

# *Rails ile Frontend Kodlama Eğitimi*

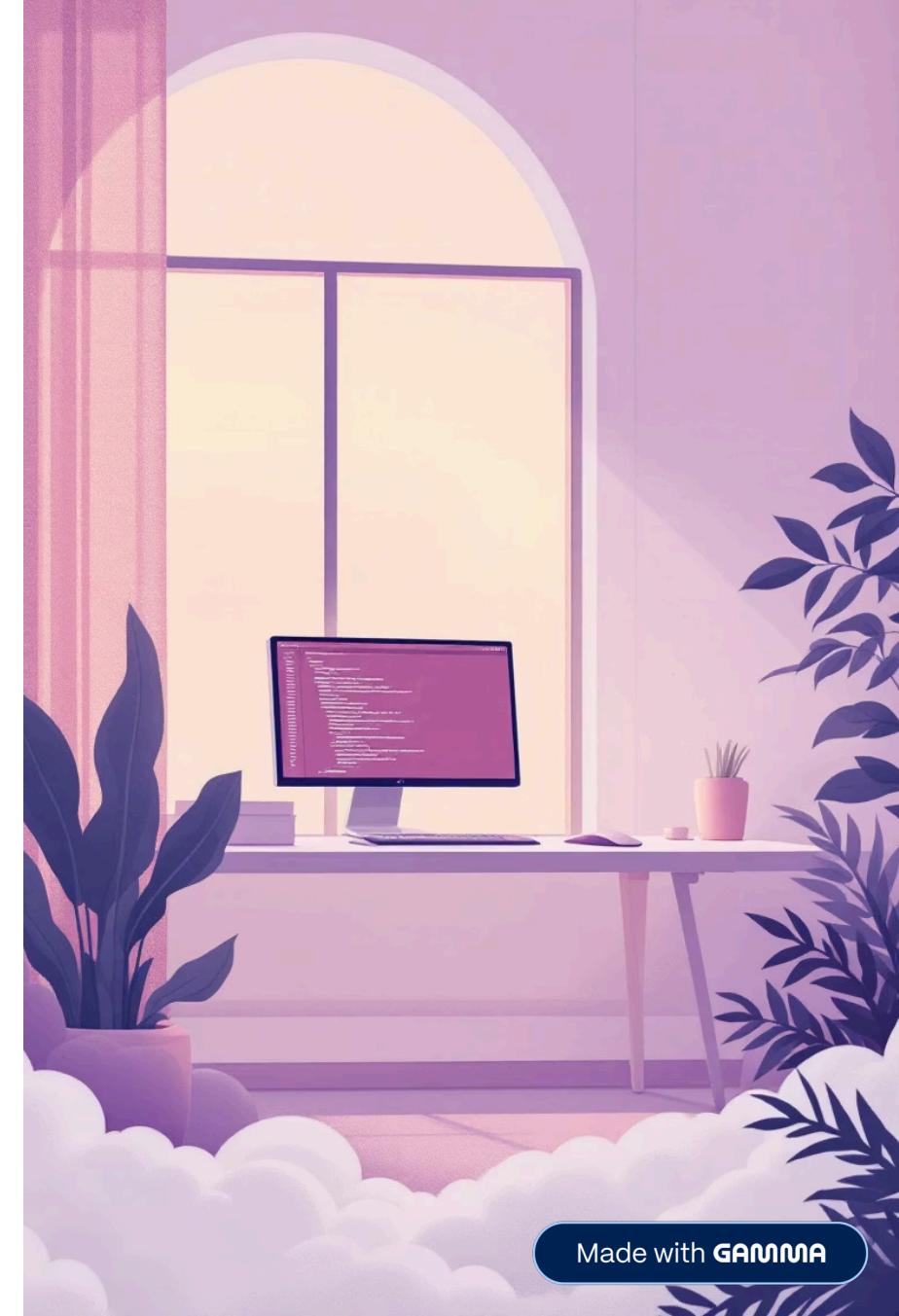
«Ruby veya Rails Bilmeniz Gerekmez!»

Eğitmen: Levent Özbilgiç

[LinkedIn](#) | [YouTube](#) | [Instagram](#)

# Rails ile Frontend Kodlama Eğitimi

- Temel Bilgiler ve Dosya Yapısı
- Fontlar, Görseller, HTML, CSS... Kullanım Örnekleri
- Stimulus ile Etkileşimli Basit Örnekler
- Rails Frontend CLI
- SSS
- Bonus: Bilinmesi Gereken Birkaç Komut



# Eğitim için yeni proje oluştur

Eğitime başlamak için yeni bir proje oluşturup üzerinde denemeler yapmanız başarınızı oldukça artıracaktır. Yeni bir proje başlatmak için öncelikle [Rails Frontend CLI](#) aracını sisteminize kurup terminalizde aşağıdaki komutu yürütün.

```
rails-frontend new proje_ismi --clean
```

Bu komut belirttiğiniz isimde frontend odaklı bir proje oluşturur.

# Rails View ve Partials Temelleri

## ERB Şablonları (HTML)

ERB (Embedded Ruby) ile HTML dosyalarınızı oluşturun. Statik HTML yazarken tüm bildığınız HTML, CSS yapısını kullanabilirsiniz. Ruby yada Rails bilmeniz gerekmekz.

## Partials ile Modülerlik

Kod tekrarını önlemek için partial'lar kullanın. Bir komponenti bir kez yazın, her yerde kullanın. Partial'lar `_header.html.erb` şeklinde alt çizgi ile başlar ve tüm partial'lar `app/views/shared` klasöründe oluşturulur. Bu konu Partials (HTML parçaları) eğitim kartında ayrıntılı öğrenilebilir.

