

<TheTooltip />

使いやすいツールチップを実装する方法

ken7253

Frontend developer

技術記事を書いたりするのが趣味。

最近はReactを使ったアプリケーションを書いています。

ユーザーインターフェイスやブラウザが好き。

 <https://github.com/ken7253>

 <https://zenn.dev/ken7253>

 <https://dairoku-studio.com>



TooltipとはどういうUIなのか

TooltipとはどういうUIなのか

ARIA Authoring Practices Guide の説明が分かりやすい。

“

A tooltip is a popup that displays information related to an element when the element receives keyboard focus or the mouse hovers over it.

”

「ツールチップは、要素がキーボードフォーカスを受けたり、マウスカーソルをその上に置いたりしたときに、その要素に関連する情報を表示するpopupアップです。」

ツールチップの実装時に考慮すること

- Machine readability
- Cancelability
- Selectivity

基本的には ARIA Authoring Practices Guide や WCAGを参考にする。

その中で関連のありそうな項目を参考にUIを組み立てていく。

- Tooltip Pattern | APG | WAI | W3C
- WCAG 2.2 - Success Criterion 1.4.13 Content on Hover or Focus

今回の対象としては上記を参考にしています。

実装時の注意点

Machine readability

- `role` と `aria-describedby` を設定する。
- フォーカス可能な要素を含む場合はモードレスダイアログとして実装する。

汎用UIパーツは適切なマークアップをしないとテストコードが汚くなる。

見た目だけがUIではない のでブラウザからもUIが認識できるようにしておく。

Cancelability

ESC キーで一時的に非表示にできるようにする。

ARIA APGを見るとインタラクションとして **ESC** キーでのキャンセルが記載されている。

Tooltip Pattern

About This Pattern

NOTE: This design pattern is work in progress; it does not yet have task force consensus. Progress and discussions are captured in [issue 128](#).

A tooltip is a popup that displays information related to an element when the element receives keyboard focus or the mouse hovers over it. It typically appears after a small delay and disappears when **Escape** is pressed or on mouse out.

Tooltip widgets do not receive focus. A hover that contains focusable elements can be made using a non-modal dialog.

Selectivity

大抵のツールチップはツールチップ内部のテキストをコピペできない
イベントハンドラーを張る要素に気をつける。

Selectivity

Bad

```
<div>
  <div role="tooltip" hidden={!open} id={id}>
    {content}
  </div>
  <button aria-describedby={id} onPointerEnter={} onPointerLeave={}
    {children}
  </button>
</div>
```

Good

```
<div onPointerEnter={} onPointerLeave={}
  <div role="tooltip" hidden={!open} id={id}>
    {content}
  </div>
  <button aria-describedby={id}>
    {children}
  </button>
</div>
```

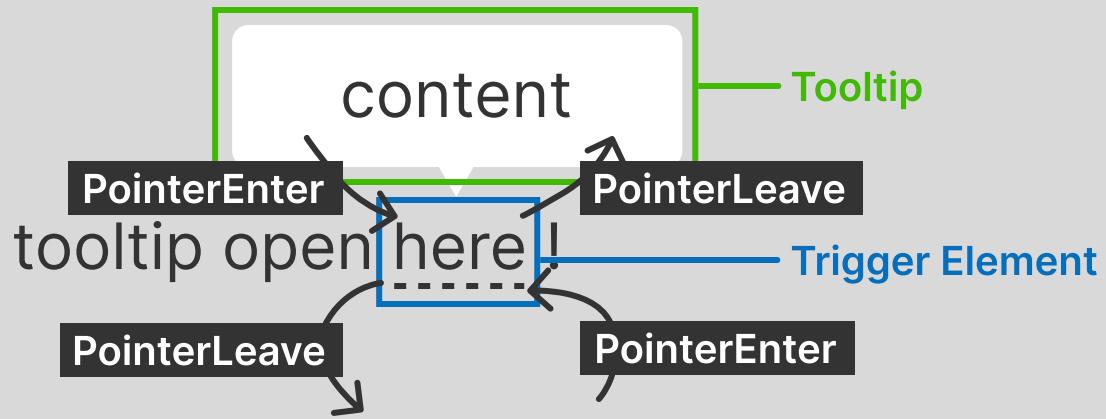
Selectivity

なぜコンポーネント全体に `onPointerEnter` と `onPointerLeave` を設定するのか。

下記は素直にボタンにだけにイベントハンドラーを貼った場合の例

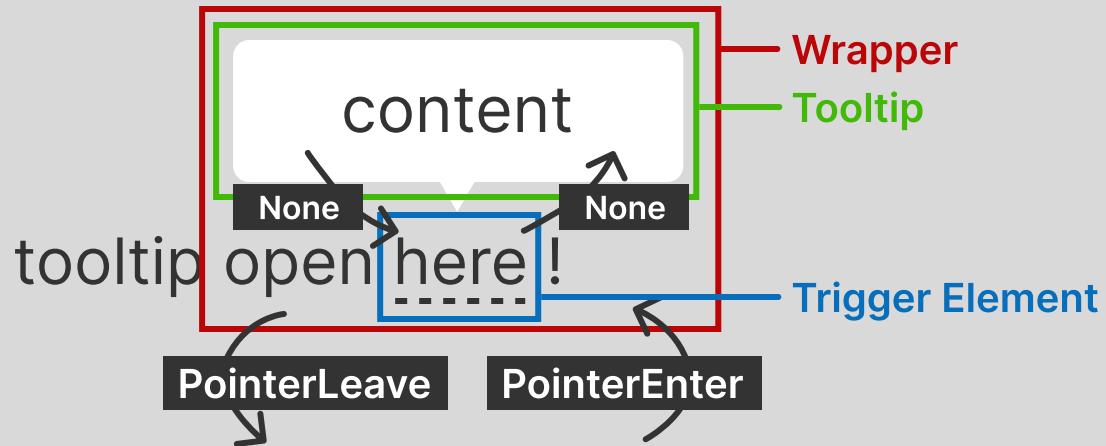
```
<div>
  <div role="tooltip" hidden={!open} id={id}>
    {content}
  </div>
  <button aria-describedby={id} onPointerEnter={()} onPointerLeave={()}>
    {children}
  </button>
</div>
```

Selectivity



トリガーとなる要素だけにイベントハンドラーを貼ると
ツールチップにポインタが移動した場合 `PointerLeave` が発火して消えてしまう。

Selectivity



ツールチップとして認識する範囲を広げてWrapperを用意すると

ツールチップにポインタが移動しても **PointerLeave** が発火せず選択できる。

解説されると分かるけど、普段はあまり気付けない

どうしてこうなった

どうしてこうなった

- ググって最初に出てきたコードをコピペ
- 既存の実装を参考にする
- UIライブラリの実装を参考にする

これらが本当にユーザビリティの高い実装をしているとは限らない。