

WEB PROGRAMLAMA I ■ HAFTA 1

Web'e Giriş ve Temel Kavramlar

İnternet, World Wide Web, HTTP/HTTPS, URL & DNS, İstemci–Sunucu Mimarisi,
Tarayıcıların Çalışma Prensipleri

Prof. Dr. Özal YILDIRIM

Fırat Üniversitesi — Teknoloji Fakültesi

Yazılım Mühendisliği Bölümü

Bu derste **sıfırdan başlayarak** modern bir web sayfasının nasıl tasarlandığını, kodlandığını ve canlandırıldığını öğreneceğiz.

- ▶ **HTML** — sayfanın iskeleti
- ▶ **CSS** — sayfanın görünümü
- ▶ **JavaScript** — sayfanın davranışı
- ▶ **PHP** — sunucu tarafına ilk adım



Dersin Felsefesi

Her hafta **teori + canlı kod + laboratuvar föyü** üçlüsü ile ilerleyeceğiz. Dönem sonunda kendi **BookVerse** kitap sitenizi sıfırdan inşa etmiş olacaksınız.

Hf.	Konu
1	Web'e Giriş ve Temel Kavramlar
2	HTML Temelleri ve Geliştirme Ortamı
3	HTML: Tablolar, Formlar, Multimedya
4	HTML5 Semantik Yapı, Erişilebilirlik, SEO
5	CSS'e Giriş: Sözdizimi ve Seçiciler
6	CSS Box Model
7	CSS Renkler, Tipografi, Arka Planlar

Hf.	Konu
8	Flexbox ile Modern Düzen
9	CSS Grid ve Duyarlı Tasarım
10	JavaScript Temelleri
11	Fonksiyonlar, Diziler, Nesneler
12	DOM Manipülasyonu ve Olaylar
13	Asenkron JavaScript, Fetch API, JSON
14	PHP ile Sunucu Tarafına Giriş

Kaynak Kitap: *Modern Web Geliştirme (HTML'den Buluta)* — Ünite 1–3

- ▶ İnternet ile World Wide Web arasındaki farkı açıklayabilecek,
- ▶ Web'in tarihsel gelişimini ve kırılma noktalarını sıralayabilecek,
- ▶ HTTP istek–cevap döngüsünü ve temel durum kodlarını yorumlayabilecek,
- ▶ HTTP ile HTTPS arasındaki güvenlik farkını açıklayabilecek,
- ▶ Bir URL'i parçalarına ayırabilecek ve DNS çözümlemesini anlatabilecek,
- ▶ İstemci–sunucu mimarisini ve rollerini tanımlayabilecek,
- ▶ Bir tarayıcının Kritik Görüntüleme Yolu adımlarını sıralayabilecek,
- ▶ Yapay zeka asistanlarını web öğrenme sürecinde etkin kullanabileceksiniz.

- 1 İnternet ve World Wide Web
- 2 Web'in Kısa Tarihi
- 3 HTTP ve HTTPS
- 4 URL, URI ve DNS
- 5 İstemci–Sunucu Mimarisi
- 6 Tarayıcılar Nasıl Çalışır?
- 7 Yapay Zeka ile Web Öğrenmek
- 8 Özet ve Değerlendirme



İnternet ve World Wide Web

BÖLÜM 1

Her Şey Görünmez Ağlar Üzerinde

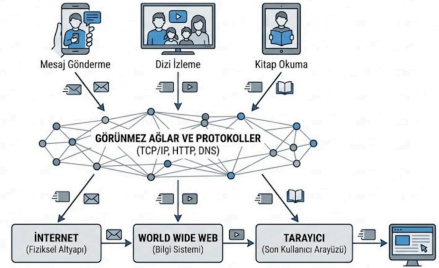
Günümüz dijital dünyasında attığımız her adım, görünmez ağlar ve protokoller üzerinde gerçekleşir.

- Bir mesaj gönderirken...
- Bir dizi izlerken...
- Bu slaytı okurken bile...

...on yıllar boyunca geliştirilmiş temel teknolojileri kullanıyoruz.

İpucu

Temelleri anlamak, üzerine inşa edeceğimiz yapıların **sağlamlığı** için kritik öneme sahiptir.



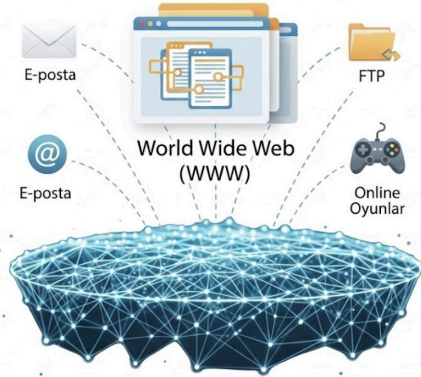
Kavram Karışıklığı: İnternet mi, Web mi?

İnternet

Birbirine bağlı milyarlarca cihazın oluşturduğu **küresel fiziksel ağ altyapısıdır**. Kablolar, yönlendiriciler, uydular, veri merkezleri...