## Dosya Yapısı

Dosya yapısı frontend geliştirmeye uygun olacak şekilde yapılandırılmıştır. Yandaki resimde örnek bir çalışma alanı görebilirsiniz. Bu çalışma ekranının yanına kodladığınız web sitesini açın. Her şey tek ekranada sizin kontrolünüzde. ❤️

The screenshot shows a code editor with several files open, illustrating the Rails view structure:

- application.html.erb**: The main layout file containing the structure for the entire application.
- hakkimizda.html.erb**: A specific view file for the "Hakkımızda" page.
- hakkimizda.css**: The corresponding CSS file for the "Hakkımızda" page.
- hakkimizda\_controller.js**: The Stimulus controller for the "Hakkımızda" page.

The sidebar shows the project structure:

- app**:
  - assets**: fonts, images, stylesheets (application.css, footer.css, hakkimizda.css, header.css, home.css, navbar.css)
  - javascript**: controllers (application.js, hakkimizda\_controller.js, hello\_controller.js, home\_controller.js, index.js), views (home, hakkimizda.html.erb, index.html.erb), layouts (application.html.erb)
  - shared**: \_footer.html.erb, \_header.html.erb, \_navbar.html.erb, \_neden\_biz.html.erb
- config**: environment.js, routes.js
- db**: migrations
- public**: assets, cache, favicon, robots
- tmp**: sessions, socket, tailwind
- views**: shared

## ::Görseller (.jpg, .png, .svg vs...)

- ❑ Görseller app/assets/images klasörüne atılır. Görseller html dosyası içine image\_tag komutu ile çağrılır.

### Örnek Kullanım:

```
<%= image_tag 'logo.png', class: 'logo-css' %>
```

Daha detaylı kullanım örnekleri için bakın: [image\\_tag kullanım örnekleri](#)

The screenshot shows a code editor interface with a dark theme. On the left, the Explorer sidebar displays a project structure under 'FRO TEST [WSL: UBUNTU]'. It includes a 'app' folder containing 'assets' (with 'images' subfolder containing 'image1.svg' and 'image2.svg'), 'controllers', 'models', and 'views' (with 'home' folder containing 'index.html.erb'). On the right, the main editor area shows the content of 'index.html.erb':

```
1 <div data-controller="home">
2   <div class="gradient bg-gradient-to-br from-blue-50 to-indigo-500">
3     <div class="container m-4 p-4 py-10">
4       <div class="text-center">
5         <h1 class="text-h1 font-bold text-gray-900 mb-4">
6           mes verdim!
7         </h1>
8       </div>
9       <%= image_tag "image1.svg", class: "w-32 h-32 mx-auto mb-4" %>
10      <button data-action="click->home#toggle" class="bg-blue-500 hover:bg-blue-600 border border-white">
11        Detayları Göster!
12      </button>
13    </div>
14  </div>
</div>
```

## ::Linkler

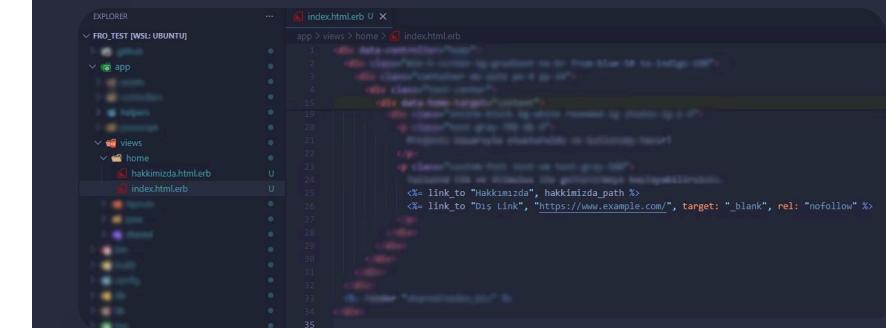
- ☐ Html dosyalarınızda link oluşturmak için `<a>` etiketi yerine `link_to` komutu kullanılır. Bu komut belirttiğiniz isimde, belirttiğiniz adrese giden bir link oluşturur. Mevcut sayfalarınıza link oluşturmak için şu yapıyı kullanın: `sayfaadi_path`

### Örnek Kullanım:

```
<%= link_to "Hakkımızda", hakkimizda_path %>
<!-- <a href="/hakkimizda">Hakkımızda</a> -->

<%= link_to "Dış Link Örneği", "https://www.example.com/", target: "_blank", rel: "nofollow" %>
<!-- <a href="https://www.example.com/" target="_blank" rel="nofollow">Dış Link Örneği</a> -->
```

Daha detaylı kullanım örnekleri için bakın: [link\\_to kullanım örnekleri](#)



## ::Fontlar

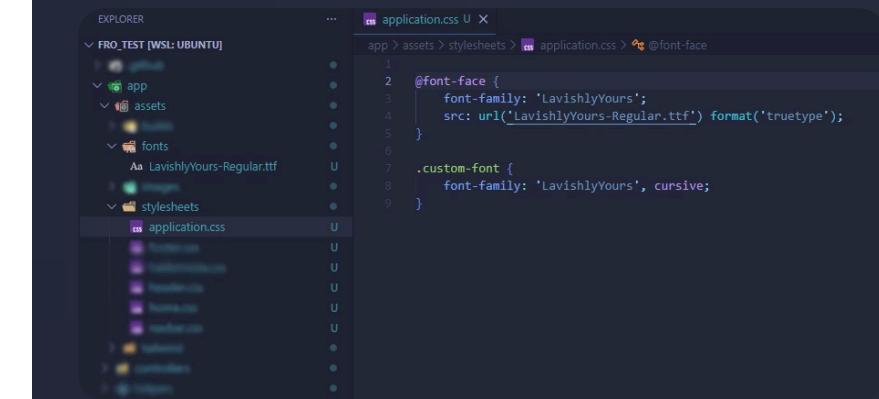
- Fontlar app/assets/fonts klasörüne atılır yada app/views/layouts/application.html.erb dosyasına arzuladığınız font'un CDN linkini ekleyebilirsiniz.

