

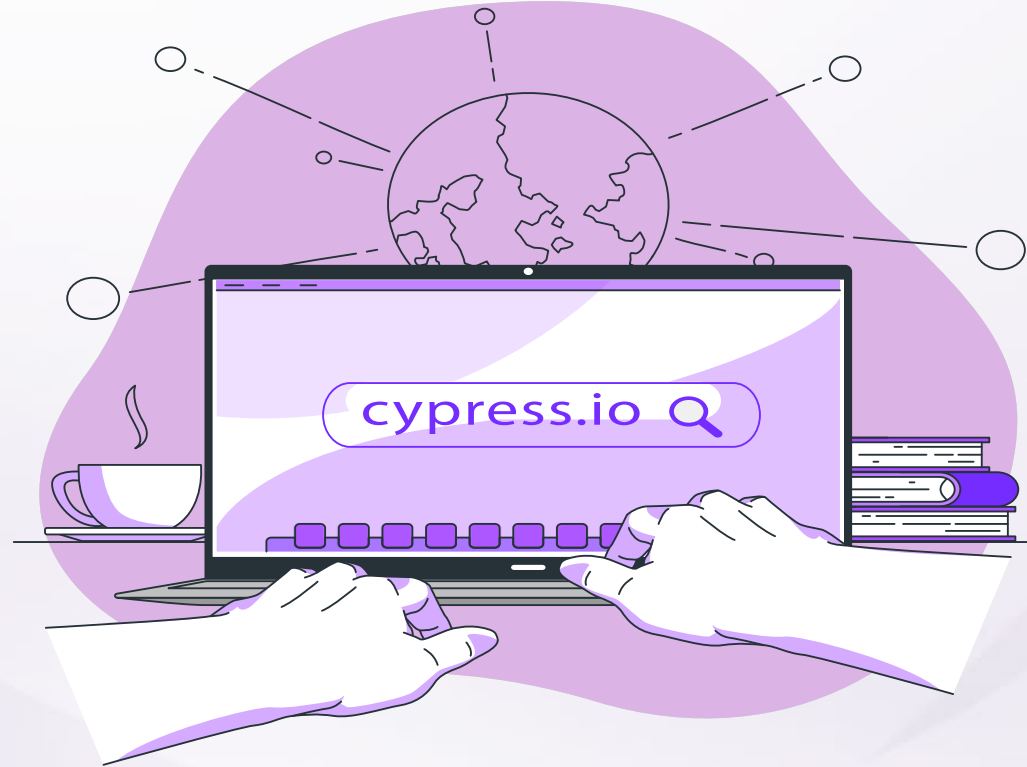
# Cypress Ders Notlari

## Cypress: Güvenilir ve Etkili Bir Test Otomasyon Aracı

Cypress, modern web uygulamalarının test otomasyonu için kullanılan bir JavaScript çerçevesidir. Geliştiricilerin, web uygulamalarını test etmek ve hata ayıklamak için kullanabilecekleri güçlü bir araçtır. Cypress, kullanımı kolay bir arayüze sahip olmasıyla dikkat çeker ve her seviyeden kullanıcının test sürecini daha etkili hale getirebilmesini sağlar.



- ☐ Javascript tabanlı bir test aracıdır.
- ☐ Geliştiriciler ve QA Mühendislerinin görüşleri alınarak dizayn edilmiştir.
- ☐ Doğrudan tarayıcıda çalışır.
- ☐ Mocha ve Chai gibi test araçlarının içindeki bir çok method ve özelliği içinde barındırır.



## Cypress'in Özellikleri



## TIME TRAVEL

Cypress, testleriniz çalışırken anlık görüntüler alır. Her adımda tam olarak ne olduğunu görmek için



## DEBUGGING

Testlerinizin neden başarısız olduğunu tahmin etmeyi bırakın. Okunabilir hata mesajları, hızlı bir şekilde hata ayıklamanıza yardımcı olur.