World Wide Web (WWW)

İnternet üzerinde çalışan **servislerden yalnızca biridir**. Hipermetin bağlantılarıyla birbirine bağlanmış dokümanlar sistemidir.



İnternet altyapısı ve üzerinde çalışan servisler

Analoji: İnternet **otoyol ağıdır**; Web ise bu otoyollar üzerindeki **duraklardan biridir**.

İnternet Üzerinde Çalışan Diğer Servisler

Web, İnternet'in tek servisi değildir. Aynı altyapı üzerinde çalışan diğer servisler:

E-Posta

SMTP, POP3, IMAP protokolleri. Web'den tamamen bağımsız çalışır.

Dosya Transferi

FTP / SFTP. Sunucular arası dosya taşımak için kullanılır.

Uzak Erişim

SSH, Telnet. Sunucuyu komut satırından yönetmek için.

Gerçek Zamanlı

VoIP, WebRTC. Sesli/görüntülü görüşme trafiği.

Mobil Uygulamalar

Çoğu API üzerinden HTTP kullanır ama tarayıcı gerektirmez.

Zaman/İsim Servisleri

NTP (saat), DNS (isim çözümleme).

*Bu derste odak noktamız **Web** olacak; ancak alttaki katmanların varlığını unutmayacağız.*



Web'in Kısa Tarihi

BÖLÜM 2

CERN, 1989: Bir Problemin Çözümü

CERN, dünyanın dört bir yanından gelen binlerce araştırmacının **farklı ağlar ve işletim sistemleri** kullandığı devasa bir kurumdu.

Problem: Araştırma sonuçlarını, dokümanları ve verileri verimli biçimde paylaşmak neredeyse imkânsızdı.

Çözüm önerisi (Tim Berners-Lee): Bilgiyi **hipermetin (hypertext)** kavramıyla birbirine bağlayan, **merkezi olmayan küresel bir bilgi yönetim sistemi**.

İpucu

Herhangi bir kullanıcı, herhangi bir yerden, herhangi bir bilgiye kolayca erişebilmeliydi.



Sir Tim Berners-Lee
Web'in mucidi, W3C kurucusu

Temel Taşların İcadı (1989–1991)

Berners-Lee, bugünün Web'inin hâlâ temelini oluşturan **üç teknolojiyi** icat etti:

</> HTML

HyperText Markup Language

Web sayfalarını oluşturmak ve metin, resim gibi içerikleri **yapılandırmak** için kullanılan işaretleme dili.

🔗 URI / URL

Uniform Resource Identifier

Web üzerindeki her kaynağa atanan ve onu **benzersiz** kılan adresleme sistemi.

↔ HTTP

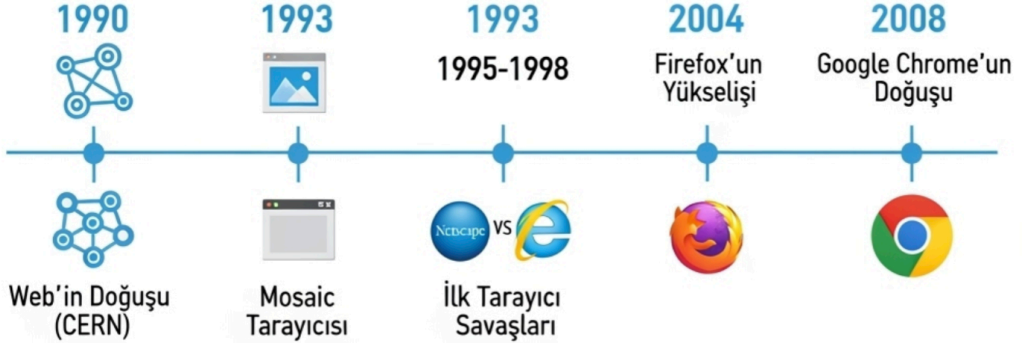
Hypertext Transfer Protocol

Tarayıcıların sunuculardan kaynak **istemesini ve almasını** sağlayan uygulama katmanı protokolü.

🚩 Tarihi An

1990 sonunda Berners-Lee ilk tarayıcıyı (aynı zamanda editör olan **“WorldWideWeb”**) ve ilk web sunucusunu (**“httpd”**) geliştirdi. **6 Ağustos 1991**'de dünyanın ilk web sitesi CERN'deki bir NeXT bilgisayarında yayına girdi.

Web'in Zaman Çizelgesi



1993 — Mosaic

NCSA'da Marc Andreessen ve ekibi tarafından geliştirildi. Devrimi: **metin ve resimleri aynı pencerede** gösterebilen ilk kullanıcı dostu tarayıcı olmasıydı.

1994 — Netscape Navigator

Mosaic'in başarısından doğdu. 1990'ların ortasında pazarın **%90'ına** hakimdi.

1995 — Internet Explorer

Microsoft, IE'yi Windows ile birlikte **ücretsiz** dağıtmaya başladı.