Fonts klasörüne attığınız bir fontun tüm sayfalarda kullanılabilir olmasını istiyorsanız bu fontu app/assets/stylesheets/application.css dosyasında tanımlayın.

### Örnek Kullanım Senaryosu:

Fonts dosyasına "LavishlyYours-Regular.ttf" adında bir font attığınızı düşünelim. application.css dosyasına aşağıdaki tanımı yaparak artık tüm sayfalarda bu fontu kullanabilirsiniz.

```
@font-face {  
    font-family: 'LavishlyYours';  
    src: url('LavishlyYours-Regular.ttf') format('truetype');  
}
```

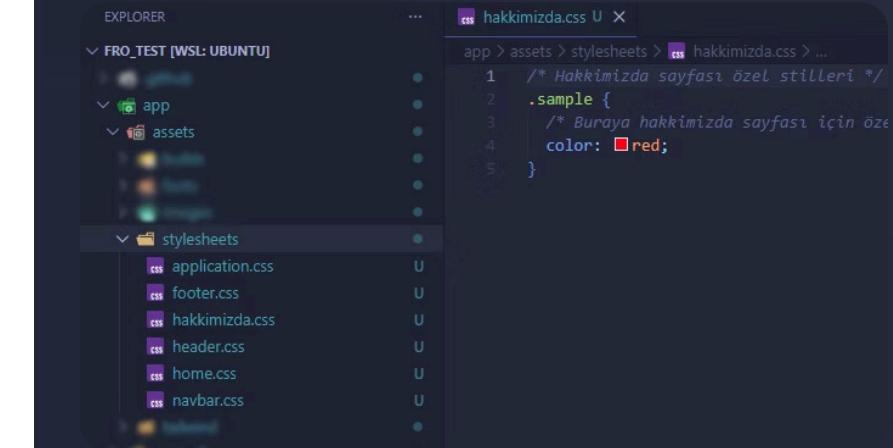


## ::CSS

- CSS/SCSS dosyaları app/assets/stylesheets klasörüne atılır. –  
Bulduğum bir css dosyasını stylesheets klasörüne kopyaladım,  
projeye eklemem için ne yapmam gerekiyor? Cevap: Hiçbir şey!  
Rails sistemi bu dosyaya atılan stil dosyalarını otomatik olarak  
projenize dahil eder. Sadece stil dosyasının yüklenme sırası sizin  
için önemliyse application.css dosyası içinde yükleme sırasına  
göre @import etmeniz gereklidir.

Peki SCSS de nedir? Dosya uzantınızı .scss'ye çevirmeniz  
durumunda css içine değişkenler ve kodlar eklemenizi sağlar.  
Ayrıntılı bilgi için bakın: [sass-lang.com](https://sass-lang.com)

- ⚠ app/assets/stylesheets/application.css dosyasında yapılan  
değişiklikler tüm html sayfalarını etkiler. Yani application.css  
dosyası tüm html sayfaları için ana stil dosyasıdır.



The screenshot shows the VS Code interface with the Explorer, Editor, and Search panes visible. The Explorer pane shows a project structure under 'FRO\_TEST [WSL: UBUNTU]'. The 'app' folder contains 'assets' and 'stylesheets' subfolders. 'assets' contains 'application.css', 'footer.css', 'hakkimizda.css', 'header.css', 'home.css', and 'navbar.css'. The 'stylesheets' folder contains 'application.css', 'footer.css', 'hakkimizda.css', 'header.css', 'home.css', and 'navbar.css'. The Editor pane displays the 'hakkimizda.css' file with the following content:

```
/* Hakkımızda sayfası özel stilleri */
.sample {
    /* Buraya hakkımızda sayfası için özel stil ekle
    color: red;
}
```

::HTML

- HTML dosyaları `app/views/home` klasöründe oluşturulur. Rails Frontend CLI aracı kullanılarak bir proje oluşturduğunuzda bu klasörde `index.html.erb` dosyası ön tanımlı olarak gelir ve bu ana sayfanızdır. Yeni sayfalar Rails Frontend CLI aracı kullanılarak eklenebilir veya silinebilir. Eklenen tüm sayfalar otomatik olarak `app/views/home` klasöründe oluşturulur.

-  Bulundığınız sayfanın başlığını değiştirmek için aşağıdaki satırı ekleyin. `hakkimizda.html.erb` dosyasının içinde:

<% content\_for :title, "Hakkımızda" %>

Tüm sayfalarda geçerli olmasını istediğiniz varsayılan başlığı application.html.erb dosyasında <title> ile başlayan satırdaki tırnak işaretleri arasına yazın.

**content\_for** ile istediğiniz bilgileri ana şablon dosyanıza gönderebilirsiniz. Örnekler için **content\_for** eğitim kartına bakın.

- ⚠️ app/views/layouts/application.html.erb dosyasında yapılan değişiklikler app/views/home dosyasındaki tüm html sayfalarını etkiler. Yani application.html.erb dosyası tüm diğer html sayfaları için ana şablon dosyasıdır. Bu dosyanın içinde <%= yield %> adında bir satır var, app/views/home klasörü içindeki sayfalar bu satır sayesinde şablon ile birleşerek tam bir sayfa görünümü elde ederler.

