# ☀️ INSTALLATION AND CONFIGURATION

## 📚 YOU WILL LEARN

- Installing Playwright
- Installing Cypress
- Configuration and project options
- Documentation

# QUICK CHECK: NODE.JS VERSION

```
$ node -v
v20.11.0
$ npm -v
10.2.4
```

If you need to install Node, see Basics Requirements and 📹 Install Node and Cypress

# QUICK CHECK: VERSIONS

You can use available-versions NPM package to quickly list the published package versions.

```
$ npx available-versions cypress
...
13.13.3                     a month
13.14.0                     a month
13.14.1                     25 days
13.14.2                     19 days     latest, dev

$ npx available-versions playwright
...
1.47.1                      10 days
1.47.2                      3 days      latest
```

**Try it:** check the current latest Cy and Pw versions

# RELEASE FREQUENCY

- Cypress https://on.cypress.io/changelog
- Playwright https://github.com/microsoft/playwright/releases

# TODO: SCAFFOLD PLAYWRIGHT

- clone repo `https://github.com/bahmutov/cy-vs-pw-example-todomvc`

In the folder with bahmutov/cy-vs-pw-example-todomvc repo

```
$ git checkout a1
$ npm install
$ npm init playwright@latest
```

Pick folder pw for E2E tests and use JavaScript for the specs.

# TODO: INSPECT THE CREATED FILES

Playwright installation creates:

- `playwright.config.js`
- `pw/example.spec.js`
- `tests-examples/demo-todo-app.spec.js`

Let's look at each file 👀

🤔 Have any other files been modified?

# TODO: USE JUST THE CHROMIUM BROWSER

- modify the `playwright.config.js` to run the tests on the single bundled Chromium browser

# TODO: TEST THE LOCAL APP USING PLAYWRIGHT

**Important:** start the application in the separate terminal

💡 Start Playwright in UI mode `npx playwright test --ui` while modifying the spec file.

- modify the `pw/example.spec.js`
- have a single test that:
  1. visits `localhost:3000`
  2. confirms the page title

**Tip:** look up test command by `npx playwright help` and `npx playwright <command> help`

# PLAYWRIGHT TEST

```
// pw/example.spec.js
const { test, expect } = require('@playwright/test')

test('has title', async ({ page }) => {
  await page.goto('http://localhost:3000/')

  // Expect a title "to contain" a substring.
  await expect(page).toHaveTitle('cy-vs-pw-example-todomvc')
})
```

**Question:** do you get IntelliSense when hovering over `test` and `expect`?

# REPORT VS TRACE

- run the test and look at the test report
- run the test with a trace and look at the test report

# TIP: CLEANUP

Before going to the next step, here is how to clean up modified files:

```
# remove changes
$ git reset --hard
# remove new / untracked files
$ git clean -d -f
```

💡 You can make a shell alias

```
alias gnuke="git reset --hard && git clean -df"
```

# TODO: INSTALL CYPRESS

Install Cypress in the project

- `git checkout a2`
- `npm install -D cypress`

# HOW TO OPEN CYPRESS

```
npx cypress open
# or
yarn cypress open
# or
$(npm bin)/cypress open
# or
./node_modules/.bin/cypress open
```

# SCAFFOLD CYPRESS FILES

Open Cypress once using `npx cypress open`

Inspect the created files

# CYPRESS FILES AND FOLDERS

- "cypress.config.js" - all Cypress settings
- "cypress/e2e" - end-to-end test files (specs)
- "cypress/fixtures" - mock data
- "cypress/support" - shared commands, utilities

Read blog post Cypress is just ...

# 💡 PRO TIP

```
# quickly scaffolds Cypress folders
$ npx @bahmutov/cly init
# bare scaffold
$ npx @bahmutov/cly init -b
# typescript scaffold
$ npx @bahmutov/cly init --typescript
```

Repo github.com/bahmutov/cly and 📝 blog post Cypress vs Playwright Installation.