⚡ “İlk Tarayıcı Savaşları”

Bu rekabet iki yönlü sonuç doğurdu:

- + JavaScript ve CSS gibi teknolojiler hızla gelişti.
- Her iki şirket de **standart dışı** HTML etiketleri ekledi → geliştiriciler için uyumluluk kabusu.

Bu dönem, web geliştirme tarihinde “karanlık çağ” olarak anılır.

Standartlar Çağı ve Modern Tarayıcılar

Tarayıcı savaşlarının yarattığı kaos, **açık ve evrensel standartların** ne kadar gerekli olduğunu gösterdi.

W3C

World Wide Web Consortium

Tim Berners-Lee tarafından kuruldu. Web standartlarını belirleyen ve tüm tarayıcıların aynı kurallara uymasını sağlayan ana otoritedir.

2004 — Mozilla Firefox

Netscape'in kullerinden doğan açık kaynaklı tarayıcı, IE'nin hakimiyetini sarstı.

2008 — Google Chrome

Hız, basitlik ve güçlü geliştirici araçlarıyla oyunu değiştirdi. "İkinci Tarayıcı Savaşları" başladı.

✓ Bugün Neredeyiz?

Chrome, Firefox, Safari ve Edge **W3C standartlarına büyük ölçüde uyumludur**.
Bu sayede geliştiriciler:

- ▶ Her tarayıcı için ayrı kod yazmaktan kurtuldu,
- ▶ Daha **tutarlı, erişilebilir** ve **güçlü** uygulamalar geliştirebiliyor.



HTTP ve HTTPS

BÖLÜM 3

Tanım

HTTP (Hypertext Transfer Protocol — Hipermetin Aktarım Protokolü), bir istemcinin bir sunucudan **nasıl bilgi isteyeceğini** ve sunucunun bu isteğe **nasıl cevap vereceğini** tanımlayan kurallar bütünüdür.



`GET/index.html`
Headers: User-Agent,
Accept-Language...
HTTP İsteği (Request)

HTTP cevabı (Response)



İstek (Request) Anatomisi

- ▶ **Metot (Method)** — İsteğin amacı
GET (veri iste), POST (veri gönder)
- ▶ **Hedef (URL)** — İstenen kaynağın adresi
- ▶ **Başlıklar (Headers)** — Ek bilgiler
User-Agent, Accept-Language, Cookie
- ▶ **Gövde (Body)** — Gönderilen asıl veri
Genellikle POST isteklerinde bulunur

```
GET /kitaplar/12 HTTP/1.1
Host: www.bookverse.com
User-Agent: Mozilla/5.0 (Macintosh)
Accept: text/html,application/json
Accept-Language: tr-TR,tr;q=0.9
Cookie: oturum=a1b2c3d4
```

Ham bir HTTP GET isteği

İpucu

Bir web sayfasını görüntülemek en tipik GET isteğidir. Bir formu doldurup göndermek ise POST isteğine örnektir.

- ▶ **Durum Kodu (Status Code)**
Üç haneli sonuç bildirimi
- ▶ **Başlıklar (Headers)**
Content-Type, Content-Length
- ▶ **Gövde (Body)**
Asıl yanıt: HTML, resim veya JSON verisi

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 1547
Date: Mon, 06 Sep 2026 10:24:11 GMT