## ::Partials (HTML parçaları)

- Aynı HTML kodunu birden fazla sayfada kullandığınızı farkederseniz bu kodu app/views/shared klasörü altında bir partial dosyasına atın. Artık bu kodu her yer de <%= render 'shared/dosya\_adi' %> komutu ile kullanabilirsiniz.

### Örnek Kullanım Senaryosu:

Tüm sayfalarda tepede görünmesini istediğiniz bir menü ve sayfanın altında görünmesini istediğiniz bir bilgi alanı olduğunu düşünelim. Bu durumda menü için app/views/shared klasörü altında \_header.html.erb isminde bir dosya ve bilgi alanı için ise \_footer.html.erb isminde bir dosya oluşturup kodunuzu yazdırınız. Artık bu dosyaları app/views/layouts/application.html.erb dosyası içinde çağrıbilir ve tüm sayfalarınızda görünmesini sağlayabilirsiniz. Dilerseniz herhangi bir sayfada da çağrıbilirsiniz.

```
<div class="container">
  <%= render 'shared/header' %>

  <main> <%= yield %> </main>

  <%= render 'shared/footer' %>
</div>
```

Partial'lar \_header.html.erb şeklinde alt çizgi ile başlar ve tüm partial'lar app/views/shared klasöründe oluşturulur.

The screenshot shows a code editor interface with a dark theme. On the left, the Explorer sidebar displays a project structure under 'FRO\_TEST [WSL: UBUNTU]'. It includes 'app', 'views' (with 'header.html.erb' and 'footer.html.erb' files), 'layouts' (with 'application.html.erb'), and 'shared' (with '\_header.html.erb', '\_footer.html.erb', '\_navbar.html.erb', and '\_neden\_biz.html.erb'). The main editor area shows the 'application.html.erb' file with the following content:

```
1 <html>
2   <head>
3     <title>Rails Application</title>
4   </head>
5   <body>
6     <%= render 'shared/header' %>
7     <main class="min-h-screen">
8       <%= yield %>
9     </main>
10    <%= render 'shared/footer' %>
11  </body>
12 </html>
```

## ::content\_for

- ☐ Bu komut ile sayfaya özel istediğiniz bilgileri ana şablon dosyası olan `app/views/layouts/application.html.erb` dosyasına gönderebilirsiniz. Örneğin bulunduğuuz sayfaya özel meta etiketleri yada sadece o sayfaya özel javascript eklemeniz gerekiyorsa `content_for` kullanabilirsiniz.

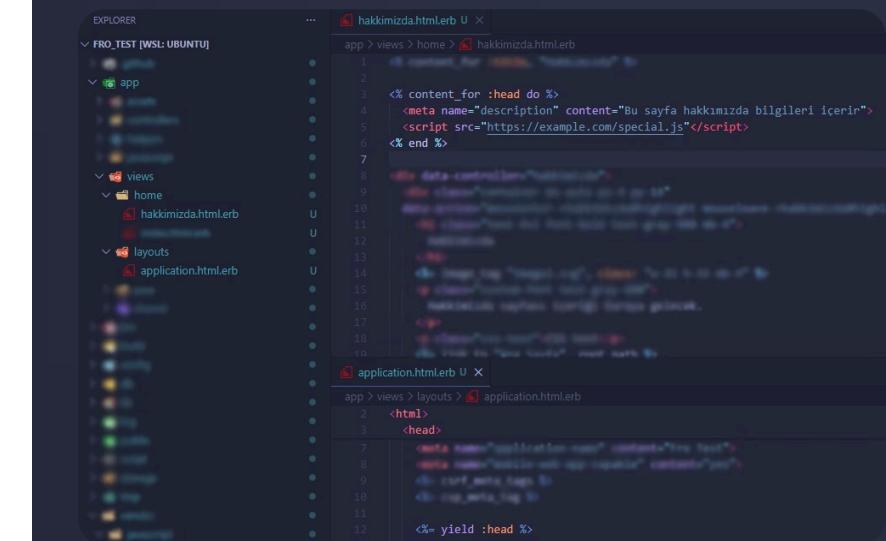
### Örnek Kullanım Senaryosu:

`hakkimizda.html.erb` dosyanıza özel sayfa açıklaması `<meta>` etiketi ve bir javascript eklemeniz gerektiğini düşünelim.

```
<% content_for :head do %>
  <meta name="description" content="Bu sayfa hakkımızda
bilgileri içerir">
  <script src="https://example.com/special.js"></script>
<% end %>
```

Yukardaki kodun çalışabilmesi için  
`app/views/layouts/application.html.erb` dosyanızda şu satırın  
bulunması yeterlidir: `<%= yield :head %>` (bu satır varsayılan  
olarak zaten bulunur)

- ⓘ Bu yapı ile `content_for`'a istediğiniz bir isim vererek ve şablon dosyanıza `<%= yield :verdiginiz_isim %>` satırını ekleyerek sayfaya özel bilgileri kullanabilirsiniz.



