@wordpress/element

- Installation
- Why React?
- API
  - Children
  - cloneElement
  - Component
  - concatChildren
  - createContext
  - createElement
  - createInterpolateElement
  - createPortal
  - createRef
  - createRoot
  - findDOMNode
  - flushSync
  - forwardRef
  - Fragment
  - hydrate
  - hydrateRoot
  - isEmptyElement
  - isValidElement
  - lazy
  - memo
  - Platform
  - PureComponent
  - RawHTML
  - render
  - renderToString

- startTransition
- StrictMode
- Suspense
- switchChildrenNodeName
- unmountComponentAtNode
- useCallback
- useContext
- useDebugValue
- useDeferredValue
- useEffect
- useId
- useImperativeHandle
- useInsertionEffect
- useLayoutEffect
- useMemo
- useReducer
- useRef
- useState
- useSyncExternalStore
- useTransition

- Contributing to this package

Element is a package that builds on top of React and provide a set of utilities to work with React components and React elements.

Installation

Install the module

```
npm install @wordpress/element --save
```

*This package assumes that your code will run in an **ES2015+** environment. If you're using an environment that has limited or no support for such language features and APIs, you should include the polyfill shipped in `@wordpress/babel-preset-default` in your code.*

Why React?

At the risk of igniting debate surrounding any single “best” front-end framework, the choice to use any tool should be motivated specifically to serve the requirements of the system. In modeling the concept of a block, we observe the following technical requirements:

- An understanding of a block in terms of its underlying values (in the random image example, a category)
- A means to describe the UI of a block given these values

At its most basic, React provides a simple input / output mechanism. **Given a set of inputs (“props”), a developer describes the output to be shown on the page.** This is most elegantly observed in its function components. React serves the role of reconciling the desired output with the current state of the page.

The offerings of any framework necessarily become more complex as these requirements increase; many front-end frameworks prescribe ideas around page routing, retrieving and updating data, and managing layout. React is not immune to this, but the introduced complexity is rarely caused by React itself, but instead managing an arrangement of supporting tools. By moving these concerns out of sight to the internals of the system (WordPress core code), we can minimize the responsibilities of plugin authors to a small, clear set of touch points.

API

Children

Object that provides utilities for dealing with React children.

cloneElement

Creates a copy of an element with extended props.

Parameters

- *element* `Element`: Element
- *props* `?Object`: Props to apply to cloned element

Returns

- `Element`: Cloned element.

Component

A base class to create WordPress Components (Refs, state and lifecycle hooks)

concatChildren

Concatenate two or more React children objects.

Parameters

- *childrenArguments* `ReactNode[][]`: – Array of children arguments (array of arrays/strings/objects) to concatenate.

Returns

- `ReactNode[]`: The concatenated value.

createContext

Creates a context object containing two components: a provider and consumer.

Parameters

- *defaultValue* `Object`: A default data stored in the context.

Returns

- `Object`: Context object.

createElement

Returns a new element of given type. Type can be either a string tag name or another function which itself returns an element.

Parameters

- *type* `?(string|Function)`: Tag name or element creator
- *props* `Object`: Element properties, either attribute set to apply to DOM node or values to pass through to element creator
- *children* `...Element`: Descendant elements

Returns

- `Element`: Element.

createInterpolateElement

This function creates an interpolated element from a passed in string with specific tags matching how the string should be converted to an element via the conversion map value.

Usage

For example, for the given string:

“This is a string with a link and a self-closing
tag”

You would have something like this as the conversionMap value:

```
{
    span: <span />,
    a: <a href={ 'https://github.com' } />,
    CustomComponentB: <CustomComponent />,
}
```

Parameters

- *interpolatedString* `Input`: The interpolation string to be parsed.
- *conversionMap* `ConversionMap< InterpolationString< Input > >`: The map used to convert the string to a react element.

Returns

- `ReactElement`: A wp element.

createPortal

Creates a portal into which a component can be rendered.

Related

- https://react.dev/reference/react-dom/createPortal

createRef

Returns an object tracking a reference to a rendered element via its `current` property as either a DOMElement or Element, dependent upon the type of element rendered with the ref attribute.

Returns

- `Object`: Ref object.

createRoot

Creates a new React root for the target DOM node.

Related

- https://react.dev/reference/react-dom/client/createRoot

Changelog

`6.2.0` Introduced in WordPress core.

findDOMNode

Finds the dom node of a React component.

Parameters

- *component* `React.ComponentType`: Component's instance.

flushSync

Forces React to flush any updates inside the provided callback synchronously.

Related

- https://react.dev/reference/react-dom/flushSync

forwardRef

Component enhancer used to enable passing a ref to its wrapped component. Pass a function argument which receives `props` and `ref` as its arguments, returning an element using the forwarded ref. The return value is a new component which forwards its ref.

