

ŞUBAT 20, 2023

Cypress 101

Test. Automate. Accelerate.

M.FURKAN PORTAKAL

FRONTEND DEVELOPER

MÜF RE DAT

Test Dünyasına Giriş

Cypress- Mocha -Chai

DOM – Web UI Testleri

Workshop

NEDEN TEST YAZARIZ

- HATAYI ÖNLEMEK
- PROBLEMLERİ DOGRU TANIMLAMAK
- DÖKÜMANTASYONU ARTTIRMAK
- KALİTELİ KODU SAKLAMAK

4 Farkli Test Tipi

- UNIT TESTING
- INTEGRATION TESTING
- END TO END TESTING
- STATIC TESTING

UNIT TESTING

Bir bütünü oluşturan küçük birimlerin testidir. Bu bir fonksiyon hatta bir fonksiyonun içinde küçük bir "if" bölümü bile olabilir.

```
1  
2  
3 function sum(arr){  
4     var s = 0;  
5     for(var i= 0; i<arr.length  
6         s = s + arr[i];  
7     }  
8     return s;  
9 }  
10  
11 sum([1,2,3,4,5]);  
12 console.log();
```

UNIT TEST

```
const should = require('chai').should();

describe('Calculator Tests', () => {
  it('should return sum of given 2 numbers', () => {

    const summ = sum(2,3);
    const expectedSum = 8;

    summ.should.be.equal(expectedSum);
  })
})

const sum = (a,b) => {
  return a + b;
}
```

INTEGRATION TESTING

Birden fazla bağımsız birimin birbiriyle doğru bir şekilde entegre çalıştığından emin olunan test tipidir. Birbiriyle etkileşim halinde olan classların, database ya da api requestleri ile gelen sonuçlara uygun testi örnek gösterilebilir.

```
app.get('/', function(req, res){  
  // firstly we will define para  
  var options = {  
    host: 'bookwriter.com',  
    port: 80,  
  }  
  
  //we will use http to make our  
  var req = http.request(options)  
  req.end();  
});
```

STATIC TESTING

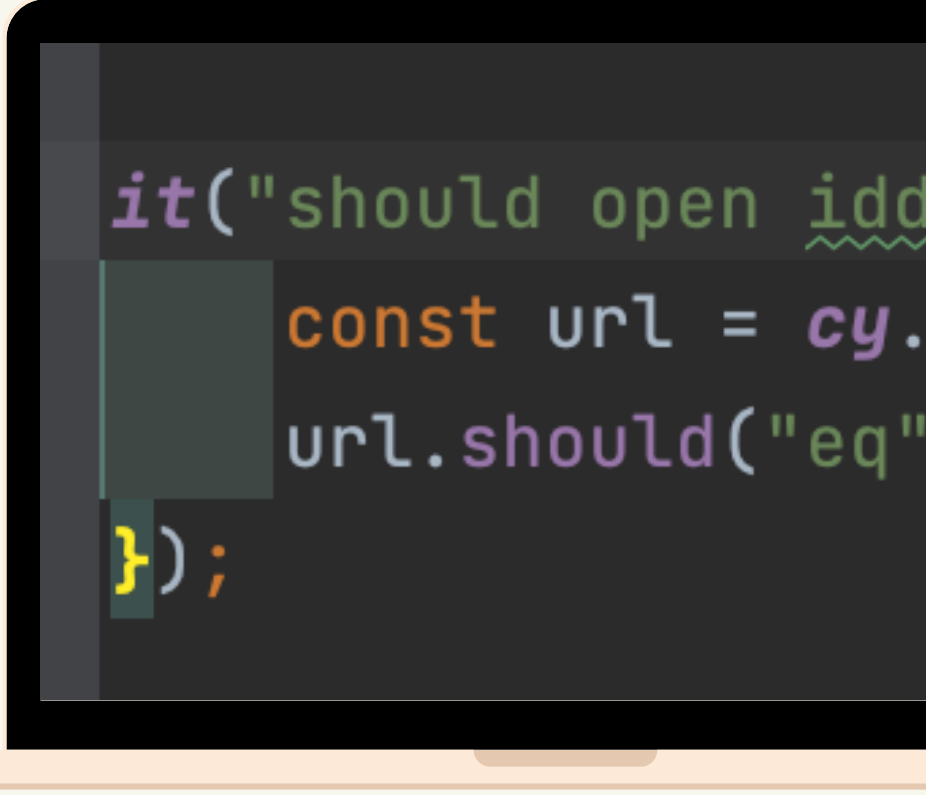
Yazi, resim vs gibi statik birimlerin doğru çalıştığından emin olmamızı sağlayan test tipidir. Genellikle typo hatalarını önlemek için kullanırız.