The screenshot shows a code editor interface with the following details:

- EXPLORER:** Shows the project structure: `FRO_TEST [WSL: UBUNTU]`, `app`, `views` (containing `home` and `hakkimizda.html.erb`), and `layouts` (containing `application.html.erb`).
- hakkimizda.html.erb:** Contains the code: 

```
<% content_for :head do %>
  <meta name="description" content="Bu sayfa hakkımızda bilgileri içerir">
  <script src="https://example.com/special.js"></script>
<% end %>
```
- application.html.erb:** Contains the code: 

```
<html>
  <head>
    <meta name="content_for_head" content="This is the head content">
    <meta name="verdiginiz_isim" content="You can use this variable">
    <meta name="layout" type="text/html" content="main">
    <meta name="meta_tag" type="text/html" content="meta tag content">
  </head>
  <%= yield :head %>
```

# ::JAVASCRIPT

- Rails HTML'i zenginleştirmek (html elemanlarına davranış eklemek, manipüle etmek vs...) için stimulus isminde javascript kütüphanesi kullanır. `app/javascript/controllers` klasöründe istediğiniz isimde controller dosyası oluşturabilir ve tüm sayfalarda tekrar tekrar kullanılabilir bileşenler yazabilirsiniz.

## Örnek Kullanım Senaryosu:

`toogle_controller.js` adında bir dosya oluşturduğunuza varsayılmış. Herhangi bir sayfadaki html elementine `data-controller="toggle"` özelliği eklerseniz, `toogle_controller.js` dosyanız bu sayfaya çağrırlı.

Ayrıntılı bilgi ve kullanım örnekleri için aşağıdaki Stimulus eğitim kartlarını okuyun.

- ⚠️** `app/javascript/application.js` dosyasında yapılan değişiklikler tüm html sayfalarını etkiler. Yani `application.js` dosyası tüm html sayfaları için ana javascript dosyasıdır. Bu dosya genellikle diğer hazır javascript kütüphanelerini projenize dahil etmek (import) için kullanılır. Bu konuya Harici Javascript Kütüphanesi Eklemek eğitim kartında ayrıntılı olarak öğrenin.

The screenshot shows a code editor interface with two tabs open. The top tab is titled 'JS toggle\_controller.js' and contains the following code:

```
1 import { Controller } from '@hotwired/stimulus'
2
3 export default class extends Controller {
4   static targets = ["content"]
5
6   toggle() {
7     this.contentTarget.classList.toggle("hidden")
8   }
9 }
```

The bottom tab is titled 'hakkimda.html.erb' and contains the following HTML code:

```
3 <div data-controller="toggle">
4   <button data-action="click->toggle#toggle">Göster/Gizle</button>
5
6   <div data-toggle-target="content">
7     Bu içerik açılıp kapanacak
8   </div>
9 </div>
```

The left side of the interface shows a file explorer with a tree view of the project structure under 'FRO\_TEST [WSL: UBUNTU]'. The 'controllers' folder contains the 'toggle\_controller.js' file. The 'views' folder contains the 'hakkimda.html.erb' file.

# Stimulus: Minimalist JavaScript Framework

01

## Controller

JavaScript davranışlarını organize eden temel yapı taşı

02

## Target

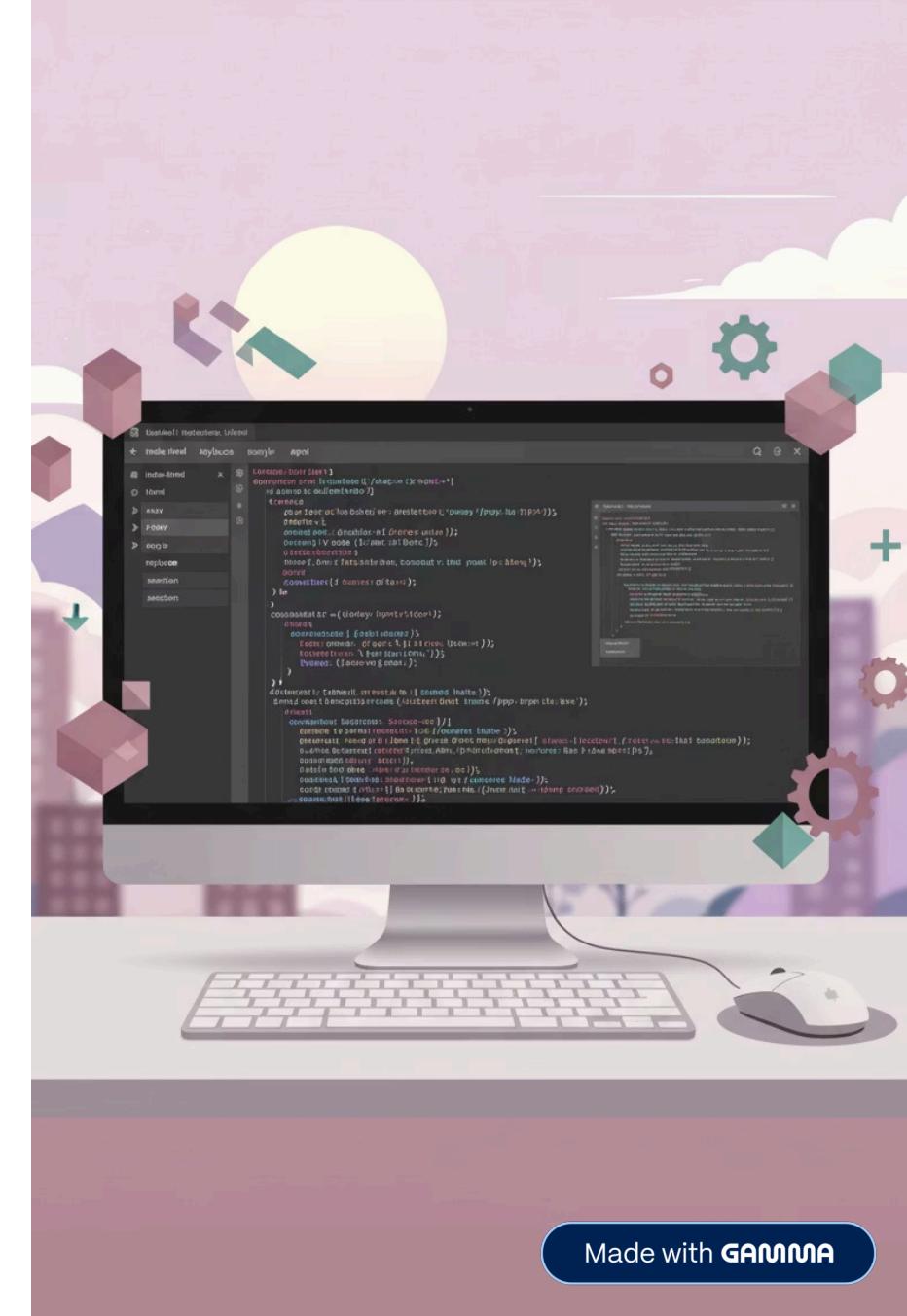
Controller'in etkileşime geçeceği HTML elemanlarını işaretler

