Testing 101:

How to Bulletproof Your Deployments with Automated Code Testing

Chris Wiegman

https://chriswiegman.com | @ChrisWiegman | http://wieg.co/wcmia20

Find today's slides at:

http://wieg.co/wcmia20

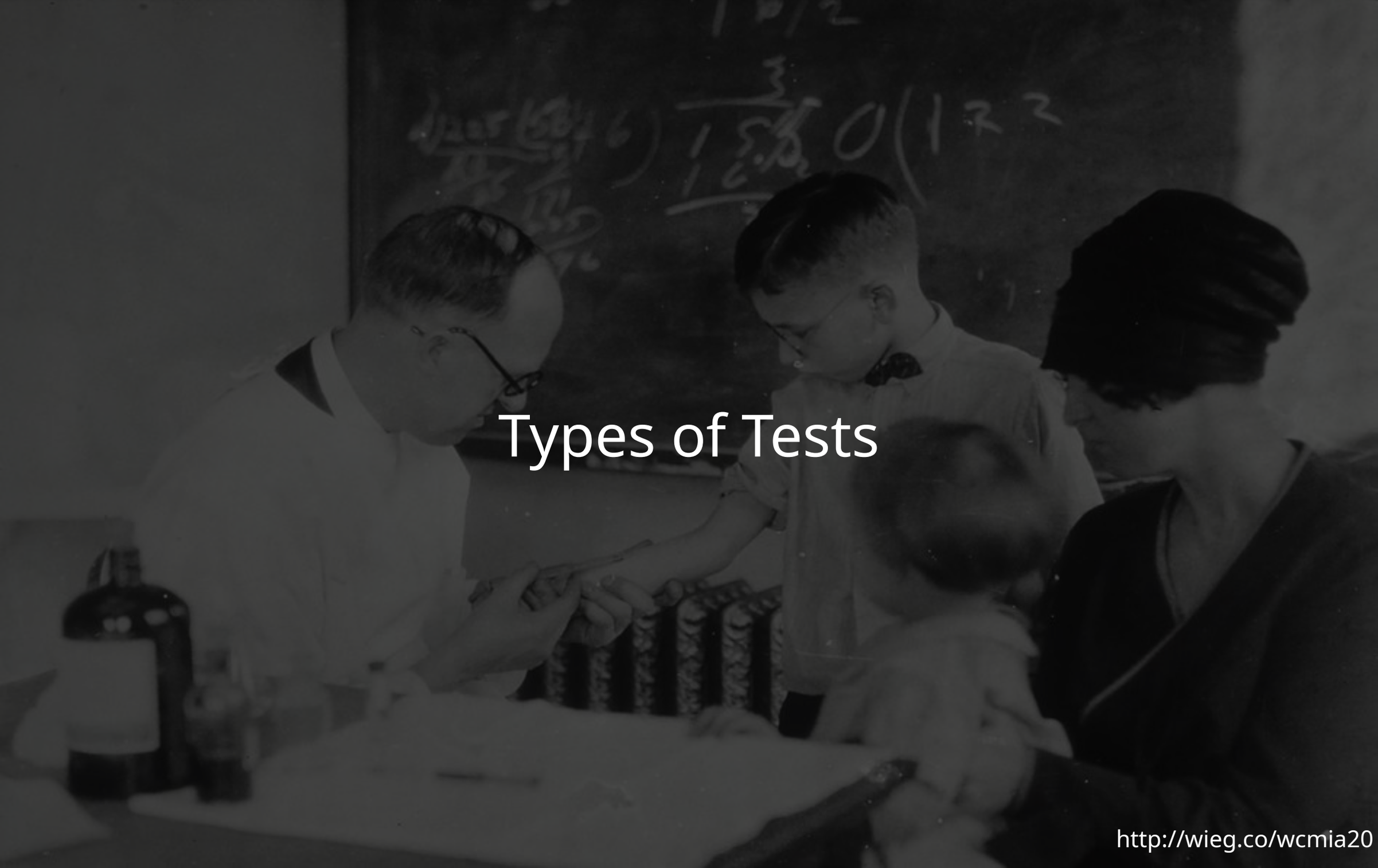
About Me

- Senior Software Engineer – WP Engine
- Speaker, Teacher, Blogger, Pilot
- Focus on
  - Privacy
  - Development Workflows
  - The Open Web

