

# Emulation

## Introduction

With Playwright you can test your app on any browser as well as emulate a real device such as a mobile phone or tablet. Simply configure the devices you would like to emulate and Playwright will simulate the browser behavior such as `"userAgent"`, `"screenSize"`, `"viewport"` and if it `"hasTouch"` enabled. You can also emulate the `"geolocation"`, `"locale"` and `"timezone"` for all tests or for a specific test as well as set the `"permissions"` to show notifications or change the `"colorScheme"`.

## Devices

Playwright comes with a [registry of device parameters](#) using `playwright.devices` for selected desktop, tablet and mobile devices. It can be used to simulate browser behavior for a specific device such as user agent, screen size, viewport and if it has touch enabled. All tests will run with the specified device parameters.

**Sync**   **Async**

---

```
from playwright.sync_api import sync_playwright, Playwright

def run(playwright: Playwright):
    iphone_13 = playwright.devices['iPhone 13']
    browser = playwright.webkit.launch(headless=False)
    context = browser.new_context(
        **iphone_13,
    )

    with sync_playwright() as playwright:
        run(playwright)
```

# Viewport

The viewport is included in the device but you can override it for some tests with `page.set_viewport_size()`.

Test file:

The same works inside a test file.

**Sync**   **Async**

---

```
# Create context with given viewport
context = browser.new_context(
    viewport={ 'width': 1280, 'height': 1024 }
)

# Resize viewport for individual page
page.set_viewport_size({"width": 1600, "height": 1200})

# Emulate high-DPI
context = browser.new_context(
    viewport={ 'width': 2560, 'height': 1440 },
    device_scale_factor=2,
)
```

## isMobile

Whether the meta viewport tag is taken into account and touch events are enabled.

**Sync**   **Async**

---

```
context = browser.new_context(
    isMobile=false
)
```

## Locale & Timezone

Emulate the user Locale and Timezone which can be set globally for all tests in the config and then overridden for particular tests.

**Sync**

**Async**

```
context = browser.new_context(  
    locale='de-DE',  
    timezone_id='Europe/Berlin',  
)
```

## Permissions

Allow app to show system notifications.

**Sync**

**Async**

```
context = browser.new_context(  
    permissions=['notifications'],  
)
```

Allow notifications for a specific domain.

**Sync**

**Async**

```
context.grant_permissions(['notifications'], origin='https://skype.com')
```

Revoke all permissions with `browser_context.clear_permissions()`.

**Sync**

**Async**

```
context.clear_permissions()
```

# Geolocation

Grant `"geolocation"` permissions and set geolocation to a specific area.

**Sync**    **Async**

---

```
context = browser.new_context(  
    geolocation={"longitude": 41.890221, "latitude": 12.492348},  
    permissions=["geolocation"]  
)
```

Change the location later:

**Sync**    **Async**

---

```
context.set_geolocation({"longitude": 48.858455, "latitude": 2.294474})
```

**Note** you can only change geolocation for all pages in the context.

## Color Scheme and Media

Emulate the users `"colorScheme"`. Supported values are 'light', 'dark', 'no-preference'. You can also emulate the media type with `page.emulate_media()`.

**Sync**    **Async**

---

```
# Create context with dark mode  
context = browser.new_context(  
    color_scheme='dark' # or 'light'  
)
```

```
# Create page with dark mode  
page = browser.new_page(  
    color_scheme='dark' # or 'light'  
)
```

```
# Change color scheme for the page
page.emulate_media(color_scheme='dark')
```

```
# Change media for page
page.emulate_media(media='print')
```

# User Agent

The User Agent is included in the device and therefore you will rarely need to change it however if you do need to test a different user agent you can override it with the `userAgent` property.

**Sync**   **Async**

---

```
context = browser.new_context(
    user_agent='My user agent'
)
```

# Offline

Emulate the network being offline.

**Sync**   **Async**

---

```
context = browser.new_context(
    offline=True
)
```

# JavaScript Enabled

Emulate a user scenario where JavaScript is disabled.

**Sync**

**Async**

```
context = browser.new_context(  
    javaScript_enabled=False  
)
```