03

## Action

Kullanıcı etkileşimlerini (click, submit vb.) controller metodlarına bağlar

- ❑ **Temel Felsefe:** Stimulus, mevcut HTML'ine JavaScript davranışını ekler. Sayfayı sıfırdan oluşturmaz, var olanı zenginleştirir.



# Stimulus Kurulumu ve İlk Controller



## Varsayılan Kurulum

Rails 7+ ile Stimulus otomatik gelir, ekstra kurulum gerekmeyez.



## Controller Oluşturma

app/javascript/controllers/  
klasöründe yeni controller dosyası  
açın



## Kodu Yazma

Basit bir toggle controller ile başlayın  
ve test edin

## toggle\_controller.js

```
import { Controller } from "@hotwired/stimulus"

export default class extends Controller {
  static targets = ["content"]

  toggle() {
    this.contentTarget.classList.toggle("hidden")
  }
}
```

## HTML'de Bağlama

```
<div data-controller="toggle">
  <button data-action="click->toggle#toggle">
    Göster/Gizle
  </button>

  <div data-toggle-target="content">
    Bu içerik açılıp kapanacak
  </div>
</div>
```



# Uygulamalı Örnek: Buton ile İçerik Göster/Gizle



## 1. Controller Tanımla

`data-controller="toggle"` ile div'i işaretle

## 2. Action Bağla

`data-action="click->toggle#toggle"` butona ekle

## 3. Target Belirle

`data-toggle-target="content"` ile erişilecek elementi işaretle

"Stimulus ile DOM manipülasyonu son derece temiz ve anlaşılır. Her elemanın rolü HTML attribute'larında açıkça belirtilir."

Bu yaklaşım, JavaScript kodunu minimize ederken HTML'de neler olduğunu net şekilde gösterir.

# Form Etkileşimleri ve Dinamik Liste Yönetimi

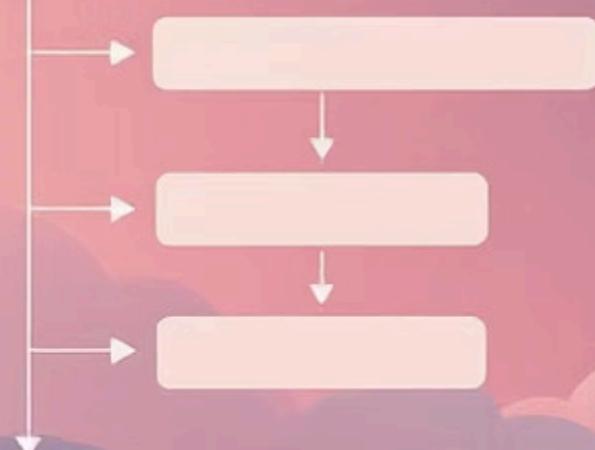
## Form Controller Örneği

```
export default class extends Controller {  
  static targets = ["input", "list", "template"]  
  
  add(event) {  
    event.preventDefault()  
    const value = this.inputTarget.value  
  
    if (value.trim()) {  
      const item = this.templateTarget  
        .content.cloneNode(true)  
      item.querySelector(".item-text")  
        .textContent = value  
      this.listTarget.appendChild(item)  
      this.inputTarget.value = ""  
    }  
  }  
}
```

## HTML Yapısı

```
<div data-controller="form">  
  <input data-form-target="input"  
    type="text" placeholder="Yeni öğe">  
  <button data-action="click->form#add">  
    Ekle  
  </button>  
  
  <ul data-form-target="list"></ul>  
  
  <template data-form-target="template">  
    <li class="item-text"></li>  
  </template>  
</div>
```

`<template>` elementi, klonlanabilir HTML parçaları için mükemmel bir çözüm.



# İyi Uygulamalar: Temiz ve Sürdürülebilir Kod

## Tek Sorumluluk Prensibi

Her controller tek bir işe odaklanmalı. Büyük controller'ları küçük parçalara bölün.

## HTML ve JS Ayrımı

JavaScript kodunda DOM selector kullanmayın. Stimulus target'ları ile temiz bağlantı kurun.

## Lifecycle Metodları

`connect()` ve `disconnect()` ile controller yaşam döngüsünü kontrol edin.

```
export default class extends Controller {  
  connect() {  
    console.log("Controller DOM'a bağlandı")  
    // Başlangıç işlemleri  
  }  
  
  disconnect() {  
    console.log("Controller DOM'dan ayrıldı")  
    // Temizlik işlemleri  
  }  
}
```

## ::Harici Javascript Kütüphanesi Eklemek

ⓘ **Otomatik Ekleme:** Rails Frontend CLI aracı ile javascript kütüphanelerini otomatik olarak projenize ekleyebilirsiniz. Bu konuyu Rails Frontend CLI eğitim kartında ayrıntılı olarak öğrenin.

