

b.ignited

- Test automation
- Consultancy
- Experts



ⓘ Note

We're nothing short of wizards

Test Automation

■ What?

- Execution of test cases, using scripts or low-coded workflows
- Software to test other software
- Automated checks on the application
- Building the user flow in the application under test

Test Automation

Why?

- Regression testing is expensive
- Humans make mistakes

⚠ Warning

Software never does ???

- Speedy thing is speedy
- Testing as a shared objective

Test Automation

Why?

- Almost a requirement in an Agile methodology
- Regression testing whenever (?)
- Continuous integration and continuous deployment

Test Automation

How?

- Write code
- Use framework
- Drag&Drop with low-code tooling

In essence:

- Test using technical solutions

Advantages & Disadvantages

Advantages

- Cheap
 - Low execution cost

⚠ Warning

High investment cost

- Reliable
- Automate all the things
 - Except the hard thing
- High consistency
 - Besides the flaky tests

Advantages & Disadvantages

Disadvantages

- Easy to get lost in code
- Complexity
- Slow to create tests
 - Decent code takes time
- Higher requirements for testers
- Testing as a shared objective

Advantages & Disadvantages

■ Mitigating the Disadvantages

■ Easy to get lost in code

- Consider the over-engineering is a risk
- The focus should remain on the tests

Advantages & Disadvantages

■ Mitigating the Disadvantages

■ Complexity

- Coding isn't easy
- `git gud`
 - Continuous Learning

Advantages & Disadvantages

■ Mitigating the Disadvantages

■ Slow to create tests

- Initial validation could be manual testing

■ ⚠ Caution

■ Automation could be forgotten

- Slow is smooth, smooth is fast

Advantages & Disadvantages

■ Mitigating the Disadvantages

■ Higher requirements for testers

- Continuous learning
- On the job training

■ ⚠ Caution

■ More demanding job

Implementation of Cypress

Cypress

- Modern UI testing framework
- Built for developers
- Design for UI testing
- e2e testing
- Component testing

Implementation of Cypress

Cypress

What is the part we care about?

- Not Selenium
- Free
- Easy to use
- Incredibly fast
- Not Selenium

=====

Implementation of Cypress

Cypress

- Node framework
- Highly customizable
- Many many supporting plugins

⚠ Caution

The documentation has been written by developers,
that happen to write tests.

=====

Implementation of Cypress

Cypress

```
describe('Homepage validation checks' () => {  
  beforeEach(() => {  
    cy.visit('/home');  
  });  
  
  it('Should see the banner on the homepage', () => {  
    cy.get(home.field.banner).should('be.visible')  
  });  
});
```

Implementation of Cypress


Basic

Navigating to a website

```
cy.visit('{url}')
```

=====

Implementation of Cypress

 Basic

Visit the application!

Implementation of Cypress

Basic

A better way to navigate to the baseUrl would be

```
export default defineConfig({  
  e2e: {  
    baseUrl: '{baseUrl}'  
  },  
});
```

In combination with

```
cy.visit('/')
```

Access the baseUrl as a global variable

=====

Implementation of Cypress

Basic

Edit the Cypress Config!

Implementation of Cypress

■ Debugging options

- Wait for a full second

```
cy.wait(1000)
```

- Pause the test, can be resumed

```
cy.pause()
```

Implementation of Cypress

Debugging options

- Good old fashion debuggin

```
cy.debug()
```

- Debug the return of the get command

```
cy.get({element}).debug()
```

⚠ Warning

Developer tools have to be open

=====

Implementation of Cypress

■ Debugging options

- Can't go wrong with the classics

```
cy.log('Doing stuff')
```

- Requests and responses
- Can be used for assertions

```
cy.intercept('/users/**').as('users')  
// Do stuff  
cy.wait('@users')
```


Implementation of Cypress

Elements & selectors

Almost anything on the web can be considered an element. And each element can be interacted with thanks to a selector.

There are several type of selectors

- Good ones
- Bad ones
- Terrible ones

=====

Implementation of Cypress

█ Elements & selectors

█ Xpath

- Most powerful
- Most versatile
- Only as a last resort

█ ⚠ Warning

█ Xpath is incredibly untrustworthy

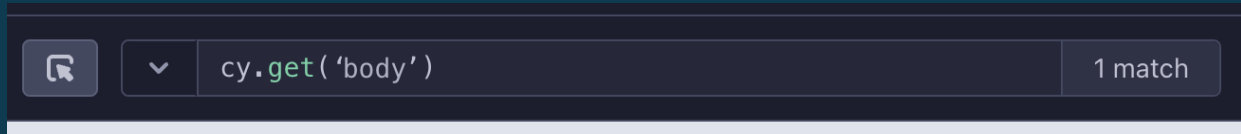
=====

Implementation of Cypress

Elements & selectors

CSS

- Powerful
- Most common
- Native to Cypress



Note

Requires some knowledge-ability to create sturdy selectors

Implementation of Cypress

█ Elements & selectors

█ Data-cy

- Unlikely to encounter in the wild
- Requires frontends to facilitate
- Best possible option

ⓘ Note

Always force the frontenders, life should be easy

=====

Implementation of Cypress

■ Elements & selectors

■ Dynamic selectors

- Selectors are just strings until used.

```
export const inbox = {  
  findMessageByTopic: (topic: string) => {  
    return `#box-message-topic${topic}`;  
  }  
}
```

```
cy.get(inbox.findMessageByTopic('Unremarkable topic about a boring subject')).click()
```

