Developing with WordPress

Needs Assessment

There are many courses about developing content with WordPress with different techniques and tools. Some of these courses conflict with each other or have not been updated to newer versions of WordPress where the techniques or the code have changed.

This course will cover several different way to build content with WordPress along with some auxiliary content that is necessary to prepare material to upload to your WordPress site.

Each module will have a companion code repository with both a beginning and final versions of the code so participants have a common starting point and a model to compare their work against.

```
TODO: Exact location TBD
```

Personas

The course is geared to different audiences with varying degrees of technical knowledge represented by the following personas:

Rebecca is a developer interested in WordPress but without much prior knowledge. She's created content with Laravel and Angular before.

Peter has developed with WordPress before using PHP in classic themes. He's had limited experience with React.

Bill has been working with Blocks since they were first introduced and wants to build better, customized blocks for their themes. He's proficient with React and with tools like `@wordpress/create-block`.

General Objectives

By the end of this course, participatns will be able to:

- Build a tooling system using Gulp and Javascript
- Explain the differences between classic and block themes
- Describe situations when you would use classic and block themes
- Identify cases when PHP is necessary to enhance block sites

Proposed modules

1. Tooling (Optional)
2. Setting up a local development server (Optional)
3. Classic Themes
4. Block Themes
5. Full Site Editing
6. Headless WordPress
7. Creating Blocks (Optional)
8. WordPress CLI (Optional)
9. Custom Post Types and how to make them work in Gutenberg

`TODO: Are the optional modules worth including as separate modules/courses or are they better of split out into separate content?`

`TODO: I'm not 100% sure that these are modules or a separate course. I'm keeping them together for now to make my life easier when writing them.`

Module 1: Tooling (optional)

Even though WordPress takes care of a lot of the tooling necessary to get auxiliary content like Javascript, CSS and image compression ready to use in a theme.

We will look at each of these three "buckets" and build a tooling workflow for them based on Gulp.js.

You are not required to build tooling for doing these things. They can all be done via the command line but automating the process will make it easier and be less time consuming and error prone.

For example. There is a lesson plan available on Learn WordPress working through the process of converting images to the WebP format: WebP images in WordPress

Module 1 Objectives

- Create a Gulp-based build and tooling system for auxiliary content for your theme
- Explain why we're creating a new build system rather than reusing the one from WordPress

Module 1 Outline

- Module assumptions and requirements
- Initializing the package
- Image compression
    - Why use this instead of WordPress image library or a third-party solution like Cloudinary?
    - Note about WebP support in WordPress
    - Using Libsquoosh
- Javascript
    - Why use Babel instead of letting browsers handle scripts natively?
    - Babel and preset-env (es2017)
- CSS
    - Why use PostCSS instead of using SCSS or native CSS?
    - PostCSS plus basic plugin ecosystem
- Extending the system
    - Possible extension points
    - Locating the plugins
    - Installing and configuring plugins

Module 2: Setting up a local development server (Optional)

The best way to work with WordPress is to install a local development server.

While the module will cover Local by Flywheel, there are other options that you're welcome to use if you're more comfortable or have already implemented them.

- XAMPP
- MAMP
- WampServer
- DevKinsta

I chose Local because it comes with Batteries ready. Every new site you create comes with the latest non-beta version of WordPress and the Twenty-Twentytwo theme ready to go.

Module 2 Objectives

By the end of this module, participants will be able to:

- Explain why a local development server is a better option than manually configuring a server
- Describe the different options available to run WordPress locally
- Import data into your local server

Module 2 Outline

- Introduction and module Overview
    - Available options
        - Web servers that require configuration
            - XAMPP
            - MAMP
            - WampServer
        - One step solutions
            - Local
            - DevKinsta
    - Why Local?
- Getting Started
    - Downloading Local
    - Installing Local
    - Creating sites with local
    - Loading demo data into your server
        - Theme Unit Test Data information and Github Repo
    - Create an export file from your production server
- Going under the hood
    - Site shell and WordPress CLI

Module 3: Classic Themes

Even though the entire WordPress world is working on Gutenberg, full site editing, and block themes, there are situations where PHP and classic themes are still necessary.

Module 3 Objectives

`TODO: Fix this. It definitely needs some polish and expansion`

- Explain what is a theme, how it works and why do we use it
- Describe the components needed for a classic theme
- List the minimal components required to build a theme
- Explain what are WordPress hooks and actions and give examples of each
- Describe what is the WordPress template hierarchy and how it is useful for developers
- Build a minimal classic theme
- Expand a theme with additional templates
    - Explain how to choose the templates we need
    - Create custom templates
    - Expalin when would you rely on the template hierarchy
- Prepare and enqueue resources for the theme

Module 3 Outline

`TODO: Fix this. It definitely needs some polish and expansion`

- Module Rationale and Overview
- What are themes? Why are they necessary?
-

Module 4: Block Themes

Block themes leverage Gutenberg to allow developers to create themes with little or no code.

Module 4 Objectives

- Describe the different components of a block theme

Module 4 Outline

- Module Overview and Rationale
- Components of a Block Theme
    - Patterns
    - Blocks
    - PHP files necessary to make the theme work with WordPress

        - theme.json configuration
    - Putting it all together
        - Creating the theme with minimal necessary files
    - Exporting the theme

Differences between classi and block themes:

> Differences and similarities between classic themes and block themes

Module 5: Full site editing

The introduction of block themes enabled the creation of entire sites on the client. This module will cover full site editing, how it works and the advantages and disadvatages of this approach.

Module 5 Objectives

Module 5 Outline

Module 6: Headless WordPress

We've covered many different ways to create content with WordPress. But they are all tightly coupled to the server we're working on. In this module we'll explore a way to decouple the backend from the code that shows the data by creating a headless WordPress application.

> Headless WordPress relies on the WordPress REST API

This module will use vanilla Javascript, the Handlebars template engine and a demo site. You can use any library or framework you want or have experience with.

> This is an advanced module and requires understanding of the WordPress backend and how to customize it.

Module 6 Objectives

- Explain what is a REST API
- Describe the WordPress REST API
- Create a basic example of using the API to retrieve the last 10 posts from a blog

Module 6 Outline

Module 7: Creating blocks (Optional)

`TODO: Should this be module 4, covered before we cover block themes or is it OK as final optional module?`

As supplementary material to learning how to work with block themes, let's take a look at what it takes to create blocks, the basic unit of block themes and full site editing.

This module will cover the basics of creating static and dynamic blocks and the SlotFills and toolbar interfaces to provide settings for the block.

Module 7 Objectives

- Describe the different types of blocks you can create
    - Explain when would you use which kind
- Build static and dynamic blocks manually
- Build static and dynamic blocks using @wordpress/scripts
- Understand and explain how to use SlotFills to add settings for your block
- Explain what's the difference between SlotFills and the Toolbar APIs
- Show how would you use third-party blocks
- Evaluate and use different ways to pack blocks for distribution

Module 8: WordPress CLI and comand line script libraries (Optional)

Module 8 Objectives

Module 8 outline

Module 9: Custom Post Types and how to make them work in Gutenberg

Module 9 Objectives

Module 9 outline