## REAL TIME RELOAD

Testlerinizde değişiklik yaptığınızda; Cypress, otomatik olarak testlerinizi yeniden yükler. Uygulamalarınızı geliştirirken testleri gerçek zamanlı olarak izleyebilirsiniz.



## Automatic Waiting

Testlerinize asla bekleme eklemeyin. Cypress, devam etmeden önce komutları ve sayfanın yüklenmesini otomatik olarak bekler .



## Spies, Stubs, and Clocks

İşlevlerin, sunucu yanıtlarının veya zamanlayıcıların davranışını doğrulayın ve kontrol edin.



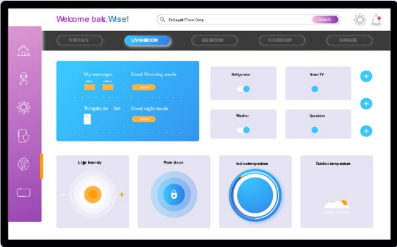
## Network Traffic Control

Sunucunuzu dahil etmeden uç noktaları kolayca kontrol edin ve test edin. Ağ trafiğini istediğiniz gibi yönetebilirsiniz.



## Consistent Results

Mimarimiz Selenium veya WebDriver kullanmaz. Hızlı, tutarlı ve güvenilir testlere merhaba deyin.



## Screenshots and Videos

Başarısızlık durumunda otomatik olarak çekilen ekran görüntülerini veya browsersız bir şekilde çalıştırıldığında tüm test paketinizin videolarını görüntüleyin.



# Cypress Download Statics

cypress

Cypress is a next generation front end testing tool built for the modern web

Enter an npm package...

cypress × + nightwatch + webdriverio + playwright + puppeteer + testcafe

Downloads in past 5 Years ▾





# cypress

Node.Js



<https://nodejs.org/en/>

Visual Studio  
Code



<https://code.visualstudio.com/>

Cypress

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

## ▼ CYPRESS

### ▼ cypress

> downloads

> e2e

> fixtures

> support

> node\_modules

{..} .cypress-cucumber-preprocessorrc.json

📄 .gitignore

JS cypress.config.js

TS cypress.config.ts

📄 package-lock.json

📄 package.json

### downloads:

Bu klasör, testler sırasında indirilen dosyaların depolandığı yerdir. Testlerinizde dosya indirmeyi kontrol ediyorsanız, bu klasörde bu dosyaları bulabilirsiniz.

### e2e:

Bu klasör, Cypress testlerinin yer aldığı klasördür. "e2e" genellikle "end-to-end" testleri anlamına gelir ve bu klasör, uygulamanızın end-to-end testlerinin bulunduğu yerdir.

### fixtures:

Bu klasör, sabit test verilerini saklamak için kullanılır. JSON, XML, CSV, JPG gibi dosya türlerindeki test verilerini burada depolayabilir ve testlerinizde bu verilere erişebilirsiniz.

### support:

Bu klasör, testlerinize yardımcı olan özel işlevlerin ve kodların bulunduğu yeri içerir. Örneğin, özel komutlar veya işlevler genellikle burada tanımlanır ve testlerde kullanılır.

### node\_modules:

Bu klasör, proje bağımlılıklarının (dependencies) ve kullanılan paketlerin depolandığı yerdir. Cypress'ı kurduğunuzda ve başka paketler eklediğinizde, bu paketler genellikle bu klasöre indirilir.

### cypress.config.js:

Bu dosya, Cypress yapılandırma seçeneklerini içeren dosyadır. Örneğin, test ortamınızı yapılandırmak, bazı ayarları değiştirmek veya özel eklentiler eklemek için bu dosyayı düzenleyebilirsiniz.

### package-lock.json:

Bu dosya, proje bağımlılıklarının (dependencies) kesin sürümlerini içeren dosyadır. NPM paketlerinin sürümlerini sabitlemek için kullanılır ve projenizin tekrar kurulmasında veya taşınmasında tutarlılık sağlar.