http://wieg.co/wcmia20

Why Automated Testing

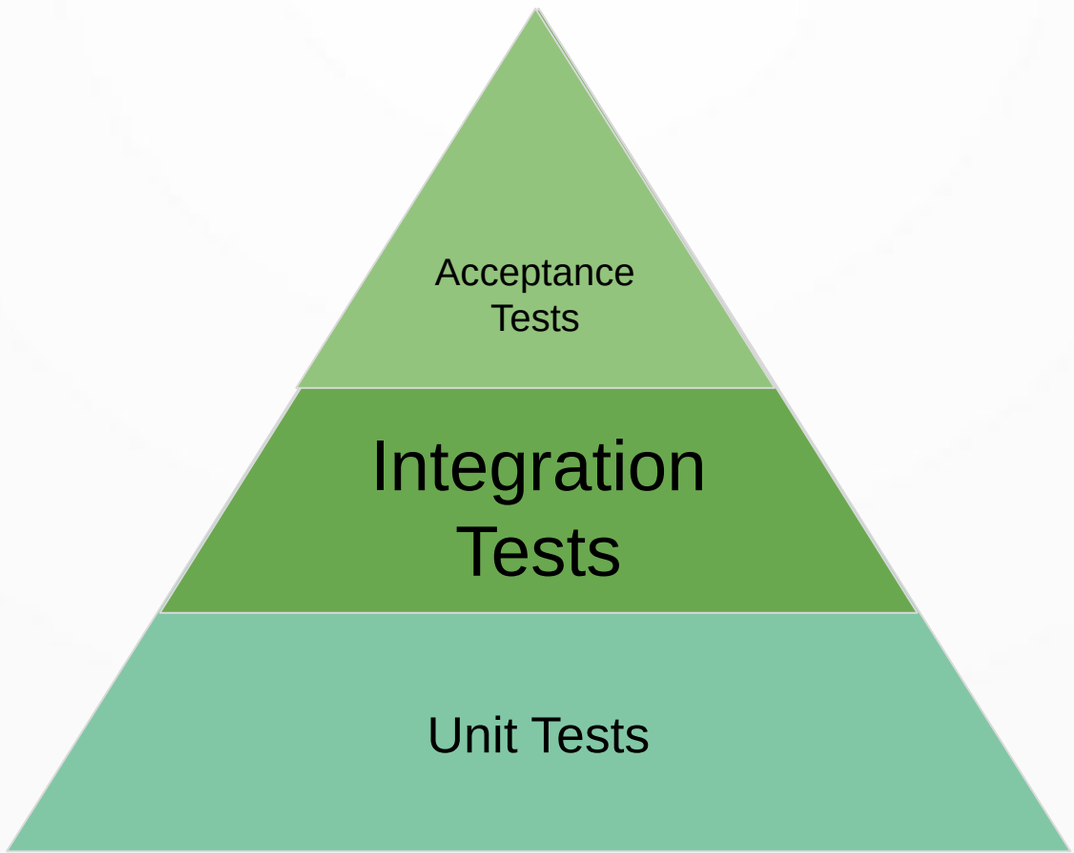
- Prevent regressions
- Can prevent bad code from ever reaching the server
- Safer refactoring
- Automated Q/A can cover the basics humans don’t need to spend time on
- Higher-quality code



