

今日話すこと

1. Svelte3,4でのevent受け渡し
2. Svelte5ではどう変わるのか

なぜこの議題か？

Svelte3,4を使用したUIライブラリを開発しており、それが割としんどかった

何がしんどい？

- UIライブラリは汎用的に作る必要がある
- 汎用的に作るにはイベント受け渡しをライブラリ側で頑張る必要がある
- 頑張ると実装コスト、レビューコストが上がってしまう
- 使ってた機能がSvelte4で非推奨になってしまった

Svelte5

Svelte4で非推奨になってた機能とかがなくなってしまった。

ただ更新しないわけにはいかないので、Svete5の機能で作り直してみる。

あれ、めっちゃ楽じゃね？

まずは、Svelte3,4のイベント受け渡しはどのようなものだったか

Svelte3,4でのevent受け渡し

Svelte3,4では`on:`ディレクティブを使用してイベントリスナーを要素にアタッチしていました

```
<button on:click={...}>
  クリック
</button>
```

共通で使用するコンポーネントを作成する場合、
子から親にイベントを伝搬する必要があります。
Svelte3,4では `createEventDispatcher` などを使用してイベントを伝搬していました。

1. `createEventDispatcher` で `dispatch` を作成
2. イベントハンドラに渡す関数を作成
3. 関数内で `dispatch` に伝搬したいイベント名と受け渡したい値を渡す
4. 関数を使用したいイベントハンドラに渡す

```
// Button.svelte
<script lang="ts">
  import { createEventDispatcher } from 'svelte'

  const dispatch = createEventDispatcher()

  function sayHello() {
    dispatch('click', {
      text: 'こんにちは!'
    })
  }
</script>

<button on:click={sayHello}>
  クリック
</button>
```

親コンポーネントでは、子コンポーネントの `dispatch` に渡されたイベント名を使用できるので、そのイベントハンドラに関数を渡して `dispatch` 経由で渡された値を使用するという使い方になるかと思います。

```
// App.svelte
<script lang="ts">
  import Button from '../lib/component/Button.svelte'

  function handleMessage(event) {
    alert(event.detail.text)
  }
</script>

<Button on:click={handleMessage}/>
```

Playground

もしくは、関数を渡すという方法を使用することもできます。

```
// Button.svelte
<script lang="ts">
  export let sayHello: (text: string) => void
</script>

<button on:click={() => sayHello('こんにちは!')}>
  クリック
</button>

// App.svelte
<script lang="ts">
  import Button from '../lib/component/Button.svelte'

  function sayHello(text: string) {
    alert(text)
  }
</script>

<Button {sayHello} />
```

Playground

もしくは`on:`ディレクティブに値を渡さず、イベントを転送してもらって親コンポーネントでイベントをリッスンするという方法もあります。

```
// Button.svelte
<button on:click>
  クリック
</button>

// App.svelte
<script lang="ts">
  import Button from './Button.svelte'

  function handleMessage(event) {
    alert('こんにちは！')
  }
</script>

<Button on:click={handleMessage}/>
```

Playground

運用が大変？

- 使用したいイベントは全て記載しなければいけない
    - だたユーザーがどのように使うかはわからないから制限するか全部を有効にする必要もあったり
- コンポーネントが大きくなってくるとイベントのforwardやdispatchなどの処理が複雑になってしまう
- プロダクトで共通コンポーネントを作成する際などで、デザインなどは同じなのにフォーム送信用のボタン、画面遷移用のボタン、モーダル表示用のボタンという風に用途ごとにコンポーネントが生まれてしまう可能性がある
- 条件判定用のフラグを用意して、スーパーコンポーネントを作成すると無闇に改修できなくなってしまうなんてことも...

汎用的に作るために

UIライブラリは汎用的に作る必要がある
そのため`dispatch`や特定の関数で作成したイベントしか使用できないものになってしまうような実装はなるべく避けたい

ライブラリが頑張って柔軟に対応できるように作らなければならない

svelte-material-uiなどのライブラリでは、これを解消するために`SvelteComponent`の`$on`をラップして、イベントをいい感じに受け渡せるようにしている

- forwardEventsBuilder
  https://github.com/hperrin/svelte-material-ui/blob/master/packages/common/src/internal/forwardEventsBuilder.ts
- useActions
  https://github.com/hperrin/svelte-material-ui/blob/master/packages/common/src/internal/useActions.ts
- etc...

ただし、Svelteの内部構造を理解しておかないといけないので大変...

Svelte5では？

まず、Svelte5ではイベントハンドラを渡すときは`on:`ディレクティブを使用するのではなく、`onxxx`を使用するように変更になりました。

なぜ変更？

「イベントハンドラとpropsを区別するアプローチを取っていたが、それが原因でコンポーネントをまたぐイベントのforwardやpropagate、dispatchやdelegationが複雑になってしまった。」[3]
というのが変更した理由のようです。

以下詳細についてはSvelte Summit Fall 2023で解説がされていますので、こちらをご覧いただければと思います。
https://www.youtube.com/live/pTglx-ucMsY?si=PZrN6ZKczM-_j8bj&t=14096

注意

Svelte5は本日(2024/02/21)時点ではまだ正規リリースされていません。
今回ご紹介する機能や実装方法は今後変更される可能性があります。

Svelte5では$props機能が追加

Svelte5から追加された`$props`とは、`export let`の代わりとして使用することができる機能です。

今までSvelte3,4で使用されていた、`$$props`や`$$restProps`といった機能の代わりとなるものです。

Svelte3,4

Playground

```
// Button.svelte
<script lang="ts">
  export let label: string
  export let type: string
</script>

<button type={type}>
  {label}
</button>
```

```
// App.svelte
<script lang="ts">
  import Button from './Button.svelte'
</script>

<Button label="クリック" type="submit" />
```