### package.json:

Bu dosya, proje hakkında bilgileri içeren dosyadır ve proje bağımlılıklarının (dependencies) ve komutlarının tanımlandığı yerdir. Projeye ilgili temel bilgileri, proje adını, yazarı, sürüm numarasını ve diğer NPM paketleri gibi bağımlılıkları burada bulabilirsiniz.

# Cypress Kullanımı





- ☐ Mocha, JavaScript'in en popüler test framework'lerinden biridir. Hem Node.js hem de tarayıcıda çalışabilen, asenkron (senkronize olmayan) testleri kolay ve keyifli hale getiren, kapsamlı özelliklere sahip bir JavaScript test framework'üdür.
- ☐ Mocha, frontend ve backend için hem senkron hem de asenkron testleri yazmamıza izin verir ve testleri hızlı bir şekilde çalıştırarak esnek ve doğru raporlama sağlar.
- ☐ Ayrıca, Mocha hata izlemeyi kolaylaştırır ve Node.js debugger'ı desteği sunar.
- ☐ Özetle, Mocha JavaScript kodlarının test edilmesi için güçlü bir araçtır ve hem geliştiricilerin hem de ekiplerin yazılımlarını güvenilir ve stabil bir şekilde geliştirmelerini sağlar.

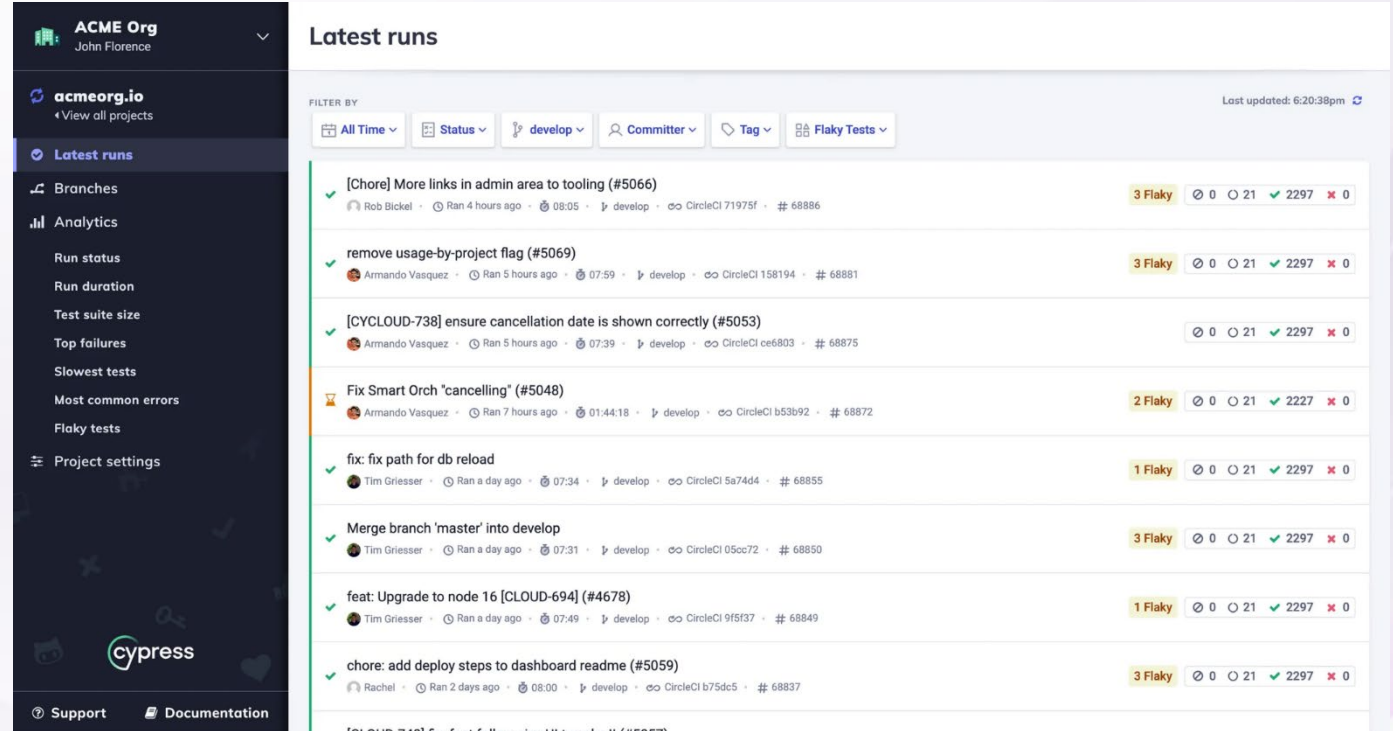


**Chai, javascript test framework ile hem node.js hem de browser için kullanılabilen bir BDD/TDD assertion kütüphanesidir.**



# Cypress Dashboard

Cypress Dashboard, Cypress test çerçevesini kullanan geliştiricilere test süreçlerini daha etkin bir şekilde yönetmelerine olanak tanıyan bir çevrimiçi hizmettir. Testlerinizi başlatmak, sonuçları gözlemlemek ve analiz etmek için kullanabileceğiniz bir platform sunar. Cypress Dashboard ile, test çalıştırmalarınızın performansını takip edebilir, hataları çözebilir ve testlerinizi daha güvenilir hale getirebilirsiniz.



