Creating interactive blocks: old, new and good ways

JuanMa Garrido



JuanMa Garrido

Javascript developer and trainer
Developer Relations

Full time contributor in WordPress

in juanmagarrido

y juanmaguitar





Interactive Blocks

- Behaviour for visitors
- Logic in frontend <u>view.js</u>
- Javascript (library?) in the client





block.json "editorStyle": "file:./index.css", "editorScript": "file:./index.js", "style": "file:./style-index.css", "viewScript": "file:./view.js"

Block folder structure

block.json
edit.js
index.css
index.js
save.js
style.css

view.js

Client JS Code!





Register Block!

gutenberg-interactive.php

```
function interactive_block_demos_register() {
    register_block_type( __DIR__ . '/build/counter-jquery/block.json' );
}

add_action( 'init', 'interactive_block_demos_register');

add_filter( 'render_block_gutenberg-interactive/counter-jquery', function( $content ) {
    wp_enqueue_script( 'jquery' );
    return $content;
});
```







Some ways to create Interactive Blocks

- Javascript
- jQuery
- Web Components
- Alpine
- React





Javascript

Pros:

- No library
- Basic knowledge

Cons:

- Imperative code





```
save.js
```

Selectors Needed

```
const blockClassPrefix = 'wp-block-gutenberg-interactive-counter-js'
const Save = ( { attributes } ) ⇒ {
        return (
                <div { ... useBlockProps.save() }>
                        <button className={blockClassPrefix + "_decrement"}>
                        </button>
                        { ... }
                        <button className={blockClassPrefix + "__increment"}>
                        </button>
                </div>
        );
```





Native

view.js

```
const blockClassPrefix = '.wp-block-gutenberg-interactive-counter-js'
const incrementButton = document.querySelector(blockClassPrefix + '__increment');
const valueBox = document.querySelector(blockClassPrefix + '__value');
incrementButton.addEventListener( 'click', () ⇒ {
         valueBox.value = parseInt(valueBox.value) + parseInt(valueBox.step)
})
```

___ Imperative Code





jQuery

Pros:

- Friendly code

Cons:

- Imperative code
- Additional knowledge
- External library





Selectors Needed

```
const blockClassPrefix = 'wp-block-gutenberg-interactive-counter-jquery'
const Save = ( { attributes } ) ⇒ {
        return (
                <div { ... useBlockProps.save() }>
                        <button className={blockClassPrefix + "_decrement"}>
                        </button>
                        { ... }
                        <button className={blockClassPrefix + "__increment"}>
                        </button>
                </div>
        );
```





jQuery.js ← Library needed

view.js

```
const blockClassPrefix = '.wp-block-gutenberg-interactive-counter-jquery'
const $input = $(blockClassPrefix + '_value');
const step = parseInt($input.attr("step"))

$(blockClassPrefix + '_increment').click(() \Rightarrow {
        const currentValue = parseInt($input.val())
        $input.val(currentValue + step)
})
```

Friendly Code





Web Components

Pros:

- Code (and styles) Encapsulation
- Native

Cons:

- Additional knowledge
- Imperative code
- NO SSR: SEO & Blocking script







#WCEU

view.js

```
class GutenbergCounter extends HTMLElement {
        constructor() { ... }
        connectedCallback() { ... }
        attributeChangedCallback( name, oldValue, newValue ) { ... }
        replaceValue() { ... }
}
window.customElements.define( 'gutenberg-counter', GutenbergCounter );

Encapsulation
```



```
Encapsulation
save.js
 const Save = ( { attributes } ) ⇒ {
         return (
                 <div { ... useBlockProps.save() }>
                         <gutenberg-counter</pre>
                                 value={ attributes.initial }
                                 increment={ attributes.increment }
                         ></gutenberg-counter>
                 </div>
         );
                                           No SSR (Blocking Script)
  No SSR (SEO)
                              view.is
```



Alpine

Pros:

- Declarative code
- HTML enhanced

Cons:

- Library required & Additional knowledge
- No components
- NO SSR: SEO & Blocking script





```
Declarative Code -
save.js
 const Save = ( { attributes } ) ⇒ (
         <div
                 { ... useBlockProps.save() }
                 x-data={ `{ count: ${ attributes.initial } }` }
                 <button x-on:click={ `count += ${ attributes.increment }` }>+</button>
                 <input width="5" type="number" x-bind:value="count" />
                 <button x-on:click={ `count -= ${ attributes.increment }` }></button>
         </div>
  No Components
                                                          No SSR (Blocking Script)
                                     Alpine.js
                                  Library needed
                                                                                    #WCEU
```

React (@wordpress/element)

Pros:

- Declarative code
- Components

Cons:

- Library required
- Additional knowledge





save.js

```
const Save = ( { attributes } ) ⇒ {
        return (
                <div
                    ... useBlockProps.save() }
                  data-gutenberg-attributes={ JSON.stringify( attributes ) }
                >
                        <button>+</button>
                        <input
                                                      No Selectors Needed
                                width="5"
                                type="number"
                                readOnly
                                value={ attributes.initial }
                        <button></button>
                </div>
        );
};
```







Declarative Code

```
view.js
 const Counter = ( { attributes } ) ⇒ {
         const [ counter, setCounter ] = useState( attributes.initial );
         const increment = () ⇒ setCounter( counter + attributes.increment );
         const decrement = () ⇒ setCounter( counter - attributes.increment );
         return (
                         <button onClick={ increment }>+</button> <
                         <input width="5" type="number" readOnly value={ counter } />
                         <button onClick={ decrement } > /button>
                 <∕>
         );
                                        Components
 [...] hydrate( <Counter attributes={ attributes } />, block );
```

Library needed ->

React.js





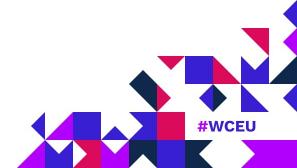
Demo

https://github.com/

wordpress-juanmaguitar/

interactive-blocks-demos





Current explorations

to improve the standard way of creating blocks



https://make.wordpress.org/core/

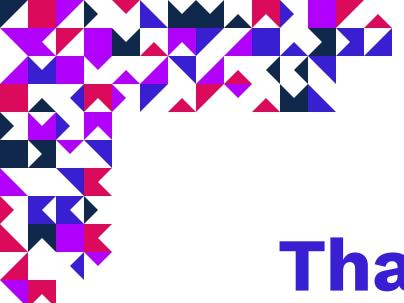
2022/04/27/

<u>exploration-to-enable-better-developer-and-visitor-experiences-with-blocks</u>

Get involved!







Thank you!