<!DOCTYPE html>
<html lang="tr"> ... </html>
```

Başarılı bir HTTP cevabı

✓ 2xx — Başarılı

200 OK — İstek başarıyla işlendi
201 Created — Kaynak oluşturuldu
204 No Content — Başarılı, içerik yok

🔍 3xx — Yönlendirme

301 Moved Permanently — Kalıcı taşınma
302 Found — Geçici yönlendirme
304 Not Modified — Önbellek geçerli

👤 4xx — İstemci Hatası

400 Bad Request — Hatalı istek
401 Unauthorized — Kimlik doğrulama gerekli
403 Forbidden — Erişim yasak
404 Not Found — Sayfa bulunamadı

🔧 5xx — Sunucu Hatası

500 Internal Server Error — Sunucu hatası
502 Bad Gateway — Geçit hatası
503 Service Unavailable — Servis kapalı

Hatırlatma: 4xx **sizin** hatanız, 5xx **sunucunun** hatasıdır.

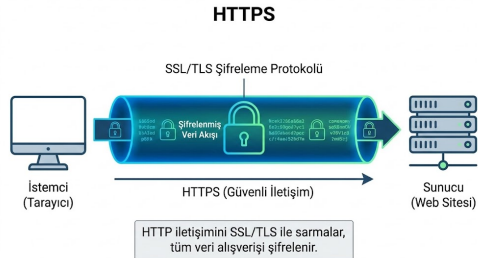
HTTPS: Güvenli İletişim

HTTP'nin zayıflığı: Gönderilen tüm veriler (kullanıcı adları, şifreler, form bilgileri) **düz metin** olarak, yani şifrelenmeden gider.

Çözüm: HTTPS, standart HTTP iletişimini **SSL/TLS** şifreleme protokolü ile **sarmalar**.

Neden Herkes için Şart?

- ▶ Kullanıcı gizliliği ve veri güvenliği
- ▶ Tarayıcılar HTTP siteleri “Güvenli Değil” diye işaretliyor
- ▶ Arama motorları HTTPS sitelere sıralamada öncelik veriyor



*HTTP ile gönderilen veri **herkesin okuyabileceği bir kartpostal**, HTTPS ile gönderilen veri ise **mühürlenmiş bir zarftır**.*



URL, URI ve DNS

BÖLÜM 4

URI

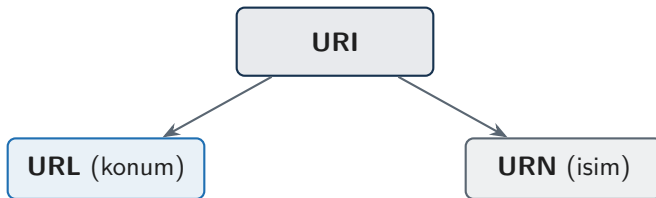
Uniform Resource Identifier

Bir kaynağı benzersiz şekilde tanımlayan **herhangi bir karakter dizisi**. Kaynağın kimliğidir.

URL

Uniform Resource Locator

Kaynağı tanımlamakla kalmaz, ona **nasıl ve nereden** erişileceğini de belirtir. Kaynağın adresidir.



*Web geliştirme dünyasında neredeyse her zaman **URL**'er ile çalışırız.*

Bir URL'in Anatomisi

https://www.ozalyildirim/q=web+teknolojileri#bolum-3



`https://www.site.com:443/arama/gorsel?q =web&kaynak =kitap#bolum-3`

Parça	Örnek	Açıklama
Protokol (Scheme)	https://	Kaynağa erişim protokolü. http://, ftp://, mailto: da olabilir.
Domain Adı	www.site.com	Sunucunun <i>insan tarafından okunabilir</i> adresi.
Port	:443	Sunucudaki iletişim “kapısı”. HTTP =80, HTTPS =443. Varsayılsa yazılmaz.
Yol (Path)	/arama/gorsel	Sunucu üzerindeki kaynağın dosya yolu.
Sorgu (Query)	?q =web&kaynak =kitap	? ile başlar, anahtar =değer çiftleri & ile ayrılır.
Parça (Fragment)	#bolum-3	Sayfa içi konum. Sunucuya gönderilmez , sadece tarayıcı kullanır.

DNS: İnternet'in Telefon Rehberi

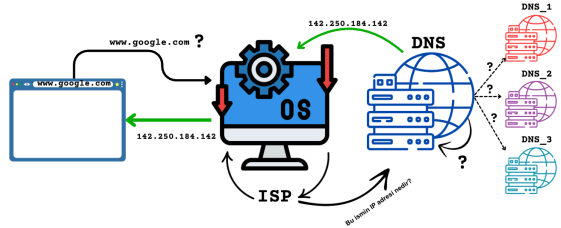
Bilgisayarlar `www.google.com` gibi isimlerle değil, **IP adresleri** ile iletişim kurar:

142.250.184.142

DNS

Domain Name System

Alan adlarını IP adreslerine çeviren, İnternet'in devasa **dağıtık telefon rehberi**.



DNS Çözümlemesi Adım Adım

- 1 Tarayıcınız, domain adını önce **kendi önbelleğine**, sonra işletim sisteminize sorar.
- 2 İşletim sistemi, ISS tarafından atanan bir **DNS Sunucusuna** “Bu ismin IP adresi nedir?” diye sorar.
- 3 DNS sunucusu kendi kayıtlarına bakar; bulamazsa diğer DNS sunucularına (Root → TLD → Authoritative) sorar.
- 4 Bulunan IP adresi tarayıcıya geri gönderilir.
- 5 Tarayıcı, artık gerçek adresi bildiği için **HTTP isteğini** bu IP adresine gönderebilir.

💡 İpucu

Bu işlem genellikle **milisaniyeler içinde** gerçekleşir ve kullanıcı tarafından fark edilmez. Ancak DNS olmadan, bugün bildiğimiz anlamda kolay ve erişilebilir bir Web olmazdı.

Uygulamalı: DNS'i Kendiniz Sorgulayın

Terminali (macOS/Linux) veya Komut İstemini (Windows) açıp deneyin:

```
# Bir alan adının IP adresini öğrenme
nslookup www.firat.edu.tr

# Daha detayli sorgulama (macOS / Linux)
dig www.firat.edu.tr +short

# Sunucuya paket gönderip yanıt süresini ölçme
ping www.firat.edu.tr

# Paketin izlediği rotayı görme
tracert www.firat.edu.tr      # Windows: tracert
```

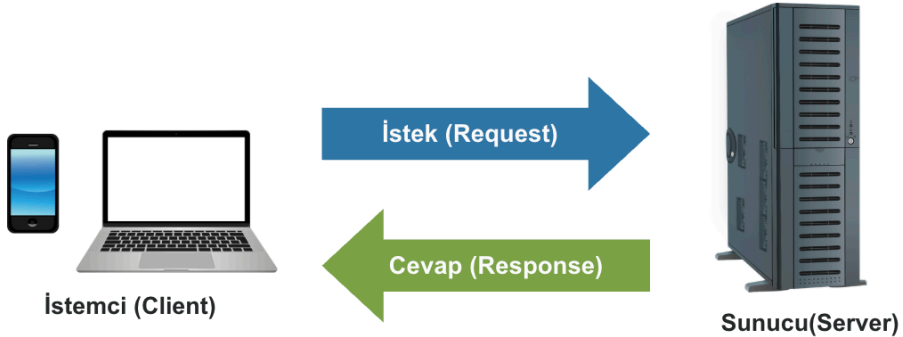
İpucu

Aynı alan adını farklı zamanlarda sorguladığınızda **farklı IP adresleri** dönebilir. Büyük siteler yük dengeleme (load balancing) için birden fazla sunucu kullanır.



İstemci–Sunucu Mimarisi

BÖLÜM 5



Tüm modern web, görevleri **iki net role** ayıran bu basit ama güçlü model üzerine kuruludur.

İstemci (Client)

İstemci, bir hizmet veya kaynak için **istekte bulunan** taratır. **Temel görevleri:**

- 1 Kullanıcıdan girdi alır (link tıklama, adres yazma, form doldurma)
- 2 Bu girdiyi bir **HTTP isteğine** dönüştürür
- 3 İsteği, DNS ile bulduğu IP adresine gönderir
- 4 Sunucudan gelen cevabı bekler
- 5 Gelen HTML/CSS/JS'i işler ve **ekrana çizer (render)**



İstemci Türleri

- ▶ Web tarayıcısı (en yaygın)
- ▶ Mobil uygulama
- ▶ Masaüstü yazılımı
- ▶ Başka bir sunucu(!)
- ▶ curl, Postman gibi araçlar

İpucu

Analoji: İstemci, restorana giden, menüye bakıp sipariş veren **müşteridir**.

Sunucu, isteklere **hizmet veya kaynak sağlayan** taratır. **Temel görevleri:**

- 1 Belirli bir portu (örn. 443) sürekli **dinler**
- 2 Gelen isteği **analiz eder**
- 3 İsteği **işler**: dosya okur, veritabanından veri çeker, program çalıştırır
- 4 Bir HTTP **cevabı oluşturur**
- 5 Cevabı istemcinin IP adresine **geri gönderir**

☰ “Sunucu” İki Anlamlı

1. **Donanım:** Dosyaları ve veritabanını barındıran fiziksel bilgisayar.
2. **Yazılım:** Üzerinde çalışan ve istekleri dinleyen program:
 - Apache HTTP Server
 - Nginx
 - Microsoft IIS

💡 İpucu

Analoji: Sunucu, siparişi alan, mutfakta hazırlayan ve geri gönderen **restoranın kendisidir**.

Frontend (İstemci Tarafı)

Tarayıcıda çalışan, kullanıcının **gördüğü ve dokunduğu** her şey. **Teknolojiler:** HTML, CSS,

JavaScript

Sorumluluk: Arayüz, etkileşim, görsel deneyim

Bu dönem: **Hafta 2–13**

Backend (Sunucu Tarafı)

Sunucuda çalışan, **iş mantığını ve veriyi** yöneten kısım. **Teknolojiler:** PHP, Node.js,

Python, C#

Sorumluluk: Veri, güvenlik, iş kuralları

Bu dönem: **Hafta 14 (giriş)** — devamı Web-2'de

Bir geliştirici olarak genellikle bu iki taraftan **biri** veya **her ikisi** (*full-stack*) için kod yazarız.

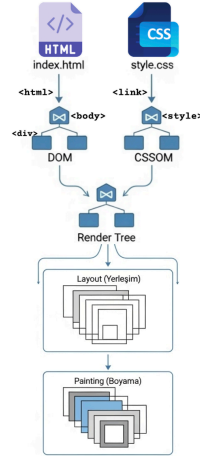
Tarayıcılar Nasıl Çalışır?

BÖLÜM 6

Koddan Piksele: Kritik Görüntüleme Yolu

Sunucudan gelen HTML, CSS ve JS dosyaları, ekranda piksellere dönüşene kadar bir dizi adımdan geçer. Bu adımlar bütününe **Kritik Görüntüleme Yolu (Critical Rendering Path)** denir.

- 1 HTML ayrıştırılır → **DOM Ağacı**
- 2 CSS ayrıştırılır → **CSSOM Ağacı**
- 3 İkisi birleşir → **Render Ağacı**
- 4 Konum ve boyut hesabı → **Layout / Reflow**
- 5 Piksellerin çizimi → **Painting**



1–2. Adım: DOM ve CSSOM Ağaçları

DOM

Document Object Model

Tarayıcı, HTML dosyasını yukarıdan aşağıya okur ve etiketleri (`<html>`, `<body>`, `<div>`) anlayarak sayfanın **yapısını temsil eden ağacı** oluşturur.

CSSOM

CSS Object Model