The screenshot shows the Cypress Dashboard interface for 'ACME Org' (John Florence). The left sidebar contains navigation links: 'acmeorg.io', 'Latest runs', 'Branches', 'Analytics', 'Run status', 'Run duration', 'Test suite size', 'Top failures', 'Slowest tests', 'Most common errors', 'Flaky tests', and 'Project settings'. The main area displays 'Latest runs' with a filter bar (All Time, Status, develop, Committer, Tag, Flaky Tests) and a table of test runs. The table includes columns for test name, author, time, status, and results. The runs listed are:

- [Chore] More links in admin area to tooling (#5066) - Rob Bickel - Ran 4 hours ago - 08:05 - develop - CircleCI 71975f - # 68886 - 3 Flaky - 0 0 21 - 2297 - 0
- remove usage-by-project flag (#5069) - Armando Vasquez - Ran 5 hours ago - 07:59 - develop - CircleCI 158194 - # 68881 - 3 Flaky - 0 0 21 - 2297 - 0
- [CYCLOUD-738] ensure cancellation date is shown correctly (#5053) - Armando Vasquez - Ran 5 hours ago - 07:39 - develop - CircleCI ce6803 - # 68875 - 0 0 21 - 2297 - 0
- Fix Smart Orch "cancelling" (#5048) - Armando Vasquez - Ran 7 hours ago - 01:44:18 - develop - CircleCI b53b92 - # 68872 - 2 Flaky - 0 0 21 - 2227 - 0
- fix: fix path for db reload - Tim Griesser - Ran a day ago - 07:34 - develop - CircleCI 5a74d4 - # 68855 - 1 Flaky - 0 0 21 - 2297 - 0
- Merge branch 'master' into develop - Tim Griesser - Ran a day ago - 07:31 - develop - CircleCI 05cc72 - # 68850 - 3 Flaky - 0 0 21 - 2297 - 0
- feat: Upgrade to node 16 [CLOUD-694] (#4678) - Tim Griesser - Ran a day ago - 07:49 - develop - CircleCI 9f5f37 - # 68849 - 1 Flaky - 0 0 21 - 2297 - 0
- chore: add deploy steps to dashboard readme (#5059) - Rachel - Ran 2 days ago - 08:00 - develop - CircleCI b75dc5 - # 68837 - 3 Flaky - 0 0 21 - 2297 - 0

## Cypress Dashboard'un Özellikleri:

**1.Paralel ve Dağıtık Test Çalıştırmaları:** Cypress Dashboard, paralel ve dağıtık test çalıştırmalarını destekler. Bu sayede test süreçlerini hızlandırabilir ve daha verimli kullanabilirsiniz.