# FIRST CYPRESS SPEC

Modify the spec `cypress/e2e/spec.cy.js`

```
// @ts-check
/// <reference types="cypress" />

it('has title', () => {
  // visit the page "localhost:3000"
  // https://on.cypress.io/visit
  // the page title should have text "cy-vs-pw-example-todomvc"
  // https://on.cypress.io/title
})
```

💡 Run the test while editing the spec with `npx cypress open`

**Important:** start the application in the separate terminal

# RUN CYPRESS TEST

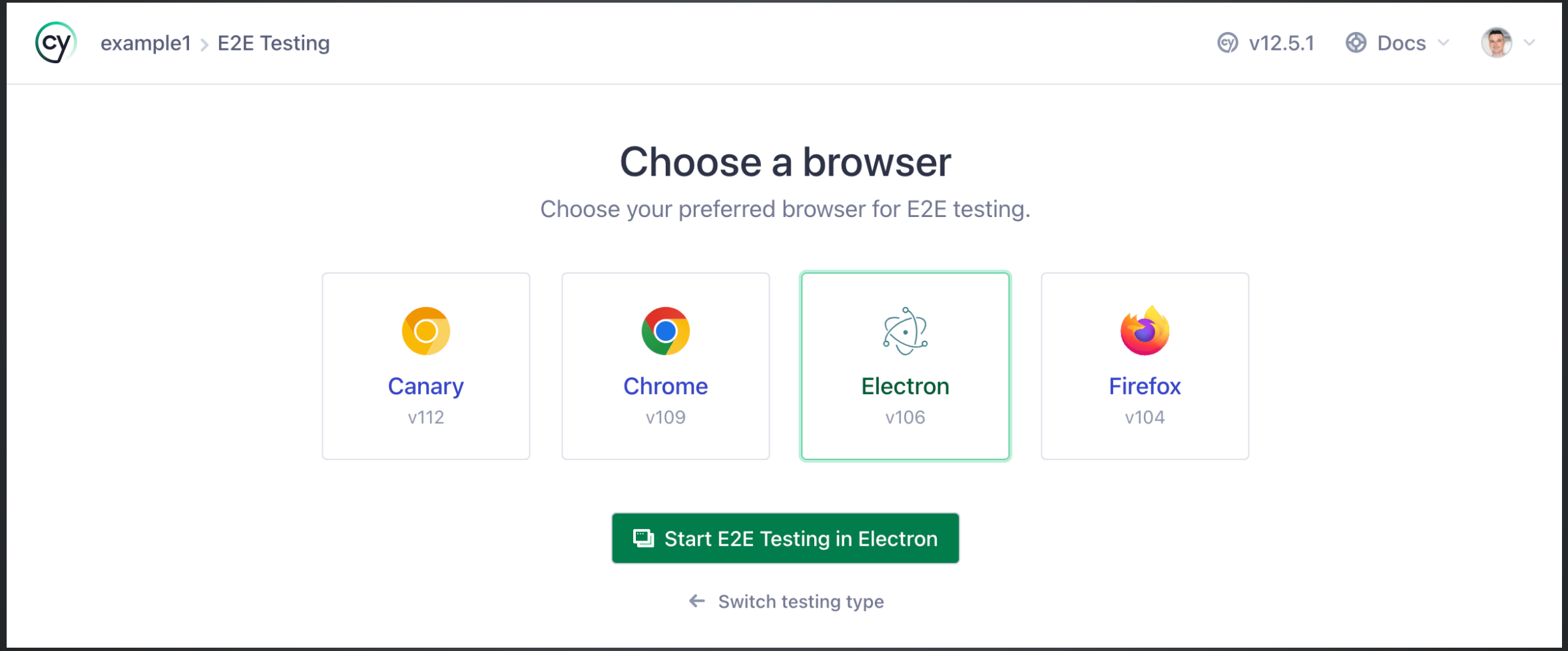
Cypress is really geared towards either working on the specs or running them headlessly

```
$ npx cypress open
$ npx cypress run
$ npx cypress help <command>
```

Run your spec headlessly.

🤔 How do you see what the test is doing?