`<link>` ile çağrılan veya `<style>` içindeki CSS kuralları ayrıştırılır; hangi elementin **hangi stile** sahip olduğunu belirten ağaç oluşur.

Dikkat

CSS, render'ı **bloklayan** bir kaynaktır (*render-blocking*). Tarayıcı, CSSOM tamamlanmadan Render Ağacını kuramaz. Bu yüzden CSS dosyaları mümkün olduğunca **küçük** ve **`<head>` içinde** olmalıdır.

3-5. Adım: Render Ağacı, Layout, Painting

3. Render Ağacı

DOM + CSSOM birleşir. Sadece **görsel olarak görünecek** elementler dahil edilir. ❌

display:none olan element DOM'da vardır ama Render Ağacında **yoktur**.

4. Layout

Her elementin ekranda **tam olarak nerede** duracağı ve **ne kadar yer** kaplayacağı hesaplanır. *Reflow* olarak da anılır. Pahalı bir işlemdir.

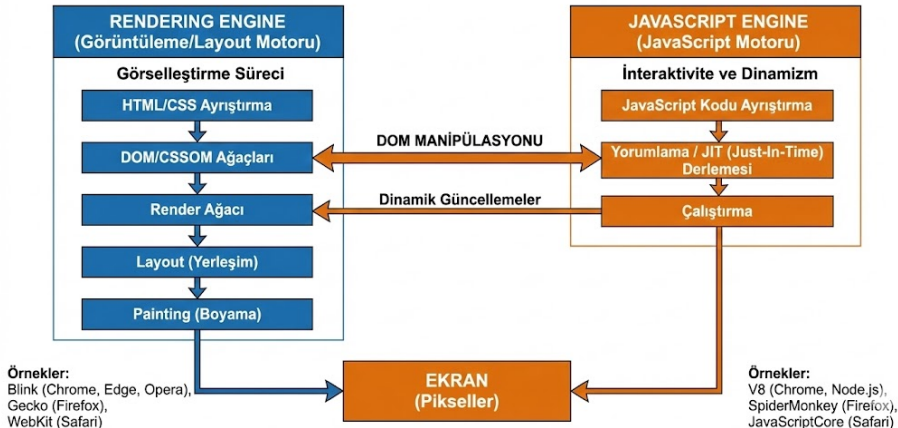
5. Painting

Hesaplanan kutuların içi; renkler, kenarlıklar ve gölgelerle **boyanır**. Kullanıcının sayfaı nihayet **gördüğü** andır.

İpucu

JavaScript ile bir elementin genişliğini değiştirdiğinizde **Layout + Painting** yeniden çalışır. Sadece rengini değiştirdiğinizde ise **yalnızca Painting**. Bu fark, performans optimizasyonunun temelidir.

WEB SAYFASI OLUŞUM SÜRECİNDE TARAYICI MOTORLARI (Rendering & JavaScript)



🔧 Rendering Engine

HTML/CSS ayrıştırma, DOM/CSSOM/Render ağaçları, Layout ve Painting'den sorumlu **çekirdek bileşen**.

- ▶ **Blink** — Google (Chrome, Edge, Opera)
- ▶ **Gecko** — Mozilla (Firefox)
- ▶ **WebKit** — Apple (Safari)

Blink, WebKit'in bir dalıdır (fork).

⚡ JavaScript Engine

JavaScript kodunu **ayrıştırır, yorumlar ve çalıştırır**. Sayfaya **interaktivite** katar.

- ▶ **V8** — Google (Chrome, Node.js)
- ▶ **SpiderMonkey** — Mozilla (Firefox)
- ▶ **JavaScriptCore** — Apple (Safari)

*Modern motorlar **JIT derlemesi** kullanır.*

JavaScript motorunun en önemli yeteneği **DOM'u manipüle etmektir**: buton tıklanınca yeni element ekleyebilir, stil değiştirebilir, sunucudan veri çekebilir.

Geliştirici Araçlarını (DevTools) Tanıyalım

Tarayıcınızda **F12** (veya macOS'ta Cmd+Option+I) tuşuna basın:

- ▶ **Elements** — Canlı DOM ağacını görün ve düzenleyin
- ▶ **Console** — JavaScript komutlarını anında çalıştırın
- ▶ **Network** — Tüm HTTP istek/cevaplarını, durum kodlarını ve sürelerini izleyin
- ▶ **Sources** — Kaynak dosyalar ve hata ayıklama (debug)
- ▶ **Application** — Çerezler, LocalStorage
- ▶ **Lighthouse** — Performans ve erişilebilirlik denetimi

💡 İpucu

Bu dönemin en önemli aracı DevTools olacak. **Network** sekmesini açıp bir siteyi yenileyin: bu haftaki tüm kavramları (istek, cevap, durum kodu, başlıklar) canlı olarak göreceksiniz.

Yapay Zeka ile Web Öğrenmek

BÖLÜM 7

Yeni Bir Ders Arkadaşı: Yapay Zeka Asistanları

ChatGPT, Claude, Gemini ve GitHub Copilot gibi araçlar artık web geliştirmenin ayrılmaz parçası. Bu derste onları **yasaklamayacağız** — **doğru kullanmayı öğreneceğiz**.

✓ Doğru Kullanım

- ▶ Anlamadığınız kavramı **açıklatmak**
- ▶ Kendi kodunuzdaki **hatayı buldurmak**
- ▶ Bir konsepti **analoji** ile anlattırmak
- ▶ Kendinize **alıştırma üretmek**
- ▶ Yazdığınız kodu **eleştirtmek**

✗ Yanlış Kullanım

- ▶ Ödevi **anlamadan** kopyalamak
- ▶ Üretilen kodu **test etmeden** teslim etmek
- ▶ Hata mesajını okumadan yapıştırmak
