THE MODERN WORDPRESS PLUGIN DEVELOPMENT TOOLSET

(PART 1)

Josh Pollock | Plugin Machine

ABOUT ME

Josh Pollock | @josh412 | JoshPress.net

- PHP & JavaScript Engineer
- Working on Plugin Machine
- WordPress core contributor
- ex: 10up, Ninja Forms, Caldera Forms, Pods.

WHO THIS IS FOR

- Plugin developers
- Anyone who writes PHP and JavaScript for WordPress

- View Slides

WHAT WE WILL COVER

- What WordPress is made of
- What we use to make WordPress

- View Slides

SLIDES AND CODE

- View Slides
- Source Code For Slides
- https://github.com/imaginarymachines/everything-all-of-the-time

THE WORDPRESS STACK

- Client (Browser)
  - JavaScript/ CSS/ HTML
- Server
  - PHP
- Database
  - MySQL

REACT IN WORDPRESS

- Used in block editor
- Can be used for:
  - Admin pages
  - Front-end
  - Headless builds

WHY DEVELOPER TOOLING FOR REACT?

- React is generally written in JSX, not JavaScript.
- Optimize for browser.
- Automated testing

DEVELOPER TOOLING

- IDE
- Local Development Environment
- Dependency Management
- Automated Testing
- Compiling and Optimizing JS(X) and (SCSS)
- Code Quality
- CI/ CD

IDE

INTEGRATED DEVELOPER ENVIRONMENT

- Code editor plus other development tools.

IDE

SOME GOOD ONES

- vsCode
  - What I use.
- phpStorm
  - What I used to use.
- Atom

LOCAL DEVELOPMENT

WHY

- Isolation
  - Don't want to break live site
- Common setup for all developers on a project
  - Automated configuration
  - Consitency

LOCAL DEVELOPMENT

SOLUTIONS

- Docker
  - docker-compose
    - WP Local Docker
    - Lando
    - @wordpress/env
- WordPress-specific GUI Apps
  - Desktop Server
  - LocalWP
- Virtual Machine
  - vvv
  - Homestead

LOCAL DEVELOPMENT

MY RECOMMENDATION: DOCKER COMPOSE

- Runs anywhere
  - Mac/ Windows/ Linux
  - Github Actions
  - vsCode dev containers
  - In the cloud
- Config as code
- Infinetly extendable

DOCKER COMPOSE

BASIC

```
version: "3.9"

services:

  wordpress:
    depends_on:
      - wpdb
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
      - ./:/var/www/html/wp-content/plugins/everything-all-of-the-time
    ports:
      - "6100:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: wpdb:3306
```

DOCKER COMPOSE

WITH WPCLI AND TESTS

```
version: "3.9"

services:

  wordpress:
    depends_on:
      - wpdb
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
      - ./:/var/www/html/wp-content/plugins/everything-all-of-the-time
    ports:
      - "6100:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: wpdb:3306
```

DEPENDENCY MANAGEMENT

- For PHP: Composer
- For JavaScript: npm or yarn

COMPOSER

- Add packages to your plugin or site
  - Use dependencies from packagist
  - Install plugins or themes using wpackagist
- Run scripts
- PHP autoloader

COMPOSER

INSTALL

- Install Composer
- For Windows, download installer.
- For Mac/ Linux, follow latest instructions
  - sudo mv composer.phar /usr/local/bin/composer

COMPOSER

BASIC COMPOSER.JSON

```
"name": "imaginary-machines/everything-all-of-the-time",
"description": "Example of Plugin Machine generated plugin with many features on.
"type": "wordpress-plugin",
"require": {
    "php": "^7.2|^8.0"
}
```

COMPOSER

COMPOSER.JSON WITH AUTOLOADER

```
{
    "autoload": {
        "psr-4": {
            "AllOfTheThings\\": "./php"
        }
    },
    "autoload-dev": {
        "psr-4": {
            "AllOfTheThings\\Tests\\": "./tests"
        }
    }
}
```

- Using PHP Namespaces And Autoloaders In WordPress Plugins.

NPM (OR YARN)

WHY

- Add additional functionality
  - For example: component library, API client, utility functions, etc.
- Compile code so it can run in the browser and optimize it
  - Babel, webpack, etc.
- Run scripts
- Configure Packages

NPM (OR YARN)

INSTALL

- npm is installed with node
  - Install Node
  - Install yarn: `npm install -g yarn`

PACKAGE.JSON

BASIC

```
{
    "name": "@imaginary-machines/everything-all-of-the-time",
    "private": true,
    "version": "0.1.0",
    "license": "GPL-2.0-or-later",
    "main": "build/index.js",
    "scripts": {
        "build": "wp-scripts build",
        "lint:css": "wp-scripts lint-style",
        "lint:js": "wp-scripts lint-js",
        "start": "wp-scripts start"
    },
    "devDependencies": {
        "@wordpress/scripts": "^16"
    }
}
```

AUTOMATED TESTS

```php
function add($one, $two){
  return $one + $two;
}

class SampleTest extends \PHPUnit\Framework\TestCase
{

    public function testAdd()
    {
        $this->assertEquals(
          4, add(2,2)
        );
    }
```

AUTOMATED TESTS

- Guide development.
- Prevent regression errors.
- Makes clear how the code is supposed to work.
- Debugging errors.
- Profiling performance.
- Surfacing accessibility issues

PHPUNIT

- PHP test runner and assertions.
- Can be used with or without WordPress test suite.

