Build a Gutenberg Plugin from the Ground Up

Presented by Lee Gillentine
https://fishburd.com/
https://github.com/geetotes/wp_docker

Table of Contents

Development Environment Setup

https://github.com/geetotes/wp_docker

Docker

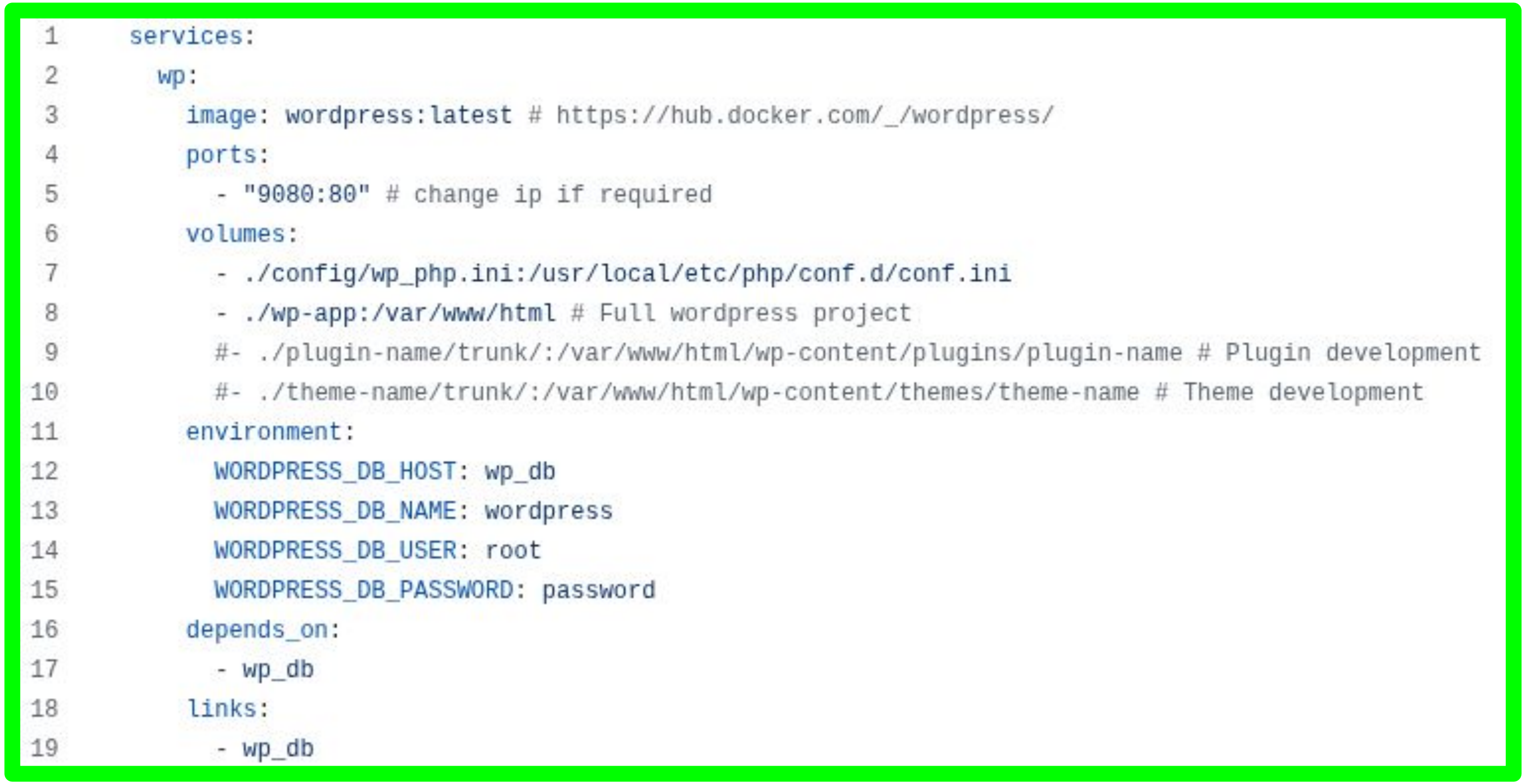
- Docker is a general purpose tool for containerized software
- Containers are kind of like “virtual machines”
    - Isolated environment that runs one application
    - Containers can share data through virtual interfaces
        - Network requests

