

CYPRESS – END TO END TEST AUTOMATION

- BY COPE AUTOMATION

WHY IS CYPRESS ?

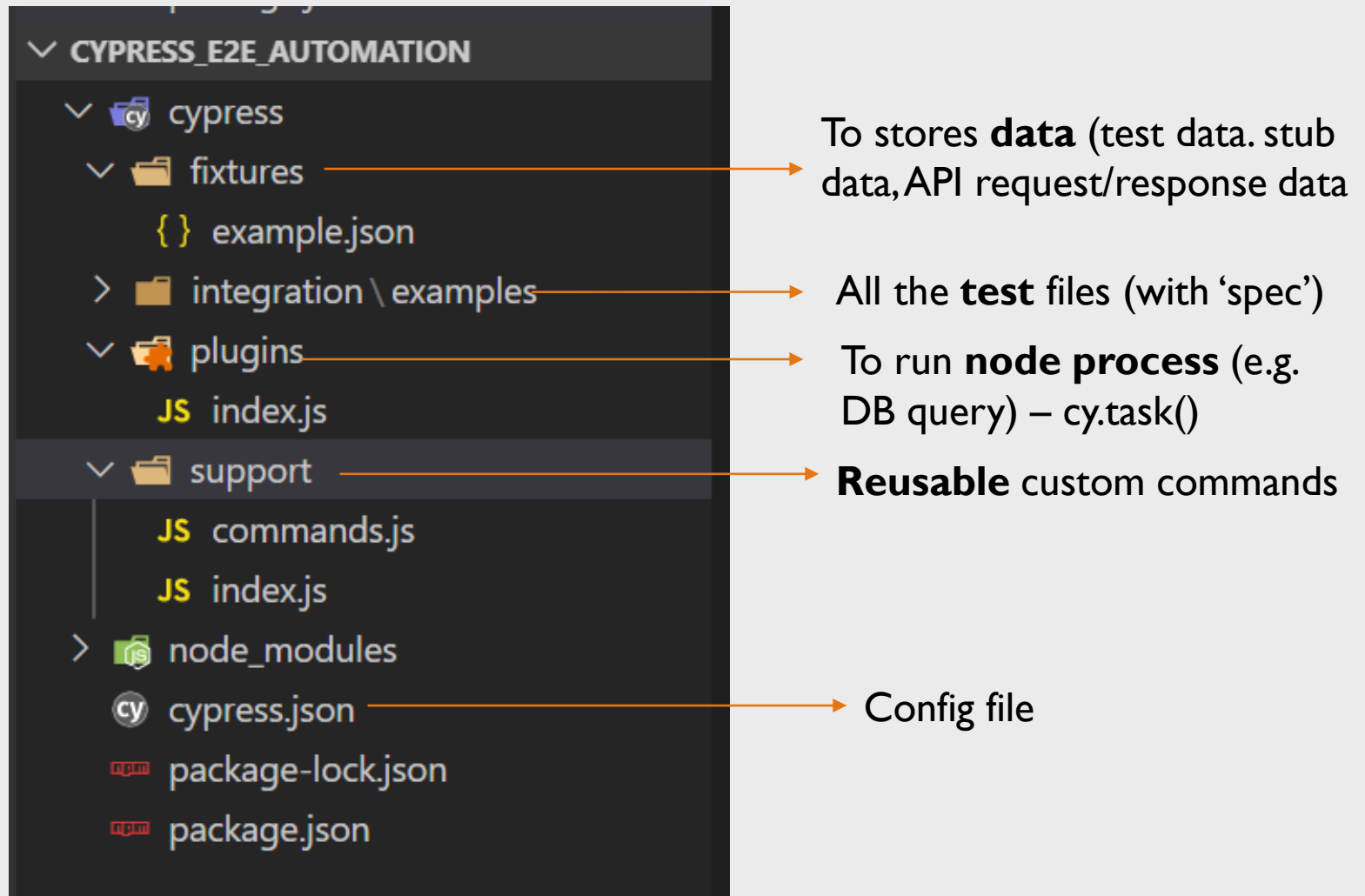
1. Time travel ability
2. Auto waiting & Retry-ability
3. Fast, less flaky and supports E2E tests
4. Complete control of the app or Browser: Network Traffic Control, API tests
5. Debuggability
6. Can be used by DEV and QA
7. Makes it possible to create **state independent, consistent tests**
8. Assertions at it's best: DOM & App level
9. Built-in screenshots and video recording capability
10. Ideal for JS based model web frameworks: Angular, React