AUTOMATED TESTS

COMPOSER.JSON FOR PHPUNIT

```
{
    "require-dev": {
        "phpunit/phpunit": "^7.0",
        "yoast/phpunit-polyfills": "^0.1.0",
        "mockery/mockery": "1.2",
        "brain/monkey": "2.*",
        "squizlabs/php_codesniffer": "^3.6"
    },
    "scripts": {
        "test:unit": "phpunit --config=phpunit-unit.xml",
        "test:wordpress": "phpunit --config=phpunit-integration.xml"
    }
}
```

AUTOMATED TESTS

EXAMPLE UNIT TEST

```php
class Url {
  protected $baseUrl;
  public function __construct($baseUrl)
  public function getUrl($endpoint){
    return sprintf("%s/%s",$this->baseUrl,$endpoint);
  }
}
class UrlTest extends \PHPUnit\Framework\TestCase
{

    public function testGetUrl()
    {
        $url = new Url('https://hiroy.club');
        $this->assertEquals(
          'https://hiroy.club/api',
```

AUTOMATED TESTS

EXAMPLE UNIT TEST WITH MOCK

- Brain Monkey

```
xtends \PHPUnit\Framework\TestCase

t we can mock WordPress functions

ps://giuseppe-mazzapica.gitbook.io/brain-monkey/functions-testing-tools/functions-

tion testMockWordPressFunction(){
e wp_insert_post() that always returns 1
ns\when('wp_insert_post' )->justReturn(1);
assertIsNumeric(
```

AUTOMATED TESTS

EXAMPLE INTEGRATION TEST WITHOUT MOCK

```
class SomethingTest extends \PHPUnit\Framework\TestCase
{

    public function testInsertPost(){
        $post_id =  wp_insert_post([
          'post_title' => 'Succulents',
          'post_content' => 'lithops and echeveria'
        ]);
        $this->assertIsNumeric($post_id);
    }
```

AUTOMATED TESTS

```
function add($one, $two){
  return $one + $two;
}

expect(add(2,2)).toBe(4);
```

WordPress Scripts

- Compiles
  - WordPress-safe.
  - Works with `wp_register_script/style()`
- Lints
- Runs tests
  - Uses and configures Jest.

COMPILING JAVASCRIPT

WHY

- Allows for alternative syntaxes
  - JSX
- Optimize JavaScript
- Optimize CSS

JSX

- React's template syntax
- Not required or blocks
  - Worth using for sure.
- HTML and JavaScript together

JSX

```
//React component with JSX
function Alert({message}){
    return (
        <div>
            <p className="alert">
            {message}</p>
        </div>
    );
}
```

Renders as:

```
<div>
    <p class="alert">
        Hi Roy
    </p>
</div>
```

JSX

```
//React component with JSX
function Alert({message}){
    return (
        <div>
            <p className="alert">
            {message}</p>
        </div>
    );
```

Is the same as:

```
//React component without JSX
function Alert({message}){
    return React.createElement( 'div', {},[
        React.createElement( 'p', {
            className: 'alert'
        }, [ message ])
```

TESTING JAVASCRIPT

- Use Jest, via WordPress Scripts
- Generally need a React testing library in adittion to Jest.
  - I recommend @testing-library/react
- Slides For My JavaScript Testing Talk

TESTING JAVASCRIPT

PACKAGE.JSON WITH TESTS

```
{
    "name": "@imaginary-machines/everything-all-of-the-time",
    "private": true,
    "version": "0.1.0",
    "license": "GPL-2.0-or-later",
    "main": "build/index.js",
    "scripts": {
        "test": "yarn test:unit",
        "test:unit": "wp-scripts test-unit-js",
        "build": "wp-scripts build",
        "test:ci": "wp-scripts test-unit-js --ci",
        "format:js": "wp-scripts format-js",
        "lint:css": "wp-scripts lint-style",
        "lint:js": "wp-scripts lint-js",
        "start": "wp-scripts start"
    },
```

TESTING JAVASCRIPT

EXAMPLE

```
//Import React
import React from 'react';
//Import test renderer
import { render, fireEvent, cleanup } from '@testing-library/react';
//Import component to test
import { Editor } from './Edit';


describe("Editor componet", () => {
    afterEach(cleanup);
    it('matches snapshot when selected', () => {
        const onChange = jest.fn();
        const { container } = render(<Editor
            onChange={onChange}
            value={'Tacos'}
```

CODE QUALITY ANALYSIS

- Tests check code by running it in various ways.
- Code quality analysis parses the code and detetchs likely issues.

CODE QUALITY ANALYSIS

- Linters
  - Does code follow coding standards?
  - Are bad smelling patterns used?
  - Is the code using deprecated functions?
- Static Analysis
  - Are variable types correct?
  - Predicts compile-time errors.

CODE QUALITY ANALYSIS

- Linters
  - Code Sniffer
    - WordPress Standards For Code Sniffer
    - PHPCompatibilityWP
- Static Analysis
  - PHP Stan
    - WordPress Extension for PHP Stan

CI/CD

WHAT

- CI: Continuous integration.
  - Using automtion to continously test, analysis and merge changes to code.
- CD: Continous deployment.
  - Using automation to continously deploy code.

CI/CD

SERVICES

- Github Actions
- Circle CI
- Gitlab CI
- BranchCI
- Buddy Works

CI/ CODE

EXAMPLE GITHUB ACTION

```
HP Unit Tests

h


s-on: ubuntu-latest
ategy:
atrix:
 php-versions: [ 7.2, 7.3, 7.4 ]

ps:
 uses: actions/checkout@v2
 name: Setup PHP
 uses: shivammathur/setup-php@v2
```

LIVE DEMO TIME

- View Slides
- Source Code For Slides
- https://github.com/imaginarymachines/everything-all-of-the-time

THANK YOU

Josh Pollock | @josh412 | JoshPress.net

- Plugin Machine