**2.Test Geçmişi ve Sonuçları:** Tüm test geçmişinizi ve sonuçlarınızı gözlemleyebilirsiniz. Başarılı ve başarısız testler hakkında detaylı bilgilere erişebilir ve başarısız testleri analiz ederek sorunları tespit edebilirsiniz.

**3.Dashboard ve Raporlama:** Görsel bir dashboard aracılığıyla test çalıştırmalarınızın durumunu takip edebilirsiniz. Ayrıca, test sonuçlarınızı paylaşmak için raporlama seçenekleri de sunar.

**4.Debug ve Video Kayıtları:** Başarısız testlerin nedenlerini daha iyi anlamak için, Cypress Dashboard'un içinde hataları tekrar üretebilmenizi sağlayan özel debug seçenekleri vardır. Ayrıca, testlerinizin video kayıtlarını izleyerek testin nasıl gerçekleştiğini gözlemleyebilirsiniz.

**5.GitHub Entegrasyonu:** Projenizi GitHub ile entegre edebilir ve test çalıştırmalarınızı GitHub iş akışlarına dahil edebilirsiniz.

## Nasıl Kullanılır?

Cypress Dashboard'u kullanmak için öncelikle Cypress Dashboard hesabı oluşturmanız gerekebilir. Daha sonra, projenizi Cypress Dashboard'a entegre ederek test çalıştırmalarınızın sonuçlarını görsel bir arayüzle izleyebilir ve analiz edebilirsiniz. Bu sayede test süreçlerinizi iyileştirebilir ve uygulamanızın kalitesini artırabilirsiniz.

Cypress Dashboard, ekip içinde testlerin yönetimi ve paylaşımı için güçlü bir araçtır ve test otomasyon sürecini daha verimli hale getirir.





## SELENIUM İLE ARASINDAKİ MİMARİ FARKLILIKLAR NELERDİR?

- ☐ Selenium, tarayıcının dışında çalışır ve ağ üzerinden uzak komutları çalıştırarak süreci tamamlar.
- ☐ Ancak Cypress doğrudan tarayıcının içinde çalışır. Yani aslında test kodunuzu yürüten tarayıcıdır.
- ☐ Bu, Cypress'in çalışma zamanında domu işleyerek ağ isteklerini ve yanıtlarını anında değiştirerek tarayıcı davranışını dinlemesini ve değiştirmesini sağlar.



## cy press

Cypress, test kodunun doğrudan tarayıcı içinde çalıştığı benzersiz bir mimari izler. Test edilen uygulamaya doğrudan erişebilir ve kontrol edebilir, hızlı ve güvenilir test yürütme sağlar.

Cypress öncelikle JavaScript tabanlıdır ve geliştirici dostu bir API kullanır. Test yazmak için tanıdık JavaScript sözdiziminden ve komutlarından yararlanır.

Cypress, Chrome ve Firefox gibi modern tarayıcıları destekler. Daha yeni tarayıcı özellikleri ve API'ler için daha iyi desteğe sahiptir.

Cypress, tarayıcı içinde gerçek zamanlı yeniden yüklemeye izin veren ve anında geri bildirim sağlayan testler yürütür. Eşzamansız davranışı otomatik olarak işleyebilir ve manuel bekleme ihtiyacını ortadan kaldırır.

Cypress, gerçek zamanlı yeniden yükleme, etkileşimli adım adım yürütme ve zamanda yolculuk gibi güçlü hata ayıklama araçları sunarak testlerde sorun gidermeyi ve hata ayıklamayı kolaylaştırır.

Cypress'in daha küçük ama büyüyen bir topluluğu var. Eklentiler ve topluluk tarafından sağlanan araçlarla aktif bir ekosisteme sahiptir.



