Cypress testing steps

1) for bare-bones setup: `npm install cypress --save-dev`

(for the `wordpress-digimod` example, do `npm install` with this `package.json`)

```
{
  "devDependencies": {
    "@faker-js/faker": "^7.6.0",
    "@frsource/cypress-plugin-visual-regression-diff": "^3.2.14",
    "@testing-library/cypress": "^9.0.0",
    "@wildpeaks/snapshot-dom": "1.6.0",
    "axios": "^1.4.0",
    "check-more-types": "2.24.0",
    "cypress": "^12.10.0",
    "cypress-email-results": "^1.8.0",
    "cypress-file-upload": "^5.0.8",
    "cypress-plugin-api": "^2.10.3",
    "diff": "^5.1.0",
    "js-beautify": "^1.13.13",
    "lazy-ass": "1.6.0",
    "snap-shot-compare": "3.0.0",
    "snap-shot-store": "1.2.3",
    "typescript": "^5.0.2",
    "url-slug": "^3.0.4",
    "xml2js": "^0.5.0"
  }
}
```

> NOTE: The snapshot plugin for Cypress library had undergone a transition into a new project and at the date of construction of this test set the plugin was not in a stable configuration. To address issues with the plugin, it has been cloned locally and modified to address an issue where comparison between actual and expected snapshots was not working properly (crashing with recursion limit).

> See https://github.com/bcgov/wordpress-digimod/tree/main/testing/lib/snapshot for this forked version of the Cypress plugin called @datashard/snapshot.

- add this to your `cypress.config.js` file to import it for use: `require("./lib/snapshot/src").tasks(on, config);`

> instructions for use of this plugin: https://github.com/datashard/snapshot

---

2) get config file built in root directory for cypress: `cypress.config.js`

> easy coniguration via the cypress UI: https://docs.cypress.io/guides/getting-started/opening-the-app#Quick-Configuration

> see: https://docs.cypress.io/guides/end-to-end-testing/testing-your-app#Step-3-Configure-Cypress

- digimod example: https://github.com/bcgov/wordpress-digimod/blob/main/testing/cypress.config.js#L0-L1
  - don't forget to add this import: `require("./lib/snapshot/src").tasks(on, config);`

> this file contains some functions for reading site URLs for comparison, which isn't strictly necessary, since we could just hard-code an object with the urls we'd want to use, e.g.: http://localhost:8889/block-theme-testing-site , which is what you might set up locally if you were

- generally, cypress will want some configuration key/value pairs for its testing steps:

```
  defineConfig({ e2e: { async setupNodeEvents(on, config) {
    // add your urls, import plugins, init your plugins,
      config.env.sitemapUrls = urls.newUrls; //
      // ...other config info to feed to cypress in its test file

    return config;
    } }}, component: { setupNodeEvents(on, config)}})
```

---

3) set up a test file for cypress to know where the snapshots are and how to compare them

- I recommend starting with the cypress docs when setting up e2e testing specs: https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test
- example: https://github.com/bcgov/wordpress-digimod/blob/1fcf85bb08e35a39d64a935809de1230a6c7b6d8/testing/cypress/e2e/sitemap-snapshot-test.cy.js#L87-L93

> the snapshot part is the only kind of e2e test we want, so look at the cy.wrap section on L89 of the above example, where it performs the comparison between the generated 'existing' file, and the generated 'new' file.

```
cy.wrap({ html: someRenderedHtmlForSnapshot })
  // use the snapshot plugin to compare
  .snapshot("testName",
     {
       // these config values could probably be hard-coded if we're only
testing on snapshot
      snapshotName: 'name-of-snapshot-file', // overrides default filename
      snapshotPath: '/location/of/snapshot/file',  // overrides default
path
      json:true // use json format for storing this snapshot file
     })
     // can wrap more than one json file and test it: `cy.wrap({
```

```
jsonFileSnapshot})
```

> - To use the snapshot plugin, see these instructions on how it should be invoked in your test file: https://github.com/bcgov/wordpress-digimod/blob/main/testing/lib/snapshot/README.md

> in the example, pay special attention to the `afterEach()` callback used in the `runTests()` function.
>
> - it waits for the current test to fail
> - it will generate three snapshot files per url: the expected, the actual, and the diff.

> runTests() will iterate through all the urls to test, and is in turn called as the callback for the only actual test in the file: `describe('snapshot-test', () => main())`

- main is just a wrapper that takes the site URLs and invokes `runTests(someSiteUrls)`

> for more info on how to write e2e tests for cypress: https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test

example snapshot comparison:

```
// cypress/integration/login.spec.js

describe('Login Form', () => {
  it('should match the saved snapshot', () => {
    cy.visit('/login'); // Navigate to the login page

    // Capture a snapshot of the login form
    cy.get('form').snapshot('login-form');

    // Perform login actions (e.g., enter credentials, click login button)

    // After login, capture another snapshot
    cy.get('form').snapshot('login-form-after-login');
  });
});
```

4) run the test: (pass cypress your test file's location and the test site's location)

```
npx cypress run --spec "path-to-snapshot-test-file.js" --browser firefox -
-env-url=http://localhost:8889/someTestSiteUrl
```

5) include extra steps for automation, such clicking elements

- see https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test#Step-3-Click-an-element
- before trying this, please walk through the above steps, looking at the digimod example config and spec.
- The cypress docs tutorial on getting started will help open a running version of cypress.
    - Follow these steps:
    1. https://docs.cypress.io/guides/getting-started/opening-the-app
    2. https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test