END TO END TESTING

Bir diğer deyişle fonksiyonel testtir. Son kullanıcının hareketlerini bir robot ya da bir simülasyon yardımı ile taklit edip ürünümüzün doğru çalışıp çalışmadığından emin olmaktır.

Diger popüler e2e test aracı ise Cypress'dir

A laptop screen displaying Cypress test code. The code is written in a dark-themed editor with syntax highlighting. The visible code is:

```
it("should open idd  
  const url = cy.  
  url.should("eq"  
});
```

```
it("should open idd  
  const url = cy.  
  url.should("eq"  
});
```



- End to End test tool'udur.
- Son kullanıcıyı: simüle etmemizi sağlar
- Son kullanıcıya ait hareketleri simüle ederken aynı zamanda video recording ile testin nasıl çalıştığını anlamamızı sağlar
- Automated Waiting yapar, selenium'a göre büyük bir avantaj sağlar
- CI/CD veya kendi dashboard' üzerinden testlerimizin scheduled bir şekilde çalışmasını sağlar

SUPPORTED BROWSERS

- browser chrome
- browser edge
- browser safari

```
→ ~ npx cypress run --spec cypress/e2e/examples/firstTest.spec.js
```



Chrome



Firefox



Edge



Electron



Brave

CYPRESS DASHBOARD

The screenshot shows the Cypress Dashboard interface. The left sidebar contains the following navigation items:

- ACME Org (John Florence)
- acmeorg.io (View all projects)
- Branches
- Analytics
- Run status (circled in blue)
- Run duration
- Flaky tests
- Project settings
- Support
- Documentation

The main area displays the 'Latest runs' section. The top bar shows the URL 'cloud.cypress.io' and a 'Last updated: 6:20:38pm' timestamp. Below the header, there are filter buttons: 'All Time', 'Status', 'develop', 'Committer', 'Tag', and 'Flaky Tests'. The table below lists the latest runs:

Run Title	Status	Committer	Tag	Flaky Tests
[Chore] More links in admin area to tooling (#5066)	✓	Rob Bickel	develop	3 Flaky
remove usage-by-project flag (#5069)	✓	Armando Vasquez	develop	3 Flaky
[CYCLOUD-738] ensure cancellation date is shown correctly (#5053)	✓	Armando Vasquez	develop	0
Fix Smart Orch "cancelling" (#5048)	✗	Armando Vasquez	develop	2 Flaky
fix: fix path for db reload	✓	Tim Griesser	develop	1 Flaky
Merge branch 'master' into develop	✓	Tim Griesser	develop	3 Flaky
feat: Upgrade to node 16 [CLOUD-694] (#4678)	✓	Tim Griesser	develop	1 Flaky
chore: add deploy steps to dashboard readme (#5059)	✓	Rachel	develop	3 Flaky



HOOKS

With its default "BDD"-style interface, Mocha provides the hooks `before()`, `after()`, `beforeEach()`, and `afterEach()`. These should be used to set up preconditions and clean up after your tests.

```
describe('hooks', function() {  
  before(function() {  
    // runs once before the first test in this block  
  })  
  
  after(function() {  
    // runs once after the last test in this block  
  });  
  
  beforeEach(function() {  
    // runs before each test in this block  
  });  
  
  afterEach(function() {  
    // runs after each test in this block  
  });  
  
  // test cases  
});
```



Chai has several interfaces that allow the developer to choose the most comfortable. The chain-capable BDD styles provide an expressive language & readable style, while the TDD assert style provides a more classical feel.