## Selenium

Selenium bir istemci-sunucu mimarisi kullanır. Test kodu ile tarayıcı arasında aracı görevi gören bir WebDriver kullanarak tarayıcıyla iletişim kurar. Selenium testleri, tarayıcıdan ayrı bir işlemde çalışır.

Selenium, Java, C#, Python, Ruby vb. gibi çoklu programlama dillerini destekler. Test yazmak için dille özel bağlamalar ve API'ler sağlar.

Selenium daha geniş bir tarayıcı desteğine sahiptir ve Chrome, Firefox, Safari, Internet Explorer ve daha fazlası dahil olmak üzere çok çeşitli tarayıcılarla çalışabilir.

Selenium, WebDriver'ı kullanarak testleri harici olarak tarayıcıya yürütür. Eşzamansız davranışları ve uygulamayla etkileşimleri işlemek için açık bekleme ve eşitleme tekniklerine dayanır.

Selenium, geliştirme ortamı aracılığıyla hata ayıklama yetenekleri sağlar. Geliştiriciler, tercih ettikleri IDE veya araçların hata ayıklama özelliklerini kullanabilir.

Selenium, kapsamlı kaynaklara, çerçevelere ve kitaplıklara sahip geniş ve köklü bir topluluğa sahiptir.



# Temel Cypress Komutları

## 1. cy.visit():

cy.visit() komutu, belirtilen URL'ye gitmek için kullanılır.  
Test senaryolarının belirli bir web sayfasını açmakla başlamasını sağlar.

```
cy.visit('https://www.example.com')
```

## 2. cy.title():

cy.title() komutu, mevcut sayfanın başlığını almak için kullanılır.  
Başlığın doğruluğunu kontrol etmek veya loglamak için kullanılabilir.

```
cy.title().should('include', 'Example')
```

### 3. cy.log():

cy.log() komutu, bir mesajı Cypress komut günlüğüne yazmak için kullanılır. Test sürecini takip etmek ve hata ayıklamak için kullanışlıdır.

```
cy.log('Bu bir test mesajıdır.')
```

### 4. cy.url():

cy.url() komutu, mevcut sayfanın URL'sini almak için kullanılır. URL'yi kontrol etmek veya loglamak için kullanılabilir.

```
cy.url().should('eq', 'https://www.example.com')
```

## 5. cy.location():

cy.location() komutu, sayfanın konum bilgisini almak için kullanılır. Sayfa URL'si, host, protokol ve diğer konum bilgilerini almak için kullanılabilir.

```
cy.location().should((location) => {  
  expect(location.host).to.eq('www.example.com')  
  expect(location.protocol).to.eq('https:')  
})
```

## 6. cy.get():

cy.get() komutu, HTML öğelerini seçmek ve bu öğeler üzerinde işlem yapmak için kullanılır. CSS selektörleri veya özel seçici fonksiyonlar ile öğeleri seçmek mümkündür.

```
cy.get('.login-button').click()
```

id → #      class → .



**npx cypress open:** Test tipi ve kullanılacak browser seçimi yaptırarak testi çalıştırır

**npx cypress run:** Electron headLess browser ile açmadan ile bütün testleri çalıştırır.  
Hata alınan testlerin ekran görüntülerini screenshots klasörüne koyar  
Bütün testlerin video kayıtlarını videos klasörüne koyar.

**npx cypress info:** Bilgisayarınızda yüklü browser bilgilerini, cypress versionunu, işletim sistemini, hafıza bilgileri gibi verileri gösterir.

**npx cypress verify:** Cypress 'in yüklü olduğunu onaylar

**npx cypress run --browser chrome:**

Chrome headless(browserlı açmadan) ile bütün testleri çalıştırır

**npx cypress run ——browser chrome —headed:**

Chrome browser 'ı açarak bütün testleri çalıştırır.

