# Giriş

Merhaba ben İsa Sarı. Patika.dev ve Protein tarafından organize edilen Vue.js Bootcamp'ine teknik test ve mülakat sürecinden sonra kabul aldım. Bootcampi düzenleyen Patika.dev ve Protein ekibine teşekkür ederim. Bu doküman serisinde, bu bootcamp boyunca derslerden aldığım notlarımı paylaşacağım. Bu dokümanda bootcampte aldığım notların yanı sıra kendi yaptığım eklemeler de yer alacak. Dokümanda karşılaşılabilecek yanlışlıklar veya eksiklikler tamamen bana aittir. Geribildirim için iletişim kurmaktan çekinmeyin. Mail adresim: dev.isasari@yandex.com. Eğitmenimiz Tolga Eğilmezel. Bootcamp 7 hafta sürecek. Bootcamp'te işlenecek ana konu başlıkları şunlardır:

1. Hafta
   1.1. Front-End Developer ve Front-End Development
   1.2. JavaScript Temel Bilgiler

2. Hafta
   2.1. Vue.js Nedir?
   2.2. Component Nedir?
   2.3. CSS Stiller
   2.4. Props

3. Hafta
   3.1. Handling Events
   3.2. If Else
   3.3. Döngüler
   3.4. Reactivity

4. Hafta
   4.1. Computed Properties
   4.2. Forms Events

5. Hafta
   5.1. Teleport
   5.2. Reactive
   5.3. Component Lifecycle Hooks

6. Hafta
   6.1. Vuex
   6.2. Pinia
   6.3. Vue Router
   6.4. Firebase

7. Hafta
   7.1. Nuxt.js

Bootcamp süresince derslerin işlenişi, ödevler, geribildirimler, duyurular ve iletişim için Eduflow, Github Classrom, Google Dokümanlar, Discord, Telegram, Google Meet ve daha birçok aracı kullanacağız.

# 1. Hafta

## İçindekiler

## 1. Front-End Developer kimdir?

Tasarlanan projeyi, hedef kitlenin kullanabileceği şekilde etkileşimli bir hale getiren kişiye Front-End Developer denir.

Front-End Developer, etkileşim tasarımı yapar. Örneğin bir menüye tıklanıldığı zaman o menünün görülmesi, bir form gönderildiyse o formun gönderildiğini bildirmek, bir ürünü sepete ekle kısmına basıldığında o ürünü sepete eklemek gibi etkileşimlerin tasarımından Front-End Developer sorumludur.

"Front-end development vazgeçilmezdir çünkü harika ürünler bile kötü kullanıcı arayüzleri yüzünden hayal kırıklığına uğrayabilir. Yazılım ürünlerinin ve web sitelerinin kullanıcı arayüzünü kullanılabilir ve çekici hale getirmek Front-End Developer'ın işidir. Front-End Developer, modern işletmeler için çok önemlidir. Çekici ve işlevsel bir web sitesi olmadan, potansiyel müşteriler başka bir yere bakacaktır. Dahası, kötü sunulan ve kullanımı zor bir uygulama, mevcut müşterilerin daha az işlevsellik sunsa bile daha ilgi çekici ise rakibin uygulamasını kullanmalarına neden olacaktır. Bunun dışında, iyi sunulan bir web sitesi veya uygulama güven yaratacaktır." (Bu paragraf, Dominic Myers'in Front-End Developer isimli kitabından alıntıdır).

Front-End Developer olmak için detaylı bir yol haritası şurada yer almakta: https://roadmap.sh/frontend

Patika.dev'den ise Front-End Development patikalarını takip ederek kendinizi geliştirebilirsiniz: https://app.patika.dev/paths

## 2. Piksel Perfect nedir?

Bir tasarımın birebir aynısını yapmaya piksel perfect denir.

## 3. Front-End Developer'ın öne çıkan kısmı nedir