Types of Tests

http://wieg.co/wcmia20

The Testing Pyramid



http://wieg.co/wcmia20

Unit Tests

- Prevent regressions
- Can prevent bad code from ever reaching the server
- Safer refactoring
- Automated Q/A can cover the basics humans don’t need to spend time on
- Higher-quality code

Integration Tests

- Tests groups of functions in aggregate
- Can be most difficult to test in WordPress backend
- Test groups of components or other functionality

Acceptance Tests

- Test the product is acceptable
- Tests the whole product
- Often done with external libraries

Linting

- Tests individual lines of codes against standards
- Doesn’t look at the output but how the code is written
- Can look at:
  - Syntax
  - Best practices (security/performance/etc)
  - Invalid code

Who needs testing?

Does Ever Project Need Testing?

Yes

Does Ever Project Need Testing?

No

http://wieg.co/wcmia20

Does Ever Project Need Testing?

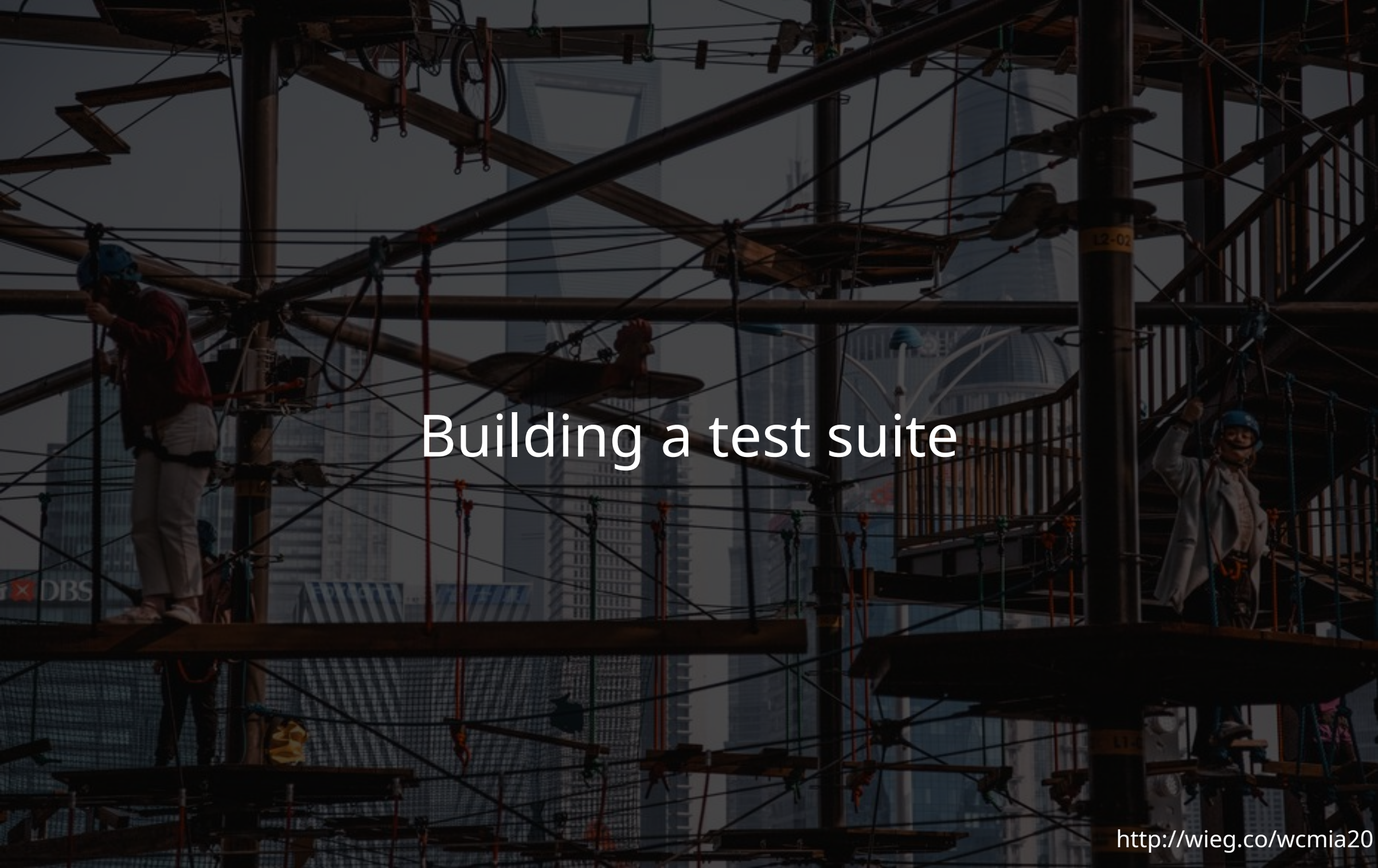
- Even smallest projects have to be tested to meet the customers requirements
- ROI on testing is often recognized over time
- Full test suites not justifiable on “throw-away” projects