## İSTENEN KLASÖRDEKİ TÜM TESTLERİ ÇALIŞTIRMAK

**npx cypress run --spec 'cypress/e2e/day02/'**

\*Default olarak Electron ile çalıştırır.

**npx cypress run --spec 'cypress/e2e/day02/' --browser chrome**

\*day02 klasöründeki bütün testleri Chrome headless (browser açmadan) ile çalıştırır.

**npx cypress run --spec 'cypress/e2e/day02/' --browser chrome --headed**

\*day02 klasöründeki bütün testleri Chrome browser ile açarak çalıştırır.

## TEK BİR TESTİ ÇALIŞTIRMAK

**npx cypress run --spec 'cypress/e2e/day02/01\_LoginTest1.cy.js'**

\* Default olarak Electron ile çalıştırır.

**npx cypress run 'cypress/e2e/day02/01\_LoginTest1.cy.js' --browser chrome**

\*day02 klasöründeki 01\_LoginTest1.cy.js' dosyasını Chrome açmadan çalıştırır

**npx cypress run 'cypress/e2e/day02/01\_LoginTest1.cy.js' --browser chrome --headed**

\*day02 klasöründeki 01\_LoginTest1.cy.js' dosyasını Chrome açarak çalıştırır



```
// AÇIKLAMA: Kullanıcı Girişi Testleri
describe('Kullanıcı Girişi Testleri', () => {
  // AÇIKLAMA: Geçerli kullanıcı adı ve şifre ile giriş yapma testi
  it('Geçerli kullanıcı adı ve şifre ile giriş yapabilmeli', () => {
    // Buraya test senaryosu ve kodları eklenecek.
    // Örneğin:
    // cy.visit('https://www.example.com/login');
    // cy.get('#username').type('gecerli_kullanici');
    // cy.get('#password').type('gecerli_sifre');
    // cy.get('.login-button').click();
    // cy.url().should('eq', 'https://www.example.com/dashboard');
  });
});
```

## describe Bloğu:

- "describe" blokları, test senaryolarını mantıksal bir şekilde gruplamak için kullanılır.
- Bir "describe" bloğu, testleri birbiriyle ilişkili oldukları bir konsept, özellik veya bileşen etrafında düzenlemek için kullanılabilir.
- "describe" bloğu, içindeki "it" bloklarını gruplayarak testlerin düzenli ve anlaşılır olmasını sağlar.
- "describe" bloğunun içinde ayrıca başka "describe" blokları da olabilir, böylece daha derin hiyerarşiler oluşturmak mümkündür.

## it Bloğu:

- "it" blokları, tek bir test senaryosunu temsil eder.
- "it" blokları, bir "describe" bloğu içinde yer almalıdır ve belirli bir test durumunu açıklamak için kullanılır.
- "it" bloğu, testin neyi kontrol ettiğini ve beklenen sonuçları belirlemek için kullanılabilir.

# Cypress Plugin Kullanımı

<https://www.npmjs.com/>



cypress-fixture-faker

**faker.js**

cypress-postgresql



cypress-mysql

**cypress.io**  
cucumber



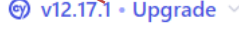
cypress-cucumber-preprocessor

<https://www.npmjs.com/> sitesinden kullanıma sunulan plugin paketleri belirtilen kullanım şekilleri, eklenecek json ve config komutları ile tüm pluginler kullanılabilir

Upgrade butonu ile Cypress update eder


Docs butonu ile Cypress.io adresine gider

Login butonu ile Dashboarda bağlanılır



Welcome to Cypress!


Review the differences between each testing type →



### E2E Testing

Build and test the entire experience of your application from end-to-end to ensure each flow matches your expectations.

