DOMクイズ！

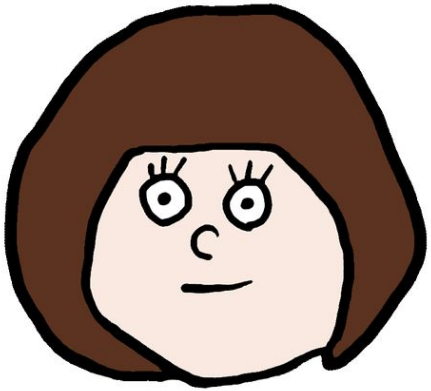
2024/03/28 Encraft #12

@yoshiko

_KNOWLEDGE WORK

自己紹介

よしこ @yoshiko_pg



株式会社ナレッジワークのフロントエンドエンジニア
GUIをSPAとして作るのが好き。

自社での技術スタックや設計を Zennで公開しています。
https://zenn.dev/yoshiko

DOMクイズというか、
DOM Eventクイズになりました！

第1問！

「最初の1回だけ呼び出したいイベントリスナー」を設定したいとき、こういうコードを書いたことありませんか？

```
window.addEventListener('click', handleClick);

function handleClick() {
  // 呼び出されたときの処理
  // ...

  // 以降は呼び出されたくないので、最後に登録を解除する
  window.removeEventListener('click', handleClick);
}
```

実はこれ、もっと簡単に書けます！

DOMクイズ！第1問 - 問題

「最初の1回だけ呼び出したいイベントリスナー」を設定できるコードは以下のうちどれでしょう？

① `addEventListenerOnce('click', fn);`

② `addEventListener('clickOnce', fn);`

③ `addEventListener('click', fn, { once: true });`

④ `addEventListener('click', fn, true);`

DOMクイズ！第1問 - 正解

「最初の1回だけ呼び出したいイベントリスナー」を設定できるコードは以下のうちどれでしょう？

① `addEventListenerOnce('click', fn);`

② `addEventListener('clickOnce', fn);`

⭕ ③ `addEventListener('click', fn, { once: true });`

④ `addEventListener('click', fn, true);`

DOMクイズ！第1問 - 解説

addEventListenerの第三引数はoptionで、

`{ once: boolean, capture: boolean, passive: boolean, signal: AbortSignal }` になっています。

- `once` : trueだと発火したあとに自動でイベントリスナーが削除される
- `capture` : trueだと「子から親へ伝搬するバブリングフェーズ」ではなく、
  その前に起きる「親から子へ伝搬するキャプチャリングフェーズ」でイベントが呼び出される
- `passive` : trueだとそのイベントリスナー関数がpreventDefaultを実行しない宣言。
  キャンセル待ちによるブロッキングが起きないのでスクロールのパフォーマンスが向上する
- `signal` : `new AbortController()` で作ったcontrollerの `controller.signal` をセットすると、
  `controller.abort()` でイベントリスナーを削除できる

DOMクイズ！第1問 - 解説

第三引数は、元々は選択肢 ④ の `addEventListener('click', fn, true)` というbooleanの形でした。

このときの第三引数booleanは `useCapture` で、今のcatpureオプションの役割でした。

once, passive, signalは後から拡張されて追加されたことがわかりますね。

ちなみに選択肢 ① ② の真似元は、JSXでのキャプチャ指定の書き方でした！

通常: `<button onClick={handler}>`

キャプチャ: `<button onClickCapture={handler}>`

第2問！

マウスのクリックでも、スクリーンへの指でのタッチでも、ペンでのタッチでも発火するイベントはどれ？

① mousedown

② touchstart

③ cursordown

④ pointerdown

マウスのクリックでも、スクリーンへの指でのタッチでも、ペンでのタッチでも発火するイベントはどれ？

① mousedown

② touchstart

③ cursordown

⭕ ④ pointerdown

mouse + touch = pointer という感じで、統一して扱えるようになりました。suffixはMouseEventに揃えられています。

MouseEvent	TouchEvent	PointerEvent
mousedown	touchstart	pointerdown
mouseup	touchend	pointerup
mousemove	touchmove	pointermove
なし	touchcancel	pointercancel
mouseover	なし	pointerover
mouseout	なし	pointerout
mouseenter	なし	pointerenter
mouseleave	なし	pointerleave

event.pointerType で
"mouse" | "touch" | "pen" を
取得できるので、
ペンなら鉛筆で
タッチなら消しゴム、
のような処理の
場合分けもできる

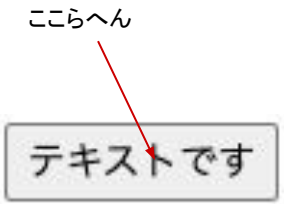
第3問！

「テキストです」の文字部分をクリックしたら、event.targetには何が入る？

```
<button id="button">
  <span>テキストです</span>
</button>

<script>
  const button = document.getElementById('button')

  button.addEventListener('click', (event) => {
    console.log(event.target)
  })
</script>
```



① button要素

② span要素

③ textContent

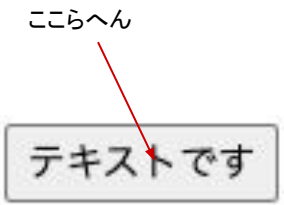
④ 'ト'

