**Testing Browser Alerts, Confirmations, and Prompts with Cypress**

A major component of building for the web is providing our customers feedback as they’re navigating and using our apps. The browser provides native ways to do that including alerts, confirmations, and prompts, but how can we use automation tools like Cypress to test those experiences?

**What are alerts, confirmations, and prompts?**

While developers tend to create custom solutions for providing a notice to someone visiting their website, all browsers actually come with a few methods right out of the box to handle this for you including alerts, confirmations, and prompts.

They all provide similar functionality but vary slightly in the feedback requested from the site visitor.

**Alerts:** triggers a message dialogue with a single action allowing the user to accept and close the dialogue.

alert('This is an alert!');

**Confirmation:** triggers a message dialogue but provides two actions allowing the user to accept or cancel the dialogue, both closing after an action has been taken.

confirm('Would you like to confirm?');

**Prompt:** triggers basically the same thing as a confirmation, except there’s an additional text input, for the visitor to return a message

prompt('Please enter your confirmation.');

All of the options are somewhat similar, but provide a different set of actions which can be useful for different situations.

**Challenges with testing some native browser features**

The issue with trying to test these native browser features is they’re not available in the DOM (Document Object Model). Cypress and other testing frameworks take advantage of JavaScript’s ability to query selectors.

For instance, if I wanted to select a button on the page with Cypress, I might run:

cy.get('#my-button')

While the Cypress get method is a bit more complex, we’re essentially looking for that element by a selector, similar to:

document.querySelector('#my-button');

When dealing with the alerts, confirmations, and prompts, we don’t have a way to select these elements.

**Alerts, confirmations, and prompts in Cypress**

To get around this limitation, we can instead listen to the events that these dialogues emit to make sure our notifications are working as expected.

For example, if we wanted to listen for the alert dialogue, we could add an event listener in Cypress like:

cy.on('window:alert', callback);

While this method is still somewhat limited and it varies a bit between the dialogue types, it should give us enough functionality to regain confidence in our tests by allowing us to confirm the text as well as confirming the event actually occurred.

Let’s dig into the code and see how each of these would look in practice.

**Testing alerts in the browser with Cypress**

Similar to what we saw above, we can test our alerts using code similar to the following:

|  |  |
| --- | --- |
|  | it('should trigger an alert with a message', () => { |
|  | cy.get('#alert-button').click(); |
|  |  |
|  | cy.on('window:alert', (text) => { |
|  | expect(text).to.contains('This is an alert!'); |
|  | }); |
|  | }); |

[**view raw**](https://gist.github.com/dannyshain/937618cf7b2a2ce09b163c6616e1ba7f/raw/9559503ff5980359a7c1dc5b6d089fb58f94c051/alerts.js)[**alerts.js**](https://gist.github.com/dannyshain/937618cf7b2a2ce09b163c6616e1ba7f#file-alerts-js) hosted with ❤ by [**GitHub**](https://github.com/)

After first triggering the alert, we’ll listen to the window:alert event. When it’s triggered, Cypress will pass the argument text containing the alert’s message to the callback function, which we can then write an assertion against.

**Testing confirmation in the browser with Cypress**

Testing a confirmation dialogue will be similar to our alert:

|  |  |
| --- | --- |
|  | it('should trigger a confirmation with a message', () => { |
|  | cy.get('#confirm-button').click(); |
|  |  |
|  | cy.on('window:confirm', (text) => { |
|  | expect(text).to.contains('Would you like to confirm?'); |
|  | }); |
|  |  |
|  | cy.get('#confirm-answer').contains('Answer: Yes'); |
|  | }); |

[**view raw**](https://gist.github.com/dannyshain/abd0816859d11e22facd2dbfb6fb69f3/raw/fc8df0e6018d8f47f70be8a286c73128fee53090/confirmation.js)[**confirmation.js**](https://gist.github.com/dannyshain/abd0816859d11e22facd2dbfb6fb69f3#file-confirmation-js) hosted with ❤ by [**GitHub**](https://github.com/)

Like the alert, we’re testing to make sure our message (text) includes exactly what we want.

