# WordPress Gutenberg Block Development with React & PHP

Build a custom Gutenberg block library using React.js and PHP for WordPress 6+. For developers ready to go beyond basic blocks.

Web & Mobile Design Instructor: David Carter • 5 sections • 51 lessons • 18.25h total

View this course online Browse all courses

## About this course

The WordPress block editor has matured significantly, and building custom blocks is now a core skill for WordPress developers. Yet many tutorials stop at simple static blocks and never cover the real-world complexity of dynamic rendering, shared state, or multi-block plugin architecture. This course changes that.

You will build a complete, production-quality block library from scratch using React.js and PHP. We start with Gutenberg fundamentals and progress through dynamic blocks, REST API integration, inner blocks, the @wordpress/data store, and automated testing. Every module includes real code examples, practical exercises, and patterns drawn from actual plugin projects. By the end, you will have a deployable plugin and the knowledge to maintain and extend it as WordPress evolves.

## Course curriculum

### Foundations: WordPress Gutenberg & the Modern Block Editor

- How the Block Editor Works Under the Hood (22m)
  A technical walkthrough of Gutenberg's architecture in WordPress 6+. We trace how blocks load, render, and save data so you understand the full lifecycle before writing any code.
- Blocks, Attributes, and the Block Registry Explained (18m)
  What actually happens when you register a block? We break down block attributes, the registry system, and how WordPress stores block data in post content as serialized HTML with comments.
- Setting Up a Local Dev Environment with Docker and wp-env (25m)
  Step-by-step setup of a modern WordPress development environment using @wordpress/env and Docker. You will have a working local site with hot reloading by the end of this lesson.
- The @wordpress/scripts Build Toolchain (18m)
  Understand what @wordpress/scripts does behind the scenes: webpack configuration, JSX transformation, asset file generation, and how to customize the default setup for your project.
- Navigating the Block Editor UI Like a Power User (12m)
  Before building blocks, you need to deeply understand the editing experience. We tour the toolbar, sidebar inspector, block inserter, and list view, focusing on how your blocks will fit in.
- Exploring Core Block Source Code (25m)
  The best way to learn block development is reading real blocks. We examine the source code of several core blocks, including Paragraph, Columns, and Image, to study patterns you will reuse.
- Understanding block.json and the Modern Block Metadata Format (15m)
  block.json has become the standard way to define blocks. We walk through every field, explain why it matters for both PHP and JavaScript registration, and set up your first metadata file.
- Planning Your Block Library: Architecture Decisions Up Front (20m)
  A short project planning session. We define the block library you will build throughout this course, discuss naming conventions, folder structures, and the decisions that save headaches later.

### Developing Custom Blocks with React and JSX

- How React Powers Gutenberg via @wordpress/element (18m)
  Gutenberg uses React, but not always in the way you expect. This lesson covers the @wordpress/element abstraction, why WordPress maintains its own React wrapper, and what that means for your code.
- Registering Your First Block with registerBlockType() (20m)
  We write your first custom block from scratch. You will define block metadata, create the edit and save components, and see the block appear in the editor inserter.
- Building the Edit Component with JSX and RichText (25m)
  The edit component is what users interact with. We build an editable block using RichText, understand how JSX maps to the rendered DOM, and wire up attribute saving.
- The Save Component: Controlling What Gets Stored (15m)
  Save components define the HTML saved to the database. We cover how to write clean, accessible save markup, when to return null, and how WordPress uses it for front-end rendering.

- Working with Block Attributes, Props, and useBlockProps() (22m)
  A deep look at the attribute system: types, sources, selectors, and default values. We also cover useBlockProps() and why every block component needs it for proper editor behavior.
- Managing Local State with React Hooks (20m)
  Not everything belongs in block attributes. We use useState for UI-only state, useEffect for side effects, and discuss when each approach is the right choice for your block.
- Building a Multi-Attribute Content Block (28m)
  A practical exercise where you build a testimonial block with multiple text fields, an image, and a rating. This reinforces everything from the previous lessons in a single, realistic project.
- Block Validation and Troubleshooting Common Errors (18m)
  Blocks fail validation more often than you think. We cover the validation process, common causes of block invalidation, and a systematic approach to debugging block issues.
- Using External React Components Inside Blocks (15m)
  Sometimes you need third-party React libraries. We cover how to import external packages, manage dependencies with npm, and handle the quirks of loading them inside the editor.
- Building a Second Block: The Feature Card Component (30m)
  You build a feature card block with an icon selector, heading, description text, and a call-to-action button. This lesson cements the block development workflow through repetition.

## Integrating PHP: Server-Side Rendering & Dynamic Blocks

- When and Why You Need Server-Side Rendering (15m)
  Static save markup is not always enough. We cover the scenarios where PHP server-side rendering is necessary and the trade-offs between static and dynamic block approaches.
- Registering Blocks in PHP with block.json (20m)
  The PHP-side registration mirrors the JavaScript side. We set up register_block_type() in PHP, configure render_callback, and load assets properly through block.json metadata.
- Building a Dynamic Block with render_callback (28m)
  We convert one of our earlier blocks to dynamic rendering. The PHP callback receives block attributes and renders server-generated HTML, including data from the WordPress database.
- Enqueuing Front-End Styles and Scripts for Blocks (18m)
  Blocks often need front-end CSS and JavaScript. We cover viewScript, viewStyle in block.json, conditional loading, and how to keep front-end assets minimal and fast.
- Fetching REST API Data Inside Block Components (25m)
  We use @wordpress/api-fetch to pull data from the WordPress REST API into our React block components. Includes loading states, error handling, and caching considerations.
- Creating Custom REST API Endpoints in PHP (25m)
  Sometimes the default REST API endpoints are not enough. We register custom routes in PHP, handle permissions, and connect them to our block's React components.
- PHP Filters, Hooks, and Block Rendering Customization (20m)
  Blocks interact with the broader WordPress hook system. We explore how to use PHP filters to customize block output, add wrapper elements, and modify rendering context.