# CYPRESS: PICK A BROWSER



# ENABLE VIDEOS

```
const { defineConfig } = require('cypress')

module.exports = defineConfig({
  e2e: {
    video: true
  }
})
```

💡 Screenshots are taken automatically on failures

# INTELLISENSE

- look at the `///` comment. This tells your code editor about Cypress globals like `cy`
- the comment `// @ts-check` tells your code editor to show any type mismatches in the specs

**Question:** why do we need to "explain" to Cypress the types? How is this different from Playwright? Take a look at https://github.com/bahmutov/local-cypress

# DOCS AND HELP

- Cypress documentation is at https://docs.cypress.io/
- Playwright documentation is at https://playwright.dev/docs/intro

When starting, read the introductions and the guides. Then consult the API docs as necessary.

# CHAT AND SUPPORT

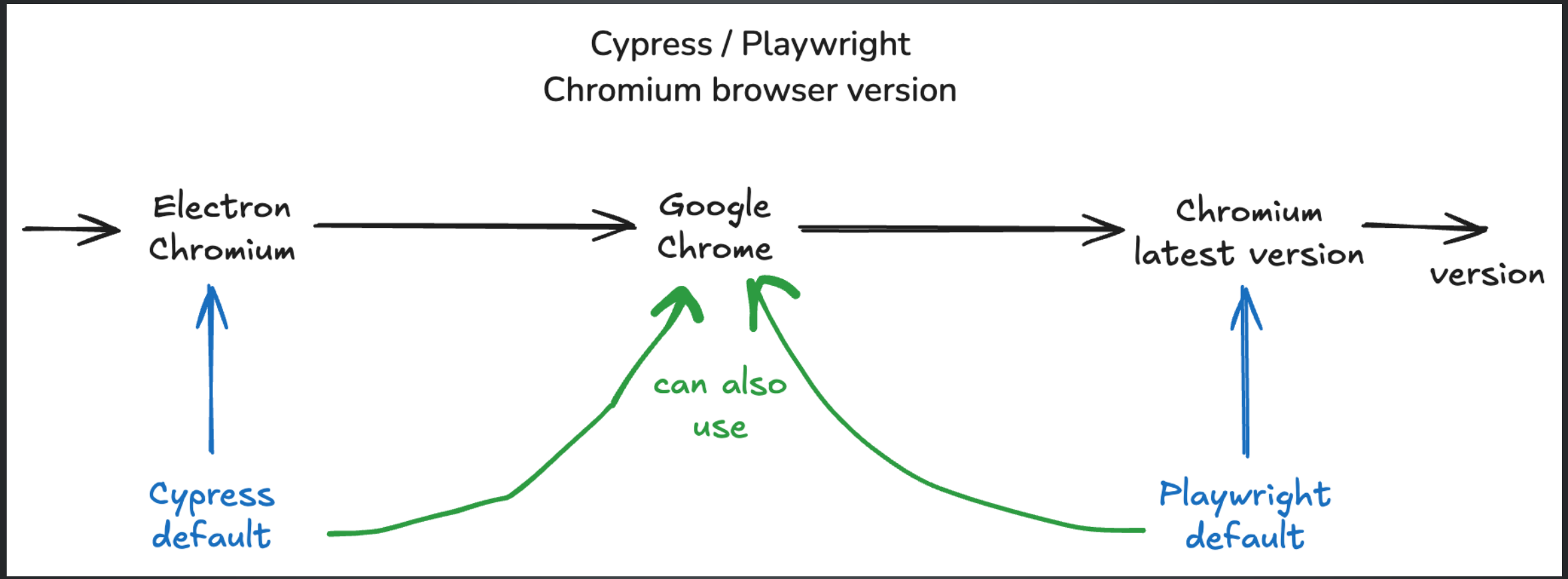
- https://aka.ms/playwright/discord
- https://on.cypress.io/discord

# PRO TIP

```
https://on.cypress.io/<command>
```

The above URL goes right to the documentation for that command.

# PLAYWRIGHT VS CYPRESS BROWSERS



# 🏁 CONCLUSIONS

- use the config file
- which browser?
- headed vs headless mode
- documentation is there

➡️ Pick the next section or jump to the 01-basic chapter