- ▶ Kaynak doğrulamadan **“doğrudur”** demek

*Yapay zeka bir **kaldıraçtır**, bir **koltuk değneği** değil. Kaldıraç, ancak sağlam bir zemine basıyorsanız işe yarar.*

- 1 **Rol** — “Sen bir web geliştirme eğitmenisin.”
- 2 **Bağlam** — “1. sınıf öğrencisiyim, HTTP'yi yeni öğreniyorum.”
- 3 **Görev** — “... kavramını açıkla.”
- 4 **Kısıt** — “En fazla 200 kelime, kod örneği ver, Türkçe.”
- 5 **Format** — “Madde madde, sonunda 3 soruluk mini test.”

Q Yapay Zeka Çıktısını Doğrulama

Prompt ne kadar **belirsiz** ise, cevap o kadar **genel** ve **işe yaramaz** olur. ✗ “HTTP nedir?”

✓ “Bir web dersi öğrencisine HTTP istek-cevap döngüsünü posta analogisiyle, 5 maddede, Türkçe anlat.”

Yapay Zeka İstemi (Prompt) — Kavramı Analoji ile Açıklat

Sen deneyimli bir web programlama egitmenisin. Üniversite 1. sınıf öğrencisiyim ve HTTP protokolünü yeni öğreniyorum.

HTTP istek-cevap döngüsünü, "postane ve mektup" analojisini kullanarak Türkçe anlat.

Anlatımda şu bölümleri kullan: 1) Analoji ile giriş 2) İstek bileşenleri 3) Cevap bileşenleri 4) 3 maddelik özet 5) Kendimi test edeceğim 3 soru

Teknik terimlerin İngilizce karşılığını parantez içinde ver. Toplam 350 kelimeyi geçme.

Yapay Zeka Çıktısını Doğrulama

Aldığınız cevabı **ders slaytlarıyla karşılaştırın**. Farklılık varsa: hangisi doğru? MDN Web Docs (developer.mozilla.org) üzerinden **üçüncü bir kaynakla** teyit edin.

Yapay Zeka İstemi (Prompt) — Kendine Quiz Ürettir

Asagidaki konulardan 10 soruluk coktan secmeli bir quiz hazirla: - Internet ve WWW farki - HTTP metotlari (GET/POST) ve durum kodlari (2xx/3xx/4xx/5xx) - URL bilesenleri (scheme, domain, port, path, query, fragment) - DNS cozumleme adimlari - Istemci-sunucu mimarisi - Kritik Goruntuleme Yolu (DOM, CSSOM, Render Tree, Layout, Paint)
Kurallar: Her sorunun 4 secenegi olsun. Once SADECE sorulari yaz, cevap anahtarini en sona ayri bir baslik altinda koy. Zorluk seviyesi "universite vize sinavi" duzeyinde olsun. Turkce yaz.

Bu prompt'u her hafta konu başlıklarını değiştirerek yeniden kullanabilirsiniz.

Yapay Zeka İstemi (Prompt) — Gerçek Bir Siteyi Çözümlet

Tarayıcimda DevTools > Network sekmesini actim ve `www.firat.edu.tr` sitesini yeniledim.

Aşğıdaki başlıkları gördüm:

Status: 200 | Content-Type: text/html; charset =UTF-8 Server: nginx | Cache-Control: max-age =3600

Bu bilgilerin her birinin ne anlama geldiğini, bir öğrenciye anlatır gibi tek tek açıkla. Ayrıca bu sitenin hangi web sunucu yazılımını kullandığını ve "max-age =3600" değerinin kullanıcı deneyimine etkisini belirt.

Yapay Zeka Çıktısını Doğrulama

Yapay zeka **sizin verdiğiniz veriyi** yorumlarken güçlüdür. Ancak “şu sitenin sunucusu nedir?” diye **doğrudan** sorarsanız **uydurabilir** (halisinyon). **Veriyi siz verin, o yorumlasın.**

☰ Her Zaman Sorun

- 1 Bu bilgi **ders kitabıyla** örtüşüyor mu?
- 2 Verilen kod **gerçekten çalışıyor** mu? (Denedim mi?)
- 3 Bahsedilen etiket/fonksiyon **gerçekten var** mı? (MDN'de aratın)
- 4 Bilgi **güncel** mi? Yoksa 10 yıl önceki bir yöntem mi?
- 5 Bunu bir arkadaşıma **açıklayabilir** miyim?



Halüsinasyon Nedir?

Yapay zeka modelleri, emin olmadıklarında bile **kendinden emin bir tonda yanlış bilgi** üretebilir. Özellikle:

