Test configuration

Playwright has many options to configure how your tests are run. You can specify these options in the configuration file. Note that test runner options are **top-level**, do not put them into the `use` section.

Basic Configuration

Here are some of the most common configuration options.

```
import { defineConfig, devices } from '@playwright/test';

export default defineConfig({
  // Look for test files in the "tests" directory, relative to this
configuration file.
  testDir: 'tests',

  // Run all tests in parallel.
  fullyParallel: true,

  // Fail the build on CI if you accidentally left test.only in the source
code.
  forbidOnly: !!process.env.CI,

  // Retry on CI only.
  retries: process.env.CI ? 2 : 0,

  // Opt out of parallel tests on CI.
  workers: process.env.CI ? 1 : undefined,

  // Reporter to use
  reporter: 'html',

  use: {
    // Base URL to use in actions like `await page.goto('/')`.
    baseURL: 'http://localhost:3000',

    // Collect trace when retrying the failed test.
    trace: 'on-first-retry',
  },
  // Configure projects for major browsers.
  projects: [
    {
      name: 'chromium',
      use: { ...devices['Desktop Chrome'] },
    },
  ],
  // Run your local dev server before starting the tests.
  webServer: {
    command: 'npm run start',
    url: 'http://localhost:3000',
    reuseExistingServer: !process.env.CI,
  },
});
```

Option	Description
testConfig.forbidOnly	Whether to exit with an error if any tests are marked as test.only. Useful on CI.
testConfig.fullyParallel	have all tests in all files to run in parallel. See Parallelism and Sharding for more details.
testConfig.projects	Run tests in multiple configurations or on multiple browsers
testConfig.reporter	Reporter to use. See Test Reporters to learn more about which reporters are available.
testConfig.retries	The maximum number of retry attempts per test. See Test Retries to learn more about retries.
testConfig.testDir	Directory with the test files.
testConfig.use	Options with use{}
testConfig.webServer	To launch a server during the tests, use the webServer option
testConfig.workers	The maximum number of concurrent worker processes to use for parallelizing tests. Can also be set as percentage of logical CPU cores, e.g. '50%'.. See Parallelism and Sharding for more details.

Filtering Tests

Filter tests by glob patterns or regular expressions.

playwright.config.ts
```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  // Glob patterns or regular expressions to ignore test files.
  testIgnore: '*test-assets',

  // Glob patterns or regular expressions that match test files.
  testMatch: '*todo-tests/*.spec.ts',
});
```

Option	Description
testConfig.testIgnore	Glob patterns or regular expressions that should be ignored when looking for the test files. For example, '*test-assets'
testConfig.testMatch	Glob patterns or regular expressions that match test files. For example, '*todo-tests/*.spec.ts'. By default, Playwright runs .*(test|spec).(js|ts|mjs) files.

Advanced Configuration

playwright.config.ts

```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  // Folder for test artifacts such as screenshots, videos, traces, etc.
  outputDir: 'test-results',

  // path to the global setup files.
  globalSetup: require.resolve('./global-setup'),

  // path to the global teardown files.
  globalTeardown: require.resolve('./global-teardown'),

  // Each test is given 30 seconds.
  timeout: 30000,

});
```

Option	Description
testConfig.globalSetup	Path to the global setup file. This file will be required and run before all the tests. It must export a single function.
testConfig.globalTeardown	Path to the global teardown file. This file will be required and run after all the tests. It must export a single function.
testConfig.outputDir	Folder for test artifacts such as screenshots, videos, traces, etc.
testConfig.timeout	Playwright enforces a timeout for each test, 30 seconds by default. Time spent by the test function, test fixtures and beforeEach hooks is included in the test timeout.

Expect Options

Configuration for the expect assertion library.

playwright.config.ts
```
import { defineConfig } from '@playwright/test';

export default defineConfig({
  expect: {
    // Maximum time expect() should wait for the condition to be met.
    timeout: 5000,

    toHaveScreenshot: {
      // An acceptable amount of pixels that could be different, unset by
default.
      maxDiffPixels: 10,
    },

    toMatchSnapshot: {
      // An acceptable ratio of pixels that are different to the
      // total amount of pixels, between 0 and 1.
      maxDiffPixelRatio: 0.1,
    },
  },

});
```

Option	**Description**
testConfig.expect	Web first assertions like expect(locator).toHaveText() have a separate timeout of 5 seconds by default. This is the maximum time the expect() should wait for the condition to be met. Learn more about test and expect timeouts and how to set them for a single test.
expect(page).toHaveScreenshot()	Configuration for the expect(locator).toHaveScreenshot() method.
expect(value).toMatchSnapshot()	Configuration for the expect(locator).toMatchSnapshot() method.