Parameters

- *forwarder* `Function`: Function passed `props` and `ref`, expected to return an element.

Returns

- `Component`: Enhanced component.

Fragment

A component which renders its children without any wrapping element.

hydrate

> **Deprecated** since WordPress 6.2.0. Use `hydrateRoot` instead.

Hydrates a given element into the target DOM node.

Related

- https://react.dev/reference/react-dom/hydrate

hydrateRoot

Creates a new React root for the target DOM node and hydrates it with a pre-generated markup.

Related

- https://react.dev/reference/react-dom/client/hydrateRoot

Changelog

`6.2.0` Introduced in WordPress core.

isEmptyElement

Checks if the provided WP element is empty.

Parameters

- *element* `unknown`: WP element to check.

Returns

- `boolean`: True when an element is considered empty.

isValidElement

Checks if an object is a valid React Element.

Parameters

- *objectToCheck* `Object`: The object to be checked.

Returns

- `boolean`: true if objectToTest is a valid React Element and false otherwise.

lazy

Related

- https://react.dev/reference/react/lazy

memo

Related

- https://react.dev/reference/react/memo

Platform

Component used to detect the current Platform being used. Use Platform.OS === 'web' to detect if running on web environment.

Here is an example of how to use the select method:

Usage

```
import { Platform } from '@wordpress/element';

const placeholderLabel = Platform.select( {
    web: __(
        'Drag images, upload new ones or select files from your library.'
    ),
    default: __( 'Add media' ),
} );
```

PureComponent

Related

- https://react.dev/reference/react/PureComponent

RawHTML

Component used to render unescaped HTML.

Note: The `renderElement` serializer will remove the `div` wrapper unless non-children props are present; typically when preparing a block for saving.

Usage

```
import { RawHTML } from '@wordpress/element';

const Component = () => (
    <RawHTML>
        <h3>Hello world</h3>
    </RawHTML>
);
// Edit: <div><h3>Hello world</h3></div>
// save: <h3>Hello world</h3>
```

Parameters

- *props* `RawHTMLProps`: Children should be a string of HTML or an array of strings. Other props will be passed through to the div wrapper.

Returns

- Dangerously-rendering component.

render

> **Deprecated** since WordPress 6.2.0. Use `createRoot` instead.

Renders a given element into the target DOM node.

Related

- https://react.dev/reference/react-dom/render

renderToString

Serializes a React element to string.

Parameters

- *element* `React.ReactNode`:
- *context* `any`:
- *legacyContext* `Record< string, any >`:

startTransition

Related

- https://react.dev/reference/react/startTransition

StrictMode

Component that activates additional checks and warnings for its descendants.

Suspense

Related

- https://react.dev/reference/react/Suspense

switchChildrenNodeName

Switches the nodeName of all the elements in the children object.

Parameters

- *children* `ReactNode`: Children object.
- *nodeName* `string`: Node name.

Returns

- `ReactNode`: The updated children object.

unmountComponentAtNode

> **Deprecated** since WordPress 6.2.0. Use `root.unmount()` instead.

Removes any mounted element from the target DOM node.

Related

- https://react.dev/reference/react-dom/unmountComponentAtNode

useCallback

Related

- https://react.dev/reference/react/useCallback

useContext

Related

- https://react.dev/reference/react/useContext

useDebugValue

Related

- https://react.dev/reference/react/useDebugValue

useDeferredValue

Related

- https://react.dev/reference/react/useDeferredValue

useEffect

Related

- https://react.dev/reference/react/useEffect

useId

Related

- https://react.dev/reference/react/useId

useImperativeHandle

Related

- https://react.dev/reference/react/useImperativeHandle

useInsertionEffect

Related

- https://react.dev/reference/react/useInsertionEffect

useLayoutEffect

Related

- https://react.dev/reference/react/useLayoutEffect

useMemo

Related

- https://react.dev/reference/react/useMemo

useReducer

Related

- https://react.dev/reference/react/useReducer

useRef

Related

- https://react.dev/reference/react/useRef

useState

Related

- https://react.dev/reference/react/useState

useSyncExternalStore

Related

- https://react.dev/reference/react/useSyncExternalStore

useTransition

Related

- https://react.dev/reference/react/useTransition

Contributing to this package

This is an individual package that’s part of the Gutenberg project. The project is organized as a monorepo. It’s made up of multiple self-contained software packages, each with a specific purpose.

The packages in this monorepo are published to npm and used by WordPress as well as other software projects.

To find out more about contributing to this package or Gutenberg as a whole, please read the project’s main contributor guide.

□□□ #2
□ □□□ □□□ 2026-10-31 18:18:58 UTC
□ □□□ □□□ 2026-10-31 18:18:59 UTC