The best time to start a testing plan is when you start your project.

http://wieg.co/wcmia20

The 2nd best time to start a testing plan is today.

http://wieg.co/wcmia20



Building a test suite

http://wieg.co/wcmia20

What Tests Should You Implement?

- How long will your project last?
- What tests are appropriate?
- Are you starting from scratch or inheriting code?

Every project can benefit from linting.

After you know what tests you need to keep your product/project healthy, you can determine the budget.

Linting

Installing Linters: PHP

- PHP_CodeSniffer - https://github.com/squizlabs/PHP_CodeSniffer
- WordPress Coding Standards - https://github.com/WordPress/WordPress-Coding-Standards
- Set your IDE's rule or PHP_CodeSniffer to "WordPress"

```
----------------------------------------------------------------------------------------

FOUND 8 ERRORS AND 10 WARNING AFFECTING 11 LINES

----------------------------------------------------------------------------------------

 24 | WARNING | [ ] error_reporting() can lead to full path disclosure.

 24 | WARNING | [ ] error_reporting() found. Changing configuration at runtime is rarely

    |         |     necessary.

 37 | WARNING | [x] "require_once" is a statement not a function; no parentheses are

    |         |     required
```

Installing Linters: JavaScript

- *npm -g install eslint*, turn on option in editor
- Add to package.json*

```
{
  "name": "mypackage",
  "version": "0.0.1",
  "eslintConfig": {
    "env": {
      "browser": true,
      "node": true
    }
  }
}
```

http://wieg.co/wcmia20

Linter Gotchas

- Existing/old code will fail… a lot
- Tune to as strict as *practical*
- You don’t need all the rules
- Looks at the lowest hanging fruit, doesn’t care about your logic (or lack thereof)
  - Might be enough to get your code through an interview… and get you in over your head

http://wieg.co/wcmia20

Unit Testing: PHP

- Install PHPUnit via https://phpunit.de/

./phpunit --bootstrap src/autoload.php tests

PHPUnit 9.0.0 by Sebastian Bergmann and contributors.

... 3 / 3 (100%)

Time: 70 ms, Memory: 10.00MB

OK (3 tests, 3 assertions)

Code

src/Email.php

```php
<?php declare(strict_types=1);
final class Email
{
    private $email;

    private function __construct(string $email)
    {
        $this->ensureIsValidEmail($email);

        $this->email = $email;
    }

    public static function fromString(string $emai
    {
        return new self($email);
    }

    public function __toString(): string
    {
        return $this->email;
    }

    private function ensureIsValidEmail(string $em
    {
        if (!filter_var($email, FILTER_VALIDATE_EMA
            throw new InvalidArgumentException(
                sprintf(
                    '"%s" is not a valid email addr
                    $email
                )
            );
        }
```

