Cypress notes Day 1:

Features

Cypress comes fully baked, batteries included. Here is a list of things it can do that no other testing framework can:

- **Time Travel:** Cypress takes snapshots as your tests run. Hover over commands in the Command Log to see exactly what happened at each step.
- **Debuggability:** Stop guessing why your tests are failing. Debug directly from familiar tools like Developer Tools. Our readable errors and stack traces make debugging lightning fast.
- **Automatic Waiting:** Never add waits or sleeps to your tests. Cypress automatically waits for commands and assertions before moving on. No more async hell.
- **Spies, Stubs, and Clocks:** Verify and control the behavior of functions, server responses, or timers. The same functionality you love from unit testing is right at your fingertips.
- **Network Traffic Control:** Easily control, stub, and test edge cases without involving your server. You can stub network traffic however you like.
- **Consistent Results:** Our architecture doesn't use Selenium or WebDriver. Say hello to fast, consistent and reliable tests that are flake-free.
- **Screenshots and Videos:** View screenshots taken automatically on failure, or videos of your entire test suite when run from the CLI. Record to Cypress Cloud to store them with your test results for zero-configuration debugging.
- **Cross browser Testing:** Run tests within Firefox and Chrome-family browsers (including Edge and Electron) locally and optimally in a Continuous Integration pipeline.
- **Smart Orchestration:** Once you're set up to record to Cypress Cloud, easily parallelize your test suite, rerun failed specs first with Spec Prioritization, and cancel test runs on failures with Auto Cancellation for tight feedback loops.
- **Flake Detection:** Discover and diagnose unreliable tests with Cypress Cloud's Flaky test management.

Assertions:

Cypress notes Day 1:

Implicit : in-built assertion

Should(),or, and() commands will be use in implicit.

Should-contain:

Ex: cy.get('locater').should('contain','Button')

Should-have:

EX: should('have. Attribute name',' Attribute value')

Should-be:

.should('be.visible')

Be.selected

Be.disabled

Be.focused.

Explicit: expect() ,or, assert() will be use in explicit.

EX: expect(true).to.be.true

Let name='cypress';

Expect(name).to.be.equal('cypress')

To.not.equal()

To.be.a('string')

To.be.true

To.be.false

To.be.null

To.exist

Cypress notes Day 1:

Assert.equal.

Assert.equl(3,3,'message')

Page Object Modle:

Ex:

Export class loginpage{

enterUsername(){

cy.get("textusername).type('admin')

}

}

Login test

Import{loginPage} from "./pages/login_page"

Const loginpage=new loginpage()

If('pom demo',function(){

Cy.visit('url')

Loginpage.enterusername()

})

Locaters:

Cypress notes Day 1:

Type()---------- this method is use for type data in text box.

Cy.get("locater") ------this method used to identify the elements.

It use css selector and Xpath locter only

CSS selector:

Tag id----------(#idvalue)

Tag class-------(.classvalue)

Tag attribute----------[attribute='value']

Tag class attribute-------- (.classvalue[attribute='value'])

Writing the test script in cypress

Syntax:

```
describe('test suite',function(){
it('test case1',function(){
steps
})
It('test case 2',function(){
Steps
})
})
```

Cypress notes Day 1:

Interacting with UI Elements

Commands.

.visit() ---is user for navigate url

.url()----- is use for get current url.

.get ----- is use for select element.

.title() -------is use for get titile of the page.

CHECK BOXES.

FOR (CHECKED).

cy.get('selector').check().should('be.checked').and('have.value','cricket')

FOR UNCHECKED:

cy.get('selecter').uncheck().should('not.be.checked')

CHECK MULTIFULL:

Cy.get('selecters').check(['value','value2'])

Dropdown:

If select drop down :

.select('value').should('have.value','value')

Cypress notes Day 1:

NOT SELECT DROPDWM:

Cy.get('selector').click()

Cy.get('selector(coman selector)').contains('English').click()

Cy.get('selector(coman selector)').contains('japan').click()

Alerts:

cy.on('window:alert',(str) =>

{

Expect(str).to.equal('alert message')

}

)

Conformation alert:

cy.on('window:confirm',(str) =>

{

Expect(str).to.equal('press a button')

}

)

PROMT ALERT

---

cy.window().then((win)=>{

cy.stub(win,'prompt').returns('welcom');

})----- After click on prompt alert link.

NAVIGATE (BACK,FORWARD,REFRESH):

cy.go('forward') or cy.go(1)

cy.go('back') or cy.go(-1)

cy.reload()

CYPRESS HOOKS:

- beforeEach ------ this will execute before one test
- afterEach--------- this will execute after each test
- before---------- this will execute before all test.
- after---------- this will execute after all test.

```
EX:
describe('my test suite',() =>{
before(function(){
cy.log('----------before-----')
})
after(function(){
cy.log(--------- after------)
})
beforeEach(function(){
cy.log(---------before each------)
})
afterEach(function(){
cy.log(-------afterEach--------)-------for print in console.
})
It('search',() =>{
Code
})
```

```
It("search2", function(){
Code
})
})
```

Fixtures file

Create fixture file in fixtures folder with extension .json

```
{
"email":"1234",
"password":"123456"

}
```

Using of fixture fille in cypress.

Syntax:

```
before(function(){
cy.fixture('file path').then(function(data){
this.data=data -------- store the data in this.data(it is
global veriable).
})
})
```

Use email inside type(this.data.email)

Handling window popup in cypress.

Remove target attribute

cy.get('selecter').invoke('removeAttr','target').click();

Handling Iframes:

```
const iframe=cy.get('framelocater')
.its('0.contentDocument.body')
.should('be.visible')
.then(cy.wrap);
```

Iframe.find('locater').click() ------this for click element inside frame.

Iframe.clear().type('welcom');

MOUSE OVER ACTIONS.

Cy.get('selector').trigger('mouseover').click(); ---- this for move to element.

.trigger('contextmenu') ----- this for right click.
(or)
.rightclick();

.trigger('dblclick'); ------ for double click.
(or)
.dblclick();

Scroll particular element in page

```
.scrollIntoView();
.scrollIntoView({duration:200});
```

Tags:

.skip---- it is skip test case.

.only -------- it execute only one test case.

CUSTOM COMAND.

Creating:

```
Cypress.Comands.add('methodname',(variable) =>{

Cy.get('selector').contains(variable).click();
})
```

Overriding existingcomands

```
Cypress.Commands.overwrite('methodname',(variable,
```

SCREENSHOTS:

```
cy.screenshot("homepage");
```

take screenshot for particular element.

```
Cy.get("selector").screenshot("name");
```

Cypress notes Day 1:

Run exact file or test in comandfront.

Npx cypress run –spec filepath