        - Volumes

Docker Compose

- Containers are specified as “services” in a YAML file
  - `docker-compose.yml`
- Can specify
  - Service name
  - Ports (networking)
  - Volumes
  - Dependencies
  - Version of WordPress to use
- To bring up your dev environment, run `docker compose up`
  - This will automatically download container images and start your development environment
  - Access WordPress at http://localhost:9080
  - When you’re finished, Ctrl+C to exit
  - Run commands on individual containers with `docker compose run -rm <service name> <your command>`

Closer look at docker-compose.yml



```yaml
services:
  wp:
    image: wordpress:latest # https://hub.docker.com/_/wordpress/
    ports:
      - "9080:80" # change ip if required
    volumes:
      - ./config/wp_php.ini:/usr/local/etc/php/conf.d/conf.ini
      - ./wp-app:/var/www/html # Full wordpress project
      #- ./plugin-name/trunk/:/var/www/html/wp-content/plugins/plugin-name # Plugin development
      #- ./theme-name/trunk/:/var/www/html/wp-content/themes/theme-name # Theme development
    environment:
      WORDPRESS_DB_HOST: wp_db
      WORDPRESS_DB_NAME: wordpress
      WORDPRESS_DB_USER: root
      WORDPRESS_DB_PASSWORD: password
    depends_on:
      - wp_db
    links:
      - wp_db
```

What’s in the development environment?