INSTALLATION COMPONENTS

- Code Editor: VS code (Microsoft)
- Node.js – Node Js and NPM
- Cypress



FOLDER STRUCTURE



CYPRESS COMPONENTS – TEST RUNNER

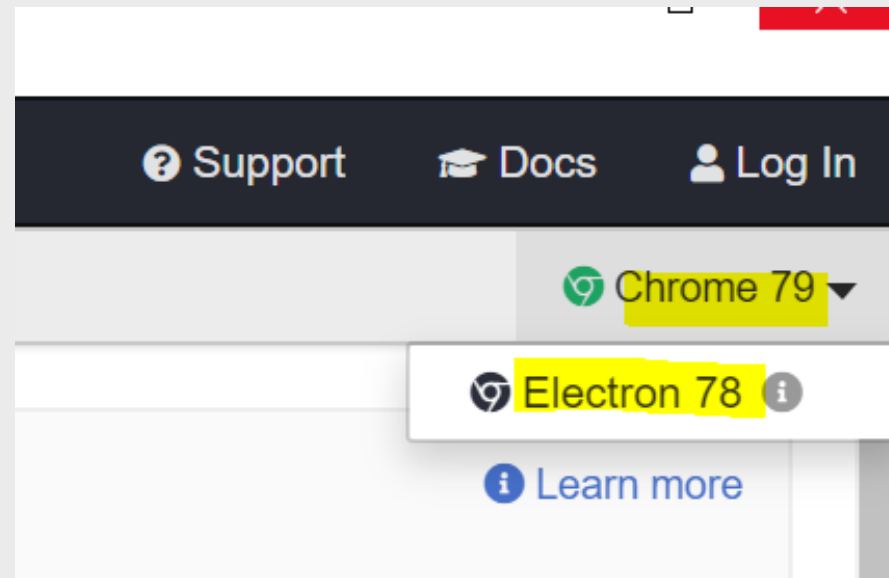
The screenshot shows the Cypress Test Runner interface with several key components highlighted by orange arrows and labels:

- Test Status Menu**: Located at the top left, it displays test results: 4 passed (green checkmarks), 2 failed (red Xs), 0 pending (circles), and a total time of 3.94 seconds.
- Url Preview**: Located at the top center, it shows the current URL being tested: `http://localhost:8888/#/`.
- Viewport Sizing**: Located at the top right, it shows the current viewport dimensions: 625 x 750 (97%).
- Command Log**: Located on the left side, it lists the commands executed during the test. The current test is "New Todo" with a "TEST" section showing commands like `GET`, `-TYPE`, `-FIRST`, `-FIND`, and `-ASSERT`. The last command, `-ASSERT`, is highlighted in green, indicating it passed.
- App preview**: Located on the right side, it shows a real-time preview of the application being tested. The app is a "todos" application with a list of items: "buy some cheese" and "feed the cat". The "buy some cheese" item is checked off, and the "feed the cat" item is also checked off.

Annotations with orange arrows point from the labels to the corresponding components in the interface.

BROWSER SUPPORT

I. What's the difference?



CODE INTELLIGENCE

1. Cypress

- **Option 1:** `/// <reference types="Cypress" />` (Triple-Slash Directives)
 - Downside: We need to add to each spec/test file
- **Option 2:** Reference type declarations

2. Cypress.json

WHAT IS MOCHA?



Mocha is a **JavaScript test framework** running on Node.js and in the browser

MOCHA'S BDD INTERFACE

```
describe('Array', function() {  
  before(function() {  
    // ...  
  });  
  
  describe('#indexOf()', function() {  
    context('when not present', function() {  
      it('should not throw an error', function() {  
        (function() {  
          [1, 2, 3].indexOf(4);  
        }).should.not.throw();  
      });  
      it('should return -1', function() {  
        [1, 2, 3].indexOf(4).should.equal(-1);  
      });  
    });  
    context('when present', function() {  
      it('should return the index where the element first appears in the array', function() {  
        [1, 2, 3].indexOf(3).should.equal(2);  
      });  
    });  
  });  
});
```

MOCHA'S BDD – SUMMARY

Suite
Level

before(), after()

describe() or context()

Test
Level

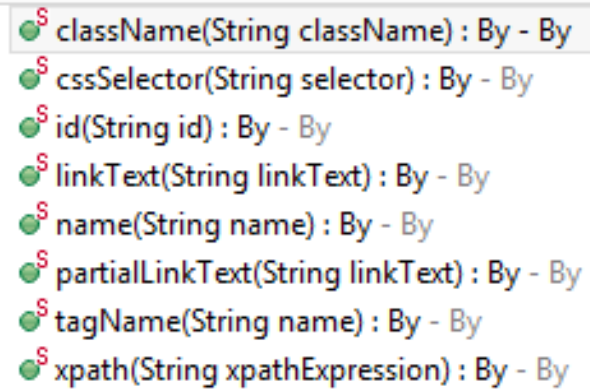
beforeEach(), afterEach()

it(), specify()

WHY UNDERSTANDING SELECTORS IS IMPORTANT ?

“The more you know about selectors, the less time you spend automating tasks”

DOES CYPRESS FOLLOW SELENIUM SELECTORS METHOD?

A screenshot of a Cypress documentation page showing a list of selectors. Each selector is preceded by a green circle with a red 'S' icon. The selectors listed are: className, cssSelector, id, linkText, name, partialLinkText, tagName, and xpath. Each selector is followed by its return type, which is 'By - By'.

```
className(String className) : By - By
cssSelector(String selector) : By - By
id(String id) : By - By
linkText(String linkText) : By - By
name(String name) : By - By
partialLinkText(String linkText) : By - By
tagName(String name) : By - By
xpath(String xpathExpression) : By - By
```

cy.
 .get()
 .contains()

WHAT IS JQUERY?

