



Pruebas de aplicaciones web con Puppeteer

Puppeteer para comprobación de existencia, navegación, tamaño, velocidad y otras métricas.

"Hacer las pruebas de existencia y rendimiento al terminar el desarrollo o tras las pruebas funcionales es como tomar el pulso o hacer una analítica a un paciente que ya está muerto."

-- 🖋️ Scott Barber

Puppeteer: Pruebas básicas.

Preparado y automatizado

Sencillo

Barato

Frecuente

Puppeteer y Node

- Un script Node que tome como dependencia a [Puppeteer](#).
- Con los comandos `assert` propios de Node.
- Estructura con la *triple A* **Arrange-Act-Assert**.

```
"dependencies": {  
  "lighthouse": "^6.0.0",  
  "puppeteer": "^3.1.0",  
  "request": "^2.88.2"  
},
```

Arrange

- Configurar y preparar: *Arrange*

```
async function arrangeBrowser() {  
  console.info(`arranging browser`);  
  const browser = await puppeteer.launch({  
    headless: true,  
    defaultViewport: { width: 1920, height: 1080 }  
  });  
  const pagePuppet = await browser.newPage();  
  return { browser, pagePuppet };  
}
```

- Intento lo que debería ocurrir

```
module.exports = async function itShouldExist(pagePuppet) {  
  let errors = 1;  
  const inputPageUrl = 'https://www.bitademy.com';  
  console.info(`GIVEN the url: ${inputPageUrl}`);  
  try {  
    console.info(`  WHEN is visited`);  
    await pagePuppet.goto(inputPageUrl, { waitUntil: 'networkidle2' });  
    console.info(`    THEN it Should Exist a page: ${inputPageUrl}`);  
    errors = 0;  
  } catch (error) {  
    console.warn({ error });  
  }  
  assertTrue(errors == 0, `Could not visit the url: ${inputPageUrl}`);  
  return errors;  
};
```

Given, when, then.

```
async function itShouldHaveTitle(pagePuppet) {  
  console.info(`GIVEN a page`);  
  const expected = 'bitAdemy';  
  ...  
}
```



Act

- Actuar para obtener la realidad: `actual` .

```
async function itShouldHaveTitle(pagePuppet) {  
  ...  
  console.info(`  WHEN we get its title`);  
  const actual = await actGetTitle(pagePuppet);  
  ...  
}  
async function actGetTitle(pagePuppet) {  
  return await pagePuppet.title();  
}
```



Assert

Comprobar que algo es cierto `assert` .

```
async function itShouldHaveTitle(pagePuppet) {
  ...
  console.info(`    THEN it Should Have Title: ${expected}`);
  return assertEquals(actual, expected);
}
function assertEquals(actual, expected) {
  try {
    assert.strictEqual(actual, expected);
    console.info(`     SUCCESS`);
    return 0;
  } catch (error) {
    console.info(`     FAIL: expected ${error.expected} but got ${error.actual} `);
    return 1;
  }
}
```


After

- Limpiar al terminar: `after` .

```
async function afterAll(browser, numErrors) {  
  await browser.close();  
  if (numErrors) {  
    console.warn(` FAIL: there are ${numErrors} site errors`);  
  } else {  
    console.info(` SUCCESS: all tests completed successfully`);  
  }  
  process.exit(numErrors);  
}
```

Interacción

- Simular acciones de usuario
 - Navegación
 - Escritura
 - Click
- Si se complica... [Cypress](#)

Un ejemplo de suscripción a una newsletter

```
module.exports = async function itShouldAllowSubscribe(pagePuppet) {  
  let errors = 1;  
  console.info(`GIVEN a page with a subscribe form `);  
  try {  
    console.info(`  WHEN we select the input `);  
    await actSelect(pagePuppet, '#MERGE0');  
    console.info(`  AND WHEN we type at the selected input `);  
    await actType(pagePuppet, 'puppet@bitademy.com');  
    console.info(`  AND WHEN we click on the subscribe button `);  
    await actClick(pagePuppet, '#subscribe-form > button');  
    console.info(`    THEN it Should Allow Subscribe`);  
    errors = 0;  
  } catch (error) {  
    console.warn({ error });  
  }  
  assertTrue(errors == 0, `Could not complete subscribe process`);  
  return errors;  
};
```

```
async function actSelect(pagePuppet, selector) {  
  await pagePuppet.evaluate(function (selector) {  
    return document.querySelector(selector).scrollBy(0, 10);  
  }, selector);  
  await pagePuppet.focus(selector);  
}  
  
async function actType(pagePuppet, value) {  
  await pagePuppet.keyboard.type(value);  
}  
  
async function actClick(pagePuppet, selector) {  
  await pagePuppet.click(selector);  
}
```

Imagen

- Capturar instantáneas y guardarlas
- Distintas resoluciones o simuladores de dispositivos.

```
module.exports = async function takeScreenshot(pagePuppet) {  
  const timeStamp = new Date().getTime();  
  const shotPath = path.join(process.cwd(), 'images', `${timeStamp}.png`);  
  await pagePuppet.screenshot({  
    path: shotPath,  
    fullPage: true  
  });  
};
```

En [el laboratorio](#) tienes más ejemplos de lo que es capaz *Puppeteer*. Y si aún quieres más puede mirar este otro repositorio aún más completo [AtomicBuilders/muon](#)