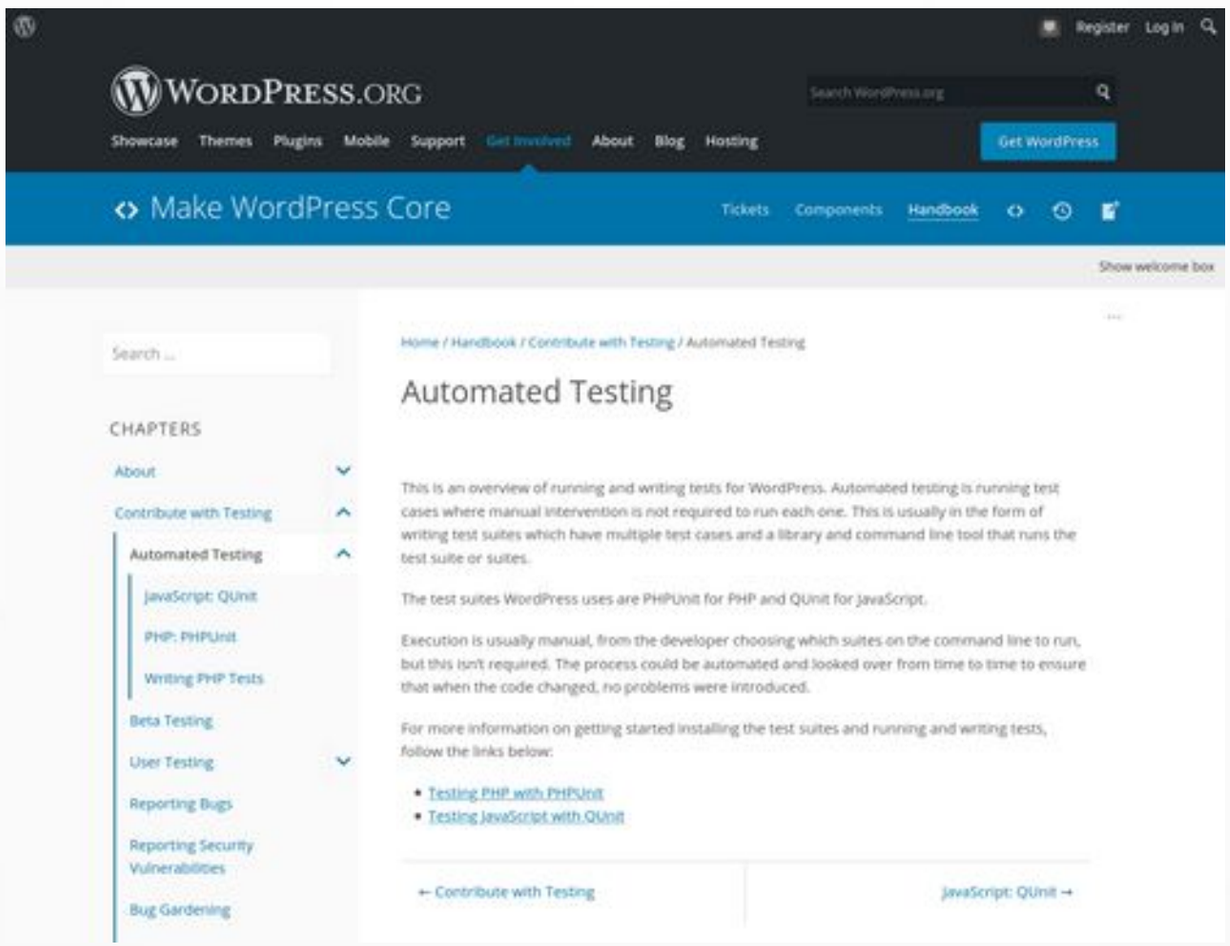
Test Code

tests/EmailTest.php

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class EmailTest extends TestCase
{
    public function testCanBeCreatedFromValidEmail
    {
        $this->assertInstanceOf(
            Email::class,
            Email::fromString('user@example.com')
        );
    }

    public function testCannotBeCreatedFromInvalid
    {
        $this->expectException(InvalidArgumentExcep