Configured

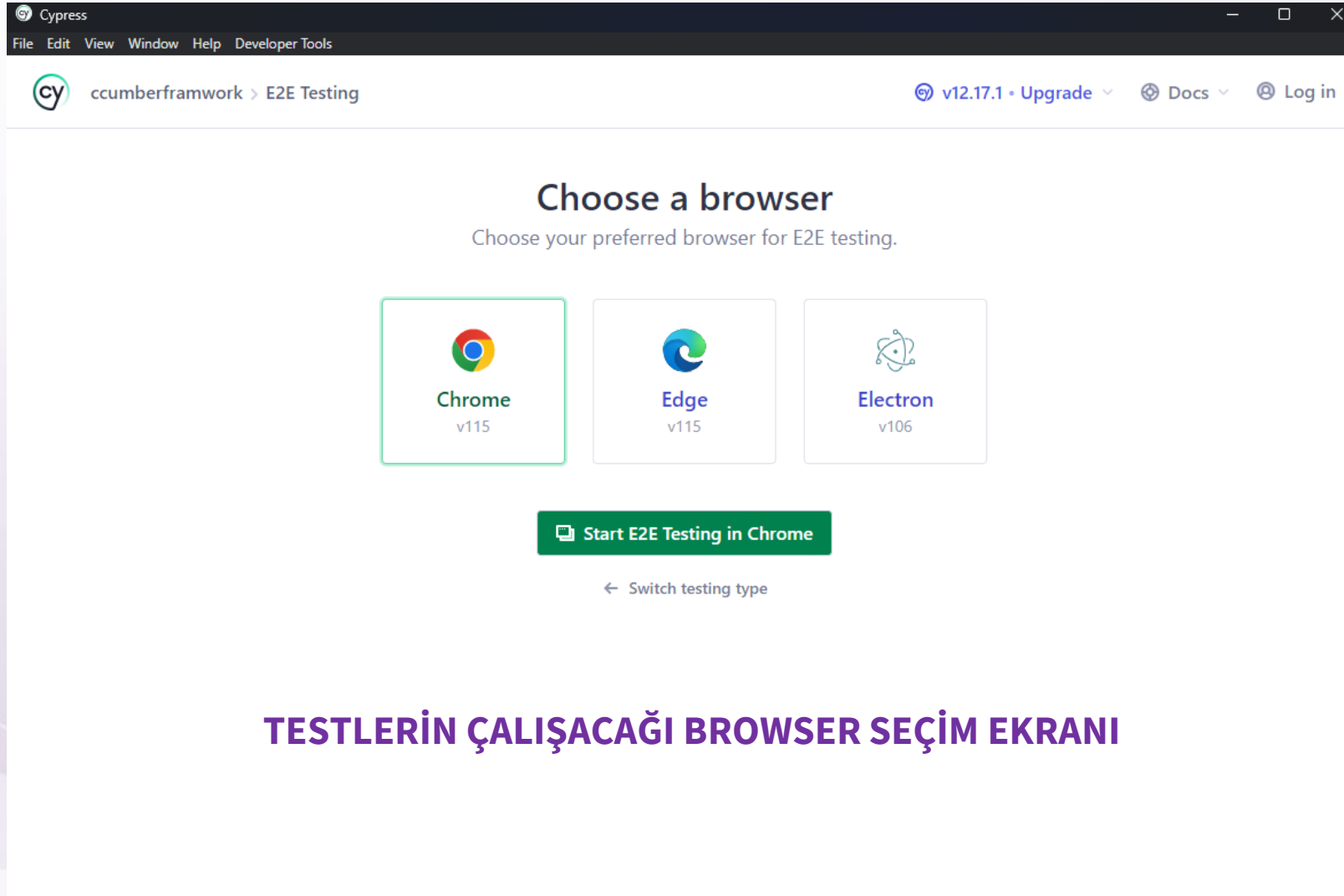


### Component Testing

Build and test your components from your design system in isolation in order to ensure each state matches your expectations.

Not Configured

## TEST TÜRÜ SEÇİM EKRANI



## TESTLERİN ÇALIŞACAĞI BROWSER SEÇİM EKRANI