- Building a Dynamic Posts Query Block (32m)
  A substantial project: build a block that queries posts with custom parameters set in the editor sidebar, renders results server-side, and supports pagination. Combines every PHP concept so far.

## Building a Reusable Block Library: Advanced Patterns

- Structuring a Multi-Block Plugin for Scale (22m)
  A single block in a plugin is simple. Multiple blocks with shared code require structure. We set up a plugin with shared components, utilities, and a clear separation between blocks.
- Creating Shared React Components Across Blocks (20m)
  We extract reusable UI components: color pickers, URL inputs, media uploaders, and icon selectors. These shared pieces reduce duplication and keep your library consistent.
- InnerBlocks: Building Container and Layout Blocks (28m)
  InnerBlocks lets blocks contain other blocks. We build a section container and a column layout block, covering allowedBlocks, templateLock, and useInnerBlocksProps.
- Block Templates and Default Inner Block Configurations (18m)
  Templates define pre-set inner block arrangements. We create block templates for common layouts and discuss when to lock templates versus letting users modify them freely.
- Building Custom Sidebar Controls with InspectorControls (25m)
  The block sidebar is prime real estate for settings. We build custom panel controls, use TextControl, SelectControl, ToggleControl, and organize them into logical panels.
- Adding Toolbar Controls and Block Alignment Options (18m)
  We add toolbar buttons for text alignment, media selection, and custom actions. Covers BlockControls, AlignmentToolbar, and building your own toolbar group.
- Block Styles, Variations, and Transforms (22m)
  Styles change appearance without changing functionality. Variations create preset configurations. Transforms convert between block types. We implement all three in our library.
- Implementing Block Patterns from Your Library (18m)
  Block patterns are pre-designed arrangements of blocks. We register custom patterns that use our library blocks, giving users ready-made layouts they can drop in and edit.
- Introduction to the @wordpress/data Store (25m)
  The data store is Gutenberg's Redux-like state management. We cover stores, selectors, actions, and resolvers, then use them to share state across blocks that need to communicate.
- Using useSelect and useDispatch Across Blocks (22m)
  We wire up interconnected blocks that read from and write to the same data store. Practical examples include a settings block that controls the behavior of a display block.
- Color, Typography, and Spacing Supports in block.json (20m)
  WordPress provides built-in support for color, font size, spacing, and more. We configure these in block.json and customize them with PHP filters for brand-specific palettes.
- Building a Complete Hero Section Block (35m)
  A capstone for this section: build a hero block with InnerBlocks, background image/color controls, text alignment toolbar, responsive spacing, and custom block styles.

## Production-Ready Block Development & Deployment

- Testing Strategy for Gutenberg Blocks (18m)
  An overview of what to test and why. We cover the testing pyramid for block development: unit tests for React logic, integration tests for block registration, and PHPUnit for PHP rendering.
- Writing Jest Unit Tests for Block Components (28m)
  We set up Jest with @wordpress/scripts, write tests for edit and save components, mock WordPress dependencies, and test attribute handling and user interactions.
- PHPUnit Tests for Server-Side Block Rendering (25m)
  Test your PHP render_callback functions in isolation. We set up PHPUnit for WordPress, write tests that verify rendered HTML, and test custom REST API endpoints.
- End-to-End Testing with Playwright and wp-scripts (22m)
  We write end-to-end tests that open the editor, insert blocks, change settings, and verify the saved output. Covers the @wordpress/e2e-test-utils-playwright package.
- Performance: Code Splitting and Lazy Loading Blocks (20m)
  Large block libraries can slow down the editor. We implement dynamic imports, lazy-load block components, and use webpack code splitting to keep initial load times short.
- Memoization and Re-Render Prevention (18m)
  Blocks re-render frequently in the editor. We use React.memo, useMemo, and useCallback strategically to prevent unnecessary renders without over-optimizing.
- Accessibility in Block Editor UIs (22m)
  Your blocks must be usable by everyone. We cover ARIA labels, keyboard navigation, focus management, screen reader announcements, and testing with automated accessibility tools.
- Security Best Practices for Block Plugins (18m)
  Sanitizing user input in attributes, escaping output in PHP render callbacks, validating REST API requests, and preventing XSS in custom block controls.
- Packaging Your Block Library as a WordPress Plugin (20m)
  We prepare the plugin for distribution: proper file headers, readme.txt, asset optimization, uninstall cleanup, and submission checklist for the WordPress plugin directory.
- Publishing to npm and Managing Versions (15m)
  If other developers will use your blocks as a dependency, we cover npm publishing, semantic versioning for block libraries, and managing breaking changes in the block API.
- The Interactivity API and the Future of Block Development (20m)
  WordPress is moving toward the Interactivity API for front-end interactivity. We cover what it does, how it differs from attaching custom scripts, and when to adopt it.
- Tracking Block API Changes and the WordPress Roadmap (12m)
  Block development evolves with each WordPress release. We discuss how to track changes, handle deprecations, migrate blocks to newer API versions, and stay informed without constant effort.
- Capstone Project: Build and Deploy a Complete Block Library (40m)
  You bring everything together. Build a block library plugin with at least five interconnected blocks, dynamic rendering, shared components, tests, and deploy it to a live WordPress site.

## Discover our blog

Learn more with free articles written by our professors and guest experts.

Latest article: **Flight Attendant Preparation: What to Study Before You Apply**

Visit the blog  All courses  This course