        Email::fromString('invalid');
    }

    public function testCanBeUsedAsString(): void
    {
        $this->assertEquals(
            'user@example.com',
            Email::fromString('user@example.com')
        );
    }
}
```

http://wieg.co/wcmia20

WordPress Testing Suite



https://make.wordpress.org/core/handbook/testing/automated-testing/

http://wieg.co/wcmia20

Adding Unit Tests to a Plugin

- Setup testing: *wp scaffold plugin-tests my-plugin*
- Install test suite: *bash bin/install-wp-tests.sh wordpress_test root '' localhost latest*
- Run the tests: *phpunit*

Testing on Windows? Look at https://make.wordpress.org/cli/handbook/plugin-unit-tests/

Unit Testing: JavaScript

- *npm install -g qunit*
- Create tests in a directory. ie. tests/qunit
- Run qunit
  - *qunit 'tests/qunit/*'*

Qunit Example

- tests.js

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>QUnit Example</title>
  <link rel="stylesheet" href="https://code.jquery.com/qunit/qunit-2.9.2.css">
</head>
<body>
  <div id="qunit"></div>
  <div id="qunit-fixture"></div>
  <script src="https://code.jquery.com/qunit/qunit-2.9.2.js"></script>
  <script src="tests.js"></script>
</body>
</html>
```

Qunit Example

- tests.js

```
QUnit.test( "hello test", function( assert ) {
  assert.ok( 1 == "1", "Passed!" );
});
```

Keys to Unit Testing

- Functions should have explicit input and output
- Assertions are key
- Not every function requires a unit test
- Testable code is key

Writing Testable Code

- Classes shouldn’t instantiate classes. Use factories
- Ask for things, don’t look for things – dependency injection
- Constructors…. Construct
- Global state (and singletons) aren’t very testable
- Inheritance != code re-use
- Polymorphism over conditionals
- *Can you isolate the function?*

http://wieg.co/wcmia20

Acceptance Testing

http://wieg.co/wcmia20

Acceptance testing is testing the whole application (website/CLI command/etc).

Keys of Acceptance Testing

- Needs the application to be complete (doesn't look at small parts)
- Runs the full application in a known good state to create **snapshots**
- Compare snapshots to determine problems

WP Acceptance

- 10up Project for easy acceptance tests in WordPress
- Full documentation: https://wpacceptance.readthedocs.io/en/latest/

```
{
    "environment_instructions": [
        "install wordpress where site url is http://wpacceptance.test and home url is http://wpacceptance.test",
        "install theme where theme name is twentynineteen"
    ]
}
```