Implementation of Cypress

█ Elements & selectors

▒ More complicated selectors

- '*' => Wildcard
- '^' => Starting with
- '\$' => Ending with

```
export const inbox = {  
  messageDelete: '[id*="delete"]',  
  messageForward: '[id^="forwardTo"]',  
  folder: '[id$="-folder-name"]'  
}
```

Implementation of Cypress

Buttons

Time to start clicking

```
cy.get({element}).click()
```



Implementation of Cypress

Dropdowns

We should select something

```
cy.get({dropdownElement}).select({value})
```

Difficult to imagine something more simple

=====

Implementation of Cypress

Input

Maybe we want to type something

```
cy.get({input}).type({whatever})
```

Note

SendKeys is for the ancient ones

=====

Implementation of Cypress

Into Practise

1. Navigate to the create form
2. Fill in the form
3. Submit the form

Implementation of Cypress

Assertions

Should we assert ?

```
cy.get({element}).should({SomeChaiAssert})
```

💡 Tip

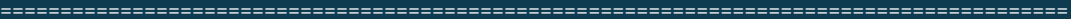
Any chai assert is possible

=====

Implementation of Cypress

Assertions

1. Assert the submit
2. Assert all the values



Implementation of Cypress

The Page-Object Model

Define a page

```
export const home = {  
  searchBar: '[data-cy=".vl-searchbar"]',  
  searchButton: '[data-cy=".vl-search-submit"]',  
}
```

Implementation of Cypress

The Page-Object Model

Define a page

```
export const home = {  
  button: {  
    search: '[data-cy=".vl-search-submit"]',  
  },  
  input: {  
    search: '[data-cy=".vl-searchbar"]',  
  }  
}  
  
cy.get(home.button.search).click()
```

=====

Implementation of Cypress

The Component-Object

Make the definition as small as possible

```
export const header = {  
  field: {  
    navigateHome: '[data-cy=".vl-home"]',  
    profile: '[data-cy=".vl-profile"]',  
  }  
}
```

Tip

Decent frontend-developers define components

Tip

Think reusability

Implementation of Cypress

The Perfect marriage

A page does contain components, usually

```
import { header } from './componentObjects/header'
import { footer } from './componentObjects/footer'

export const home = {
  button: {
    search: '[data-cy=".vl-search-submit"]',
  },
  input: {
    search: '[data-cy=".vl-searchbar"]',
  },
  header,
  footer
}
```

```
cy.get(home.header.profile).click()
```

=====

Implementation of Cypress

Too much of a good thing ?

```
import { header } from './componentObjects/header'
import { footer } from './componentObjects/footer'
import { search } from './componentObjects/search'

export const home = {
  search,
  header,
  footer
}
```

```
cy.get(home.search.input).type('Searching for something')
cy.get(home.search.button).click()
```

=====

Implementation of Cypress

Page-Objects

1. Create the Component objects
2. Create the Page objects

Implementation of Cypress

Other selector options

Sometimes we need to get creative

```
.children()  
.parent()  
.within()
```

Implementation of Cypress

■ Action-Page Model

A component is more than it's selectors

```
const search = {
  searchBar: '[data-cy=".vl-searchbar"]',
  searchButton: '[data-cy=".vl-search-submit"]',
}

export function search(param: string) {
  cy.get(searchBar).type(param)
  cy.get(searchButton).click()
}
```

⚠ Warning

Over-engineering is a clear and every present danger

=====

Implementation of Cypress

Cypress commands

The DRY rule

```
Cypress.Commands.add('login', (username: string, password: string) => {  
  cy.get(login.input.username).type(username)  
  cy.get(login.input.password).type(password)  
  cy.get(login.button.login).click()  
})
```

=====

Implementation of Cypress

Cypress commands

1. Create a custom login command
2. Create a custom create course command

Implementation of Cypress

■ Mocking API responses

Pesky unstable backend ruining my tests

- Mocking improves stability
- Easier setup
- Faster tests
- Independence from the backend

⚠ Caution

It will no longer qualify as an e2e test

⚠ Caution

Don't forget to test the backend

Implementation of Cypress

Mocking API responses

Cypress can do this right out of the gate

```
cy.intercept('GET', '/users/*', {
  statusCode: 200,
  body: {
    firstname: 'Robert',
    lastname: 'Martin',
    books: [
      'Clean Code',
      'Clean Architecture',
      'The Clean Coder'
    ]
  }
})
```


Implementation of Cypress

Mocking API responses

```
cy.intercept('GET', '/users/*', {
  statusCode: 200,
  body: {
    firstname: 'Robert',
    lastname: 'Martin',
    books: [
      'Clean Code',
      'Clean Architecture',
      'The Clean Coder'
    ]
  }
}).as('getUsers')

cy.get(userOverview.button.filter).click()
cy.wait('@getUsers')
```

Implementation of Cypress

■ Mocking API responses

And now we can even assert any call

```
cy.intercept('GET', '/users/*').as('getUsers')

cy.get(userOverview.button.filter).click()
cy.wait('@getUsers').then((interception) => {
  expect(interception.request.body.filter.status).to.eq('CREATED')
})
```

=====

Implementation of Cypress

Mocking API responses

1. Mock the get courses API to show multiple courses on the homePage

Implementation of Cypress

Mocking API responses

Mocking can create an end2end story.

Just in different tests

1. Validate the actual requestBody
2. Validate the handling of the mocked response

💡 Tip

No Backend involved, and yet all is tested

**ANY
QUESTIONS?**