The tricky part however, is we don’t have an explicit way to test the response as to whether the person hit “Ok” or “Cancel”. So we have to assert against something in the UI, where in this example, we have an element that contains the message.

Speaking of having two options, by default, Cypress will automatically hit “Ok” in our confirmation. If we wanted to instead test a use case of someone clicking “Cancel”, we can instead return false in our event callback:

|  |  |
| --- | --- |
|  | cy.on('window:confirm', (text) => { |
|  | expect(text).to.contains('Would you like to confirm?'); |
|  | return false; |
|  | }); |

[**view raw**](https://gist.github.com/dannyshain/e3d2eb7d94f3652e679f4b0df77a6799/raw/91013e81345b73a96107cbf01a41b6f4b59b86c7/confirmationCancel.js)[**confirmationCancel.js**](https://gist.github.com/dannyshain/e3d2eb7d94f3652e679f4b0df77a6799#file-confirmationcancel-js) hosted with ❤ by [**GitHub**](https://github.com/)

And similarly, if you want to test that someone cancelled, you’ll need to have a way to indicate that in the UI.

**Testing prompts in the browser with Cypress**

Prompts are where things get slightly different. Let’s dive right in:

|  |  |
| --- | --- |
|  | it('should trigger a prompt with a message', () => { |
|  | cy.window().then(win => { |
|  | cy.stub(win, 'prompt').returns('This is my answer.'); |
|  | cy.get('#prompt-button').click(); |
|  | cy.get('#prompt-answer').contains('Answer: This is my answer.'); |
|  | }); |
|  | }); |

[**view raw**](https://gist.github.com/dannyshain/1f8ece8a2bcfd6a997a00c1dc2011a33/raw/e429283181ba7fdaf2052ebe34c858c44662d3e8/prompts.js)[**prompts.js**](https://gist.github.com/dannyshain/1f8ece8a2bcfd6a997a00c1dc2011a33#file-prompts-js) hosted with ❤ by [**GitHub**](https://github.com/)

You’ll notice that we’re not testing this using a Cypress event like we did before. That’s because Cypress doesn’t include a window:prompt event similar to our alerts and confirmations.

Instead, we’re going to first access the window using cy.window(), where we’ll then “stub” that prompt event, along with the message that we want Cypress to return to the prompt dialogue.

Once stubbed, just like the other two, we can click our button and make sure that our UI includes an indication that the answer worked.

Now like our confirmation, the prompt dialogue has the ability for someone to click “Cancel”.

|  |  |
| --- | --- |
|  | cy.window().then(win => { |
|  | cy.stub(win, 'prompt').callsFake(() => null); |

[**view raw**](https://gist.github.com/dannyshain/7a31afd7ba56e1e25f60bb4eedadc5b7/raw/a92bef5ad4ebfa82f5ffc119566a431ccd9fb3cc/promptCancel.js)[**promptCancel.js**](https://gist.github.com/dannyshain/7a31afd7ba56e1e25f60bb4eedadc5b7#file-promptcancel-js) hosted with ❤ by [**GitHub**](https://github.com/)

To test out the cancel path, we can use the Cypress callsFake method and return null (it has to be null), which will then tell Cypress that we want to cancel that prompt when it gets triggered.

**What to expect in Cypress**

After implementing these tests, you might be wondering why you’re not actually seeing these dialogues inside of Cypress itself.

Like we mentioned earlier, Cypress will confirm each dialogue by default, meaning if there’s an “Ok” option, it will essentially click that immediately. As Cypress works through these tests, you won’t see that dialogue in the Cypress UI.

**Learn more about testing in Cypress and other tools!**

To get a step-by-step walkthrough for testing alerts, confirmations, prompts, and a whole lot more in Cypress and even Selenium Java, check out the [Automation Cookbook](https://applitools.com/cookbook/)!