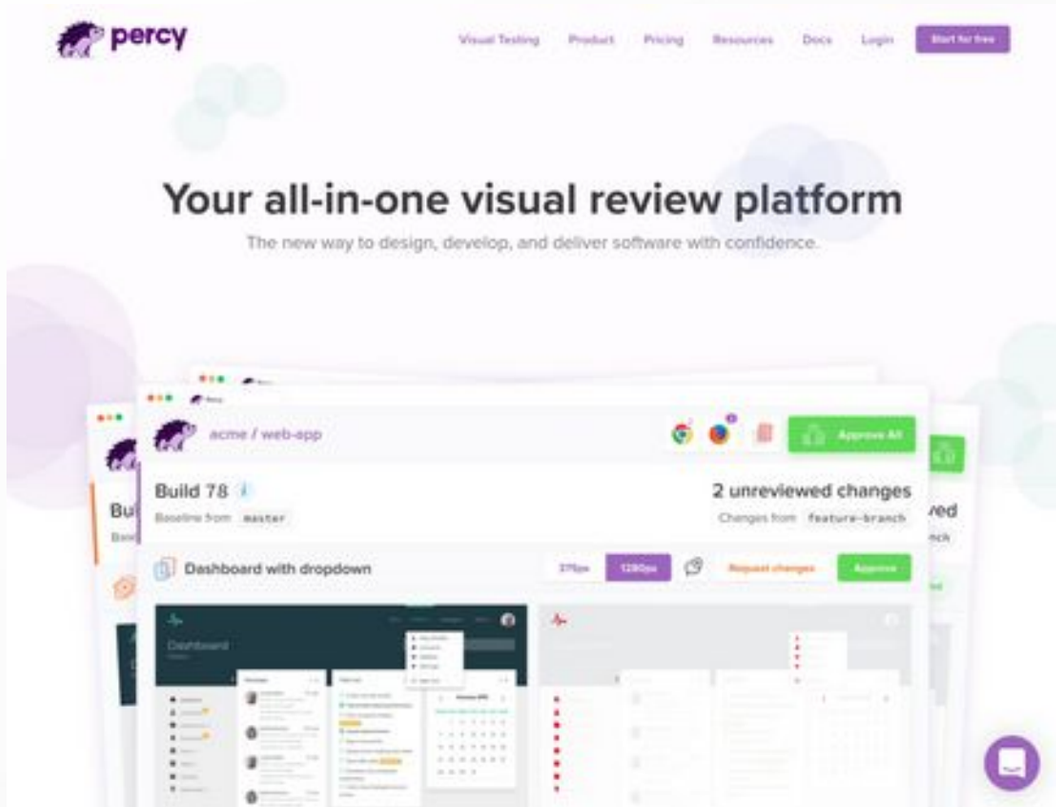
Jest

- https://jestjs.io/
- Tests CLI and other apps (great for WP-CLI code)
- JavaScript Based

```
describe('lcltun', () => {
    it('returns the correct output when called with no options.', async () => {
        const { stdout } = await exec('lcltun');
        expect(stdout).toMatchSnapshot();
    });

    describe('version', () => {
        it('checks for appropriate version strings', async () => {
          const { stdout } = await exec('lcltun version');
          expect(stdout).toContain("Version:")
          expect(stdout).toContain("Git Commit Hash:")
          expect(stdout).toContain("UTC Build Time:")
        });
    });
});

describe('lclserve', () => {
    it('returns the correct output when called with no options.', async () => {
        const { stdout } = await exec('lclserve');
        expect(stdout).toMatchSnapshot();
    });

    describe('version', () => {
        it('checks for appropriate version strings', async () => {
          const { stdout } = await exec('lclserve version');
          expect(stdout).toContain("Version:")
          expect(stdout).toContain("Git Commit Hash:")
          expect(stdout).toContain("UTC Build Time:")
        });
    });
});
```

Percy

- https://percy.io/
- Compares screenshots during CI steps
- Easily integrate with your repo



Review and References

http://wieg.co/wcmia20

What About Integration Tests

- Often written using PHPUnit or similar unit test library. Can also be written like acceptance tests.
- Might test whole of plugin functionality (a whole feature, perhaps) without looking at the full application.
- Implementation varies greatly.
- Often Unit or Acceptance tests are actually Integration tests

Which Testing to Choose?

- Your project might not need all testing types
- Unit tests ***assert*** Acceptance tests ***snapshot***
- Acceptance tests are often easier to start with for mature projects
- Unit tests, with appropriate coverage, will go further to limit regressions
- ***The only bad test is the one not written.***

http://wieg.co/wcmia20

Further Reading

- https://eslint.org/
- https://github.com/squizlabs/PHP_CodeSniffer
- https://phpunit.de/
- https://qunitjs.com/
- https://github.com/10up/wpacceptance
- https://make.wordpress.org/core/handbook/testing/automated-testing/
- https://testing.googleblog.com/2008/08/by-miko-hevery-so-you-decided-to.html
- https://en.wikipedia.org/wiki/Test-driven_development

http://wieg.co/wcmia20

http://chriswiegman.com | @ChrisWiegman | http://wieg.co/wcmia20