- ▶ Var olmayan HTML etiketleri
- ▶ Var olmayan CSS özellikleri
- ▶ Yanlış kütüphane fonksiyonları
- ▶ Uydurma kaynak/URL

Kural: Kodu çalıştırmadan “çalışıyor” demeyin.

Akademik dürüstlük: Ödevlerinizde yapay zeka kullandıysanız, **hangi prompt'u** kullandığınızı ve **neyi değiştirdiğinizi** belirtin.

Özet ve Değerlendirme

BÖLÜM 8

Bölüm Özeti: Bir İsteğin Yolculuğu

- 1 **Adresleme ile başlar:** Kullanıcı, tarayıcıya insan dostu bir **URL** yazar.
- 2 **Çeviri ile yol bulunur:** **DNS**, alan adını makinelerin anlayacağı **IP adresine** çevirir.
- 3 **İletişim kuralları belirlenir:** **İstemci–Sunucu** mimarisi devreye girer; tarayıcı, **HTTP/S** ile bir **istek** gönderir.
- 4 **Paketler teslim edilir:** Sunucu isteği işler ve HTML, CSS, JS dosyalarını içeren bir **cevap** döner.
- 5 **Sanat eseri oluşturulur:** **Rendering Engine** DOM + CSSOM → Render Ağacı → Layout → Painting adımlarını işletir. **JavaScript Engine** ise bu statik yapıya hayat verir.

Terim	Anlam	Terim	Anlam
HTTP	Hipertextin aktarım protokolü	DOM	Belge nesne modeli
HTTPS	SSL/TLS ile şifrelenmiş HTTP	CSSOM	CSS nesne modeli
URI	Kaynak tanımlayıcısı (kimlik)	Layout	Konum/boyut hesaplama
URL	Kaynak konumlayıcısı (adres)	Painting	Piksellerin çizimi
DNS	Alan adı → IP çevirisi	Blink	Chrome render motoru
IP	Ağdaki cihazın sayısal adresi	V8	Chrome JS motoru
W3C	Web standartları otoritesi	JIT	Anında derleme
Frontend	İstemci tarafı geliştirme	Backend	Sunucu tarafı geliştirme

Değerlendirme Soruları (1/2)

- 1 İnternet ile World Wide Web arasındaki temel fark nedir? İnternet üzerinde çalışan Web dışında üç servis örneği veriniz.
- 2 Bir kullanıcı tarayıcısına `https://www.ornek.com/urunler?id=5` yazıp Enter'a bastı. Bu URL'in **altı bileşenini** tek tek belirleyiniz.
- 3 404 ve 500 durum kodları arasındaki fark nedir? Hangi durumda kim sorumludur?
- 4 HTTP ile HTTPS arasındaki farkı “kartpostal–zarf” analogisini kullanmadan, teknik olarak açıklayınız.

- 5 DNS çözümlemesi olmasaydı Web'i kullanmak nasıl bir deneyim olurdu? En az iki somut zorluk yazınız.
- 6 Kritik Görüntüleme Yolu'nun beş adımını sırasıyla yazınız. `display:none` tanımlı bir element bu adımların hangisinde elenir?
- 7 Rendering Engine ile JavaScript Engine'in görevlerini karşılaştırınız. Chrome bu iki görev için hangi motorları kullanır?
- 8 **Uygulama:** Bir web sitesini DevTools'un **Network** sekmesiyle inceleyip, ilk HTML isteğinin **durum kodunu**, **Content-Type** başlığını ve **yüklenme süresini** raporlayınız.

Laboratuvar Föyü 1

- ▶ DevTools ile HTTP trafiği incelemesi
- ▶ nslookup / ping / traceroute uygulamaları
- ▶ URL bileşen çözümlemesi
- ▶ Yapay zeka destekli kavram doğrulama görevi

Teslim: Gelecek dersin başlangıcında.

Hazırlık

Gelecek hafta kod yazmaya başlıyoruz. Önceden kurunuz:

- ▶ **Visual Studio Code**
`code.visualstudio.com`
- ▶ **Live Server** eklentisi
- ▶ **Google Chrome** (güncel sürüm)

Okuma: Kitap Ünite 2.1 (HTML)

Kaynaklar: developer.mozilla.org (MDN Web Docs) ▪ [w3.org](https://www.w3.org) (W3C) ▪ caniuse.com

Haftaya Görüşmek Üzere!

Gelecek Hafta:

HTML Temelleri ve Geliştirme Ortamı

VS Code kurulumu, Live Server, HTML doküman yapısı, temel etiketler