Should

```
chai.should();
```

```
foo.should.be.a('string');  
foo.should.equal('bar');  
foo.should.have.lengthOf(3);  
tea.should.have.property('flavors')  
  .with.lengthOf(3);
```

[Visit Should Guide](#) ➔



Expect

```
var expect = chai.expect;
```

```
expect(foo).to.be.a('string');  
expect(foo).to.equal('bar');  
expect(foo).to.have.lengthOf(3);  
expect(tea).to.have.property('flavors')  
  .with.lengthOf(3);
```

[Visit Expect Guide](#) ➔

Assert

```
var assert = chai.assert;
```

```
assert.typeOf(foo, 'string');  
assert.equal(foo, 'bar');  
assert.lengthOf(foo, 3);  
assert.property(tea, 'flavors');  
assert.lengthOf(tea.flavors, 3);
```

[Visit Assert Guide](#) ➔

AND

Cypress API

```
it("should have title on iddaa home page - v2", () => {  
  const title = cy.title();  
  title  
    .should("not.be.empty")  
    .and("eq", "İddaa - En İyi, En Güvenilir Yasal Canlı Bahis Sitesi İddaa.com ile Canlı Kupon Yapın")  
    .and("have.length.greaterThan", 10);  
});
```

Birden fazla Assertion yapmamızı
sağlayan methoddur.

SCROLL TO

Cypress API

```
it("should have scroll down to 500px on iddaa home page and windows scrollY attribute should be 500px", () => {  
  cy.scrollTo(0, 500);  
  cy.window().its("scrollY").should("eq", 500);  
});
```

Sayfada scroll simülasyonu yapabilmek
için kullanırız

ITS

Cypress API

```
it("should have scroll down to 500px on iddaa home page and windows scrollY attribute should be 500px", () => {  
  cy.scrollTo(0, 500);  
  cy.window().its("scrollY").should("eq", 500);  
});
```

Bir dom objesinin özelliğini okuyabilmek
için kullanınız

CONTAINS

Cypress API

```
it("should have CANLI text in live matches page on navigation", () => {  
  cy.get('[data-comp-name="mainMenu-live-button"] > div')  
    .contains("CANLI")  
});
```

Belirti bir text içeriği var mı yok mu
kontrolü için kullanırız

EQ

Cypress API

```
cy.get('tbody>tr').eq(0) // Yield first 'tr' in 'tbody'  
cy.get('ul>li').eq(4) // Yield fifth 'li' in 'ul'
```

Element Arrayinden belirli bir indexteki
itemi seçmek için kullanırız

EACH

Cypress API

```
cy.get('ul>li').each(($el, index, $list) => {  
  // $el is a wrapped jQuery element  
  if ($el.someMethod() === 'something') {  
    // wrap this element so we can  
    // use cypress commands on it  
    cy.wrap($el).click()  
  } else {  
    // do something else  
  }  
})
```

EQ

Cypress API

```
cy.get('tbody>tr').eq(0) // Yield first 'tr' in 'tbody'  
cy.get('ul>li').eq(4) // Yield fifth 'li' in 'ul'
```

Element Arrayinden belirli bir indexteki
itemi seçmek için kullanırız

CLICK

Cypress API

```
cy.get('[data-comp-name="slip-shortCode-button"]').click();
```

```
cy.get('.btn').click() // Click  
cy.focused().click() // Click on  
cy.contains('Welcome').click() /
```

TYPE

Cypress API

```
cy.get('input').type('Hello, World')
```

CHECK

Cypress API

```
cy.get('[type="checkbox"]').check() // Check checkbox  
cy.get('[type="radio"]').first().check() // Check first radio button
```


GO

Cypress API

```
cy.go('back')
```

VIEWPORT

Cypress API

```
before(() => {  
  cy.viewport(1920, 1080);  
  cy.viewport("iphone-x");  
});
```

BİRLİKTE DENEYELİM

KURULUM

```
npm install cypress --save-dev
```

TEŞEKKÜRLER