☐ **El ile ekleme:** İndirdiğiniz harici javascript kütüphanelerini vendor/javascript klasörüne atın. config/importmap.rb dosyasını açın şu satırı ekleyin: pin "ozel\_kutuphane", to: "ozel\_kutuphane.js" Son olarak uygulamanıza dahil edin: app/javascript/application.js dosyanızı açın ve şu satırı ekleyin: import "ozel\_kutuphane"

### Örnek Kullanım Senaryosu:

İnternetten severek kullanılan "sweetalert2" kütüphanesini indirdiğinizi varsayalım.

1. Bu dosyayı vendor/javascript dosyasına kopyalayın.  
vendor/javascript/sweetalert2.js
2. config/importmap.rb dosyasını açın ve şu satırı ekleyin pin "sweetalert2", to: "sweetalert2.js"
3. Proje genelinde kullanabilmek için dahil edin:  
app/javascript/application.js dosyanızı açın ve şu satırı ekleyin:  
import Swal from "sweetalert2"

Kütüphaneyi stimulus controller dosyanızda da kullanabilirsiniz. Bunun için controller dosyanızı açıp import { Controller } from '@hotwired/stimulus' satırından sonra şu satırı ekleyin: import Swal from "sweetalert2"

The screenshot shows a dark-themed VS Code interface. On the left, the Explorer sidebar shows a project structure with a 'vendor' folder containing a 'javascript' folder and a file named 'sweetalert2.js'. In the center, the 'application.js' file is open, showing the following code:

```
// Import your imports here
import './stimulus_controller'
import 'bootstrap'
import 'font-awesome'

import Swal from "sweetalert2"
```

Below it, the 'importmap.rb' file is also open, showing:

```
pin "sweetalert2", to: "sweetalert2.js" # @11.26.10
```

# Rails Frontend CLI

Rails Frontend CLI aracı frontend kodlama yapan programcılar işini oldukça kolaylaştırır, ruby yada rails bilinmesine gerek kalmadan rails ile frontend kodlamayı sevdiren bir araç. Bu aracın ayrıntılı kullanım kılavuzu için bakın: [Rails Frontend CLI](#)

Komut	Komut Açıklaması
rails-frontend new blog --clean	blog isminde frontend odaklı yeni bir proje başlatır. Artık terminalinizden > <b>cd blog/</b> komutu ile proje konumuna geçebilirsiniz.
rails-frontend run	Projenizi çalıştırır. Tarayıcıda görüntülemek için şu adrese gidin: <a href="http://localhost:3000">http://localhost:3000</a> Projeniz canlı bir sunucuda çalıştırılır. Html, css veya javascript dosyalarında yaptığınız değişikleri kaydettiğinizde sunucuya yeniden başlatmanıza gerek kalmaz. Sayfayı yenilemeniz yeterlidir. Sunucuyu durdurmak için terminal ekranınızda CTRL+C tuşlarına basın. <b>İpucu:</b> Sayfayı yenilemek için F5 tuşu yerine CTRL+F5 tuşuna basarsanız o sayfaya ait tarayıcı önbelleği silinir ve her şey sunucudan tekrar yüklenir.
rails-frontend add-page	Belirttiğiniz isimde sayfa ekler. Örneğin: <b>rails-frontend add-page ürünler</b> Komut app/views/home klasörü altında urunler.html.erb sayfanızı oluşturur. Türkçe karakterler otomatik olarak çevrilir (ürünler → urunler)
rails-frontend remove-page	Mevcut bir sayfayı siler. Örneğin: <b>rails-frontend remove-page ürünler</b> Komut urunler.html.erb sayfanızı siler.
rails-frontend add-stimulus	Belirttiğiniz isimde stimulus controller ekler. Örneğin: <b>rails-frontend add-stimulus dropdown</b> Komut app/javascript/controllers klasörü altında dropdown_controller.js dosyanızı oluşturur.
rails-frontend remove-stimulus	Mevcut bir stimulus controller dosyasını siler. Örneğin: <b>rails-frontend remove-stimulus dropdown</b> Komut, silme işlemine geçmeden önce bu controller'in herhangi bir dosyada kullanılmış kullanılmadığını araştırır. Eğer bulursa sizin uyarır ve onay bekler. Y harfine basıp onaylarsanız controller silinir.
rails-frontend add-layout	Belirttiğiniz isimde layout ekler. Örneğin: <b>rails-frontend add-layout ürünler</b> Komut, layout adı ile eşleşen view dosyası arar ve bulursa app/views/layouts klasörü altında layout dosyanızı oluşturur. Bulamazsa size hangi view ile kullanacağınızı seçenek olarak sorar. Belirttiğiniz seçenek uygulanarak layout dosyanız oluşturulur.
rails-frontend remove-layout	Mevcut bir layout'u siler. Örneğin: <b>rails-frontend remove-layout ürünler</b> Komut, app/views/layouts klasörü altındaki urunler.html.erb layout dosyanızı siler.
rails-frontend add-pin	Belirttiğiniz isimde bir javascript kütüphanesi bulunursa bu kütüphaneyi projenize ekler. Örneğin: <b>rails-frontend add-pin sweetalert2</b> Komut, sweetalert2 isimli javascript kütüphanesini projenize pin'ler. app/javascript/application.js dosyanızda <b>import Swal from "sweetalert2"</b> satırını ekleyerek kullanmaya başlayabilirsiniz. Stimulus controller dosyanızda kullanmak isterken dosyanızı açıp bir önceki import satırından sonra <b>import Swal from "sweetalert2"</b> satırını ekleyip kullanabilirsiniz. <b>İpucu 1:</b> Pin'lediğiniz kütüphanenin nasıl import edileceğini bilmiyorsanız bu kütüphanenin dökümanını okuyun. <b>İpucu 2:</b> Kütüphaneleri bu siteden bulabilir ve araştırabilirsiniz. <a href="#">jsDelivr</a>
rails-frontend remove-pin	Daha önce eklediğiniz bir javascript kütüphanesini projenizden siler. Örneğin: <b>rails-frontend remove-pin sweetalert2</b> Komut, silme işlemine geçmeden önce bu kütüphanenin herhangi bir dosyada kullanılmış kullanmadığını araştırır. Eğer bulursa sizin uyarır ve onay bekler. Y harfine basıp onaylarsanız kütüphane silinir.
rails-frontend --version	Rails Frontend CLI'nın hangi versyonunu kullandığınızı gösterir.
rails-frontend --help	Terminal ekranında komut yardımlarını gösterir.