`wp`

`wp_db`

`wpcli`

`pma`

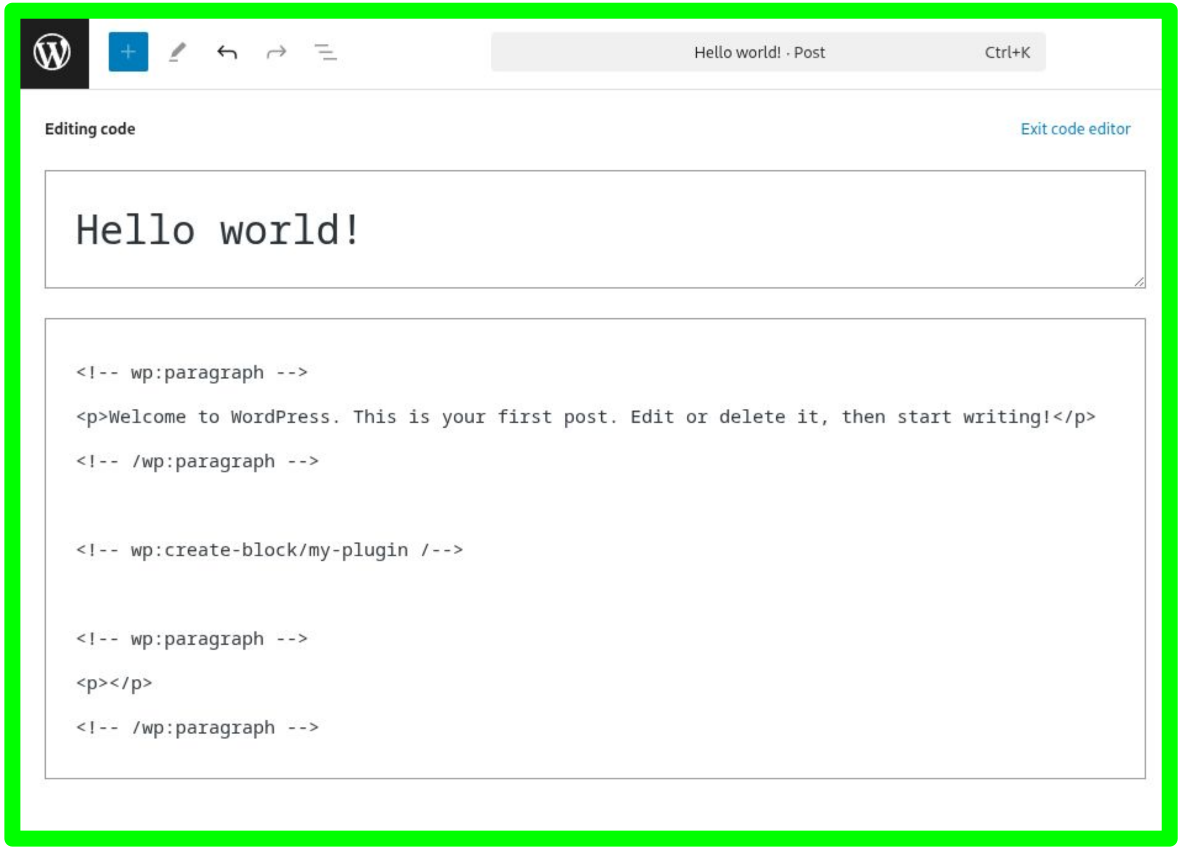
Plugin Creation

Plugin Creation

- Prerequisites:
    - Node.js installed
    - `npx` command available
- From outside docker, in the same directory, scaffold the plugin with
    - `npx @wordpress/create-block@latest my-plugin --variant=dynamic`
    - Note how `my-plugin` matches what's in the docker-compose.yml
        - This is the directory that will be used by `npx @wordpress/create-block`
- Now enable the plugin
    - Plugins -> My Plugin -> Activate
    - It's now available in the Gutenberg editor
- The Gutenberg editor plugin is a React component!

Ontology of blocks

- Use the code editor to see what's happening behind the scenes
- The newly created block type is represented by plain text
  - When the page is served by WordPress, the plain text is replaced by the output of the block's `render.php` or `view.js`



Hello world!

```
<!-- wp:paragraph -->
<p>Welcome to WordPress. This is your first post. Edit or delete it, then start writing!</p>
<!-- /wp:paragraph -->

<!-- wp:create-block/my-plugin /-->

<!-- wp:paragraph -->
<p></p>
<!-- /wp:paragraph -->
```