*It's a **JavaScript library** - It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers*

WHAT IS CSS

Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents.

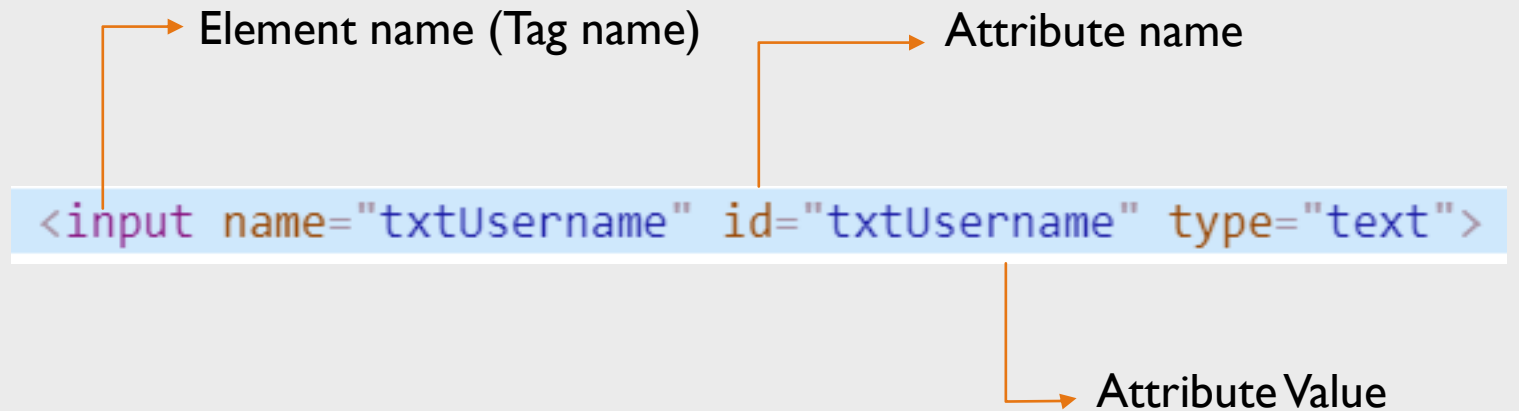
WHAT'S THE RELATION BETWEEN JQUERY AND CSS?

```
<script>  
$( "#myDiv" ).css( "border", "3px solid red" );  
</script>
```

Jquery queries DOM element to **apply CSS**
(styles)

Cypress uses the same mechanism to **Action**
on the element

WHAT IS AN ELEMENT (TAG) & AN ATTRIBUTE ?



MOST USED PATTERNS

Pattern 1: ID

Syntax	#id
Description	<i>Selects element with the given id attribute</i>
Example	<pre>cy.get('#user-name').type('standard_user')</pre>

MOST USED PATTERNS

Pattern 2: Class

Syntax	.class
Description	<i>Selects element(s) with the given class</i>
Example	<code>cy.get('.btn_action').click()</code>

MOST USED PATTERNS

Pattern 3: Combination of Element name (Tag name) and Attribute

Syntax	tagname[attName = value]
Description	<i>Selects element(s) which matches given combination</i>
Example	<code>cy.get('div[type=username]').type('standard_user')</code>

MOST USED PATTERNS

Pattern 4: Attribute Equals

Syntax	[attName = value] or [attName = “value”]
Description	<i>Selects element which matches attribute name and value</i>
Example	<pre>cy.get('[class=inventory_list]').click()</pre>

MOST USED PATTERNS

Pattern 5: Multiple Attributes

Syntax	[attName1 = value1] [attName2 = value2]
Description	<i>Selects element(s) which matches this combination *</i>
Example	<code>cy.get(' [name=txtPassword][type=password]')</code>

*Optionally, the element name/tagname can be added

OTHER PATTERNS

Pattern	Syntax	Desc
6	Tagname(id/class) e.g div.p or div#username	Combination of tag and id/class
7	parent>child	Selects the direct child
8	[attName*=value]	Attributes contains given substring
9	[attName^=value]	Attributes starts with given string
10	[attName\$=value]	Attributes ends with given string
11	:eq(index)	Selects the specific index (starts from 0)
12	tagname	Selects the element(s) with given tagname
13	(selector 1, selector 2, selector n)	Multiple valid selectors

* Note: cy.contains() covers text based search

CY COMMANDS TO FIND AN ELEMENT/ELEMENTS

■ cy.

1. `get()`
2. `contains()`
3. `root()`

On existing DOM Element

1. `contains()`
2. `find()`
3. `filter()`
4. `not()`
5. `children()`
6. `first()` , `last()`
7. `next()`, `nextUntil()`
8. `parent()`, `parents()`, `parentsUntil()`
9. `Prev()`, `prevAll()`, `prevUntil()`
10. `siblings()`
11. `window()`
12. `within()`

REFERENCE LINKS

- jQuery: <https://api.jquery.com/category/selectors/>