# Önemli Bilgileri Hatırla!

## application.html.erb

app/views/layouts/application.html.erb dosyasında yapılan değişiklikler app/views/home dosyasındaki tüm html sayfalarını etkiler. Yani application.html.erb dosyası tüm diğer html sayfaları için ana şablon dosyasıdır. Bu dosyanın içinde <%= yield %> adında bir satır var, app/views/home klasörü içindeki sayfalar bu satır sayesinde şablon ile birleşerek tam bir sayfa görünümü elde ederler.

## application.css

app/assets/stylesheets/application.css dosyasında yapılan değişiklikler tüm html sayfalarını etkiler. Yani application.css dosyası tüm html sayfaları için ana stil dosyasıdır.

## application.js

app/javascript/application.js dosyasında yapılan değişiklikler tüm html sayfalarını etkiler. Yani application.js dosyası tüm html sayfaları için ana javascript dosyasıdır. Bu dosya genellikle diğer hazır javascript kütüphanelerini projenize dahil etmek (import) için kullanılır. Bu konuyu Harici Javascript Kütüphanesi Eklemek eğitim kartında ayrıntılı olarak öğrenin.

# Sıkça Sorulan Sorular

## Sayfaya özel layout dosyası oluşturabilir miyim?

Evet, her sayfaya özel layout dosyası oluşturabilirsiniz! Bunun için öncelikle `app/views/layouts` konumunda istediğiniz isimde layout dosyanızı oluşturun (sayfa isminizle aynı ismi kullanmanızı öneririm.) Daha sonra `app/controllers/home_controller.rb` dosyanızı açın ve şu komutu yazın: `layout "verdiğiniz_isim", only: :sayfa_ismi`

### Örnek Kullanım Senaryosu:

İletişim sayfanız için bir layout oluşturmak istiyorsunuz. Bu durumda `app/views/layouts` konumunda `iletisim.html.erb` dosyanızı oluşturun ve kodlarınızı yazın. Daha sonra `app/controllers/home_controller.rb` dosyanızı açın ve aşağıdaki komutu `class HomeController < ApplicationController` satırının altına ekleyin.

```
layout "iletisim", only: :iletisim
```

- i Rails Frontend CLI aracını kullanarak layout dosyalarınızı otomatik olarak oluşturabilirsiniz. Bu durumda tek yapmanız gereken oluşturduğunuz layout dosyasını açıp düzenlemek.

## El ile sayfa ekleyebilir miyim?

Evet, fakat bu kesinlikle önerilmez! Bu dosya yapınızın bozulmasına ve işin içinden çıkamayacağınız teknik hatalara sebep olabilir.

Bunu yapmak için aşağıdaki 4 adımı uygulayın.

- view dosyası oluşturun:** `app/views/home` klasörünü açın ve belirlediğiniz isimde bir görünüm dosyası oluşturun. Örneğin: `ozel_sayfa.html.erb`
- stil dosyası oluşturun:** `app/assets/stylesheets` klasörünü açın ve 1. adımda oluşturduğunuz view dosyasının ismiyle aynı olacak şekilde stil dosyası oluşturun. Örneğin: `ozel_sayfa.css`
- Home controller dosyasını güncelleyin:** `app/controllers/home_controller.rb` dosyasını açın ve aşağıdaki satırı ekleyin.

```
...
def ozel_sayfa
end
...
```

- Route dosyasını güncelleyin:** `config/routes.rb` dosyasını açın ve aşağıdaki satırı ekleyin.

```
get '/ozel_sayfa', to: 'home#ozel_sayfa'
```



# Komut Listesi



## image\_tag

Html dosyalarına app/assets/images konumundaki dosyaları eklemek için kullanılır.

## link\_to

Html dosyalarına link eklemek için kullanılır.

## content\_for

Şablon dosyanızda mevcut sayfadan veri göndermek için kullanılır.  
Örneğin: sayfa başlığı bilgisi veya açıklama (description meta etiketi)

## yield

Şablon dosyanızdan gelen verileri göstermek için kullanılır.

## render

app/views/shared konumunda bulunan html parçalarını herhangi bir sayfaya çağırabilmek için kullanılır.

## Faydalı Kaynaklar

- Stimulus Resmi Dokümantasyon (en): [stimulus.hotwired.dev](https://stimulus.hotwired.dev)
- SCSS Resmi Dokümantasyon (en): [sass-lang.com](https://sass-lang.com)
- Rails Frontend CLI Resmi Dokümantasyon (tr): [github.com](https://github.com)