Svelte5

Playground

```
<script lang="ts">
  let { label, type } = $props()
</script>

<button type={type}>
  {label}
</button>
```

```
// App.svelte
<script lang="ts">
  import Button from './Button.svelte'
</script>

<Button label="クリック" type="submit" />
```

より簡潔に値を受け渡せるようになりましたね！

$propsを使用すればイベントハンドラも渡せる

今までは単純な値か自身で作成した関数しか受け取ることはできませんでした。
しかし、`$props`を使用すれば先ほど単純な値を受け渡したような感じで、イベントハンドラも渡すことができるようになります。

```
// Button.svelte
<script lang="ts">
  let { onclick } = $props()
</script>

<button onclick={() => onclick({ text: 'こんにちは!' })}>
  クリック
</button>
```

親コンポーネントでは、子コンポーネントに直接イベントハンドラを設定するように使用することができます。

Playground

```
// +page.svelte
<script lang="ts">
  import Button from '../lib/component/Button.svelte'

  function handleMessage(event) {
    alert(event.text)
  }
</script>

<Button onclick={handleMessage}/>
```

何が嬉しい？

- イベントハンドラごとに `dispatch` や関数を用意する必要がなくなった
- 予めイベントを設定しておく必要もない
- 自身でイベントハンドラ使用したい場合も簡単になった

ここが良い

> イベントハンドラごとに`dispatch`や関数を用意する必要がなくなった

Q. 今もイベントハンドラを指定しているから3,4の時とあまり変わらないのでは？

A. イベントハンドラは指定する必要はない。コンポーネント内で使いたいものだけ指定すれば良い

どういうこと？

`$props` は `$$prosp` や `$$restProps` 機能の代わりとなるものと言いました。
つまり、受け取る引数を指定するだけでなく、渡された値を全て受け取るという風に書くことも可能ということです。

以下のようにスプレッド構文(Spread Syntax)を使用すれば `Button` コンポーネントに渡された値を全て `button` に渡すことができます。

```
// Button.svelte
<script lang="ts">
  let { ...restProps } = $props()
</script>

<button {...restProps}>
  クリック
</button>
```

子コンポーネントでは、スプレッド構文によって受け取った値を全て渡すようにしました。

イベントハンドラに対して何も処理を記載していないはずですが、親コンポーネントでは以下のようにイベントを自由に使用可能です。

Playground

```
<script lang="ts">
  import Button from '../lib/component/Button.svelte'

  function onmouseover(event) {
    console.log('マウスが乗った！')
  }

  function onmouseout(event) {
    alert('マウスが離れた')
  }
</script>

<Button {onmouseover} {onmouseout} />
```

Svelte5

用途ごとにコンポーネントを用意する必要も条件分岐を使ったスーパーコンポーネントを作成する必要もない！

イベントをそのまま渡すことが可能かつ、コンポーネント内で使いたいイベントがあれば自由に使える

Selte3,4の時みたいにSvelteComponentの `$on` をゴニョゴニョする必要がなくなった
※SvelteComponentの `$on` はSvelte5でなくなった

つまり

実装コストは下がったのに自由度は上がった！

より柔軟なコンポーネントを作成可能に！

しんどかったものが

- UIライブラリは汎用的に作る必要がある
  => 標準機能だけで実現可能に！
- 汎用的に作るにはイベント受け渡しをライブラリ側で頑張る必要がある
  => 何も頑張ってない！
- 頑張ると実装コスト、レビューコストが上がってしまう
  => 標準機能だから敷居はほとんどなくなった！
- 使ってた機能がSvelte4で非推奨になってしまった
  => むしろいらない！

うん、めっちゃ楽！

触ってみて

最初はSvelte5で内部のAPIを使用できなくなったり(Svelte4で非推奨)、いろいろな機能が変わるので大変そうとか複雑なりそうと思っていました。

しかし、実際に触ってみるとむしろ開発は楽になった印象です。

学習コストも少なくなって、今までの処理がより簡単に書けるようになったと思います。

Svelte3,4で書かれたUIライブラリも改修するのは大変だと思いますが、Svelte5の新機能を使えば標準の機能だけでほとんどのことが解決できると思います！

つまり、Svelteが書ければ誰でも保守できるライブラリになることも可能！？
(むしろライブラリなどいらないのでは？)

他にもいろんな新機能が！

- $state
- $derived
- $effect
  - $effect.pre
  - $effect.active
  - $effect.root
- $inspect
- Snippets

早く正式版を使いたいので、リリースが楽しみですね！

Svelte5 UI ライブラリ

以下でSvelte5の新機能を使ってHeadless UIライブラリを趣味で開発しています。
(まだ1個しかないけどw)

新機能の使い方の参考にしていただければ！
また、もっとこうした使い方の方がいいよなどあれば教えていただけると嬉しいです！

https://github.com/takapi327/svelte-headlessui

Nextbeatでは

Svelte, SvelteKitを採用しており、現在3つ目の技術移行中です！

こんな感じでSvelteの採用や新バージョンへの追従を積極的に行なっています。

興味がありましたら会社資料説明を見ていただけると嬉しいです。

参考文献

1. https://qiita.com/sho_U/items/2d52590bd7973fbec671
2. https://qiita.com/oekazuma/items/ab617096af10ad94356e
3. https://zenn.dev/tomoam/scraps/375fb71c09fe0f
4. https://www.youtube.com/watch?v=pTgIx-ucMsY
5. https://svelte.jp/blog/runes