React Crash Course

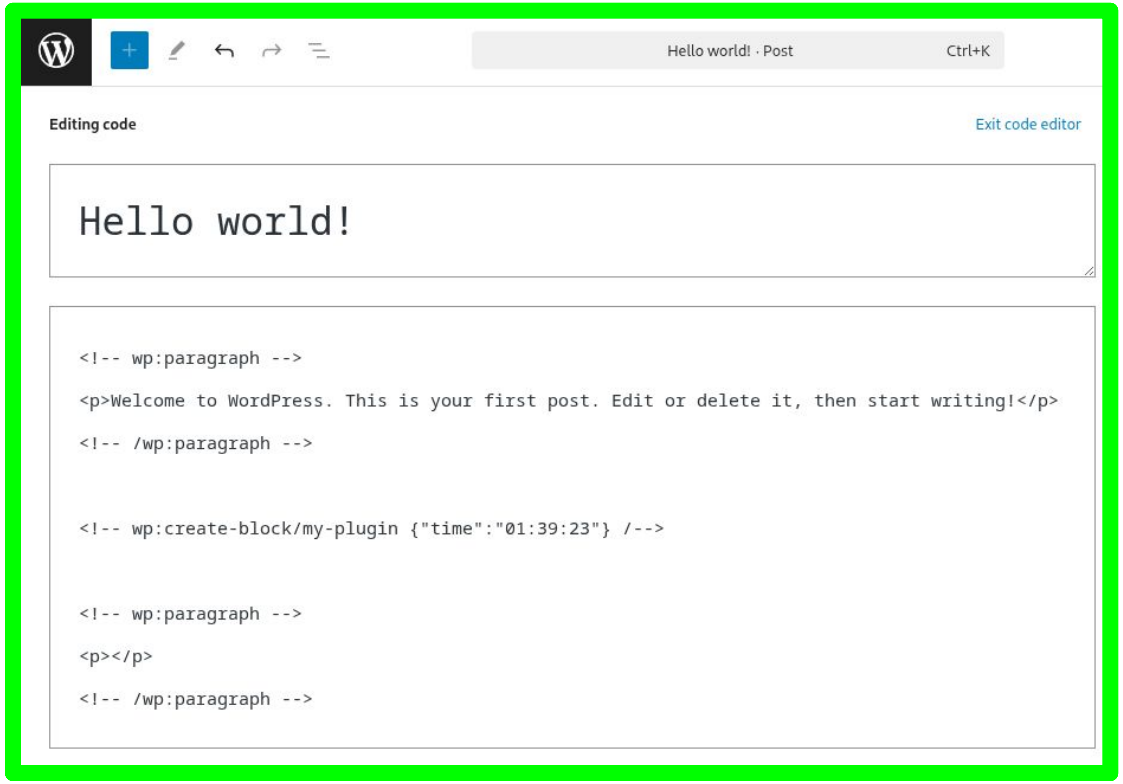
- React renders little chunks of HTML-like markup called JSX
- JSX components are composable – meaning they are encapsulated elements that can be nested inside each other and/or used wherever necessary, like HTML tags
    - React won the front-end because this approach made code reuse very easy
    - Components don't use inheritance. Instead, each component can define its functionality through props and hooks
- Many, many, many pre-built components and libraries are out there and can be imported into your plugin
    - Let's add some!

Call a 3rd party API from your block

- Every Gutenberg block plugin has a `package.json`
  - From your plugin directory, install axios (a library for HTTP requests)
    - `npm install axios`
  - Start dynamic re-complication of the plugin's javascript
    - `npm start`
  - Note: you can also use Javascript's `fetch` for HTTP requests, this is just an example of how to add npm libraries to your project
- Add a `useEffect` hook to call the 3rd party API and save the data in the component's state
- How do you save data to the block?

Block Attributes

- In `block.json`, you can define attributes
- In the Edit component, use `setAttributes` prop
- Attributes can be read when the block is displayed



Hello world!

```
<!-- wp:paragraph -->
<p>Welcome to WordPress. This is your first post. Edit or delete it, then start writing!</p>
<!-- /wp:paragraph -->

<!-- wp:create-block/my-plugin {"time":"01:39:23"} /-->

<!-- wp:paragraph -->
<p></p>
<!-- /wp:paragraph -->
```

Call the WordPress API from your block

- The `@wordpress/api-fetch` library provides an easy way to access your site's WordPress REST API
  - The REST API can access all content on your WordPress site: Posts, Tags, Users, Search, Plugins, etc
  - Secured with cookie authentication by default
  - Plugins can add additional endpoints to the REST API
- Add a `useEffect` hook to retrieve data from the REST API
  - The hook can leverage `apiFetch` to pull data from the site settings
    - Site settings is lightweight way for a plugin to store simple data, like an authentication key
- Let's see it in action!

Submitting Your Plugin

Submitting Your Plugin

- Make sure your plugin follows the:
  - Plugin Guidelines
  - Block Specific Guidelines
- Plugin banner, icon, and screenshots have specific naming conventions and are stored in assets folder
- For initial submission, zip up almost everything in the trunk folder

```
○  zip -r my-plugin.zip . -x '.*' -x 'node_modules/*'
```

- Submission review is not instant and takes about two weeks
- Once plugin is accepted, you’ll be given SVN access to push future updates

That’s all folks!

- https://github.com/geetotes/wp_docker
  - Give this repo a star if you liked the presentation
- Image Credits
  - Icons from Flaticon.com
    - And wikipedia for PhpMyAdmin

- Feel free to add me on LinkedIn: https://www.linkedin.com/in/leegillentine/