

Cypress Core Concepts



ADHITHI RAVICHANDRAN

SOFTWARE CONSULTANT, AUTHOR, SPEAKER

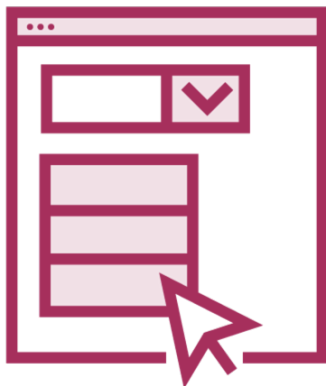
@AdhithiRavi www.adhithiravichandran.com



Cypress Core Concepts



Organization



Interacting with
Elements



Retry-ability



Aliases



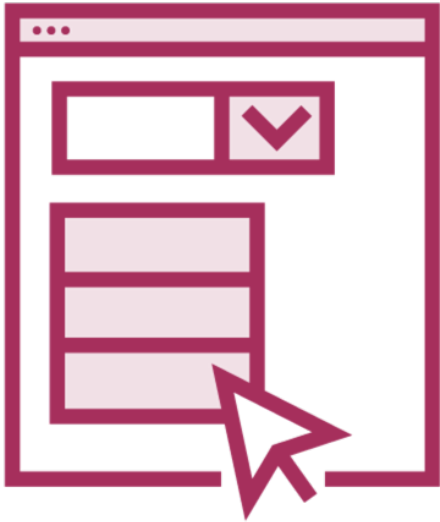
Organizing Tests



Interacting with Elements



Commands



Cypress comes with a set of commands to interact with the web page

They are categorized as parent, child, and dual commands

Cypress provides API for creating custom commands and overwriting existing commands



Parent Commands

Parent commands begin a new chain of Cypress commands

```
cy.visit('http://localhost:4100')
```

```
cy.get('form')
```

```
cy.request('http://dev.local/seed')
```

```
cy.exec('npm run build')
```

```
cy.route('/users/**')
```



Child Commands

Chained off a parent command, or another child command

```
cy.get(' [data-cy=username] ').click()
```

```
cy.get(' [data-cy=username] ').type(username)
```

```
cy.get('.article').find('footer')
```

```
cy.contains('Login').should('be.visible')
```



Dual Commands

Can either start a chain or be chained off an existing one

```
cy.contains()
```

```
cy.screenshot()
```

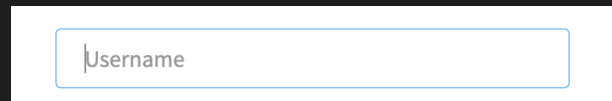
```
cy.scrollTo()
```

```
cy.wait()
```



Selecting Elements

```
<input  
  className="form-control form-control-  
lg"  
  name="username"  
  type="email"  
  data-cy="username"  
  placeholder="Email"  
  value={email}  
  onChange={this.changeEmail} />
```



Selecting Elements

Component to test

```
<input  
  className="form-control form-control-lg"  
  name="username"  
  type="email"  
  data-cy="username"  
  placeholder="Email"  
  value={email}  
  onChange={this.changeEmail} />
```

Cypress test

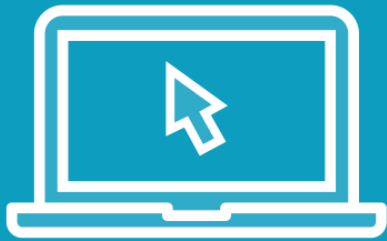
```
cy.get('form-control- form-control-lg').click()  
  
cy.get('input').click()  
  
cy.get('[name=username]').click()  
  
cy.contains('Username').click()  
  
cy.get('[data-cy=username]').click()
```



Test Register and Login Workflows



Demo



Write Cypress tests for Conduit App

- Test the **Register** and **Login** user workflow
- Test interacting with elements
- Tests will demonstrate:
 - Use of selectors
 - Use of commands
 - Chaining commands



Assertions



Common Cypress Assertions

```
cy.contains('[data-cy=profile]', username)  
  .should('be.visible')
```

```
cy.location('pathname')  
  .should('equal', '/register')
```

```
cy.get('.article-preview')  
  .should('have.length', 10)
```

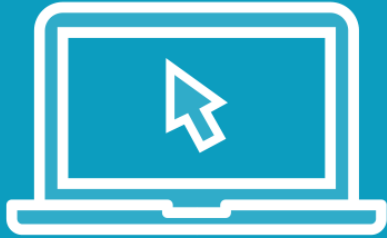
```
cy.get('[data-cy=profile]').should('not.exist')
```



Assertion Demo and More Tests



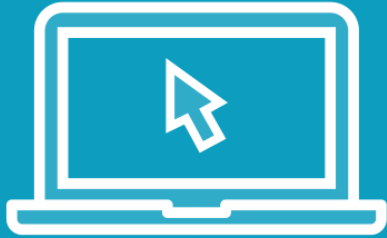
Demo



Update Register and Login tests with assertions



Demo



Add a test for writing a new post on Conduit

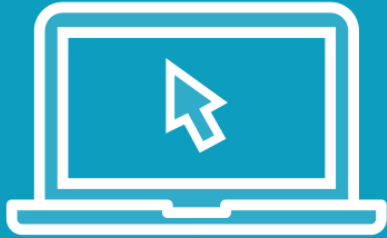
- Use commands, selectors, and assertions for complete testing



Aliases



Demo



Refactor existing tests to make use of Aliases



Cypress Retry-ability



Cypress Retry-ability



Efficiently test dynamic applications

Applications are asynchronous

- Smart commands wait for application to update
- If assertion following a DOM query command fails – Keep retrying until timeout
- No hardcoding of waits in code

Retry-ability Caveats



Cypress only retries commands that query the DOM

- *.get(), .find(), .contains(), etc.*

Commands that may change the state of application are not retried

- *.click() - Not retried by Cypress*

Only last command before assertion is retried



Demo



Write a test to favorite an article

- Demonstrate Cypress retry-ability
- More tests for the Conduit app



Summary



Cypress Core Concepts Summary

- Test Structure
- Interacting with elements
- Commands
- Assertions
- Selectors
- Retry-ability
- Aliases



Up Next:

Cypress Ecosystem and Tooling