「テキストです」の文字部分をクリックしたら、event.targetには何が入る？

```
<button id="button">
  <span>テキストです</span>
</button>

<script>
  const button = document.getElementById('button')

  button.addEventListener('click', (event) => {
    console.log(event.target)
  })
</script>
```



① button要素

⭕ ② span要素

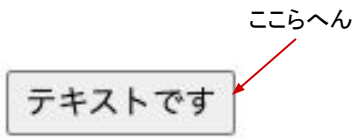
③ textContent

④ 'ト'

DOMクイズ！第3問 - 解説

`event.target` には「イベントが発生した要素」が入ります。今回だとbuttonの中のspanになります。

buttonのキワキワの縁あたりをクリックするとspanのエリアから外れるのでbuttonが入ります。



「今実行されているイベントリスナーが登録されていた要素」は `event.currentTarget` から取得できます。

こちらはどの要素をクリックしたかには関係なく、addEventListenerを設定した要素が入ります。

問題が `event.currentTarget` であれば①のbuttonが正解でした！

普段のReact/TSでのコーディングでも `target` と`currentTarget` の違いを知っていると嬉しい場面があります。

`event: React.MouseEvent<HTMLInputElement>` のとき、`event.target` の型は `EventTarget` 型ですが、`event.currentTarget` の型は `HTMLInputElement` 型になるので `.value` など独自プロパティにアクセスできます。

```
const handler = (event: React.MouseEvent<HTMLInputElement>) => {
    console.log(event.target.value)
    console.log(event.currentTarget.value)
}
```

Property 'value' does not exist on type 'EventTarget'

currentTargetがInputElementと推論されているのでオッケー

ただvalueを使いたい場面は `ChangeEvent` であることが多く、`React.ChangeEvent<HTMLInputElement>` では気を利かせて `event.target` も `HTMLInputElement` 型にしてくれるので、普段気付くことは少ないかもしれません。

`event.target` の活躍どころは

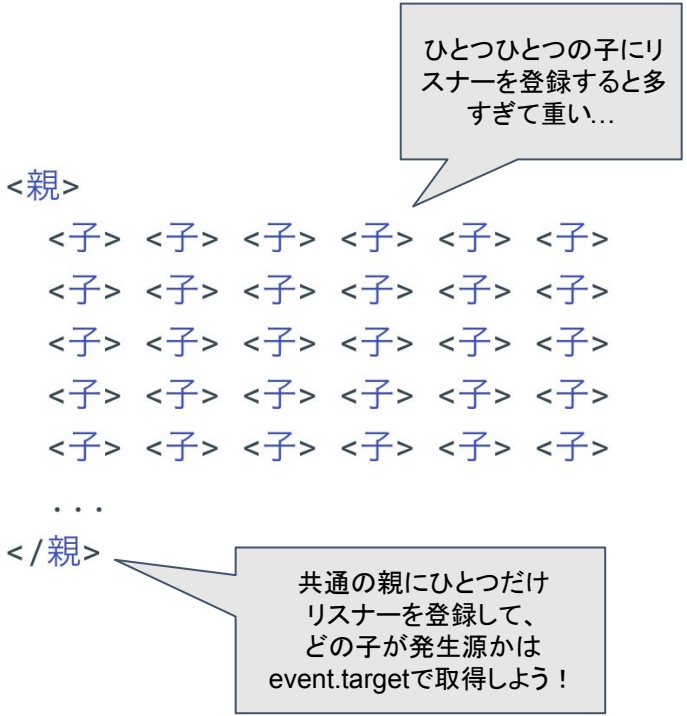
ないんですか！？

DOMクイズ！第3問 - 解説

たくさんの要素にイベントリスナーを登録したいのに、そのまま登録してしまうと数が多すぎてパフォーマンスが悪くなる場合、共通の親にだけイベントリスナーを登録して、どの要素でイベントが発生したかは `event.target` で判別する手法があります。

イベントデリゲーション と呼ばれています。

`event.target` 経由で手に入るのが生DOMなので、仮想DOMの環境下で使うことはあまりなさそうですね。



第4問！

typeがnumberやrangeのinputでもvalueはstringで返ってきてしまうの、変換めんどくさいですよね

```
<input type="number" value="1" id="input" />

<script>
  const input = document.getElementById('input')

  input.addEventListener('change', (event) => {
    console.log({ a: event.target.value }) // '1'
  })
</script>
```

実はこれ、数値で取得する方法あります！

typeがnumberやrangeのinputで、値を数値で取得できる方法はどれ？

① event.target.valueAsNumber

② event.target.value.toNumber()

③ event.target.value.cast()

④ <input type="number" autoCast />

typeがnumberやrangeのinputで、値を数値で取得できる方法はどれ？

⭕ ① event.target.valueAsNumber

② event.target.value.toNumber()

③ event.target.value.cast()

④ <input type="number" autoCast />

`event.target.valueAsNumber` で数値としてvalueを取得できます。

typeがdate系のinputではvalueAsNumberに加えて `event.target.valueAsDate` も取得できます。

詳しい変換アルゴリズムはHTML Standardの仕様に載っています。

ついにDOMの仕様からも飛び出してしまいましたね…！

DOMクイズ 改め DOM Event and HTML クイズ！でした。

できる喜びが巡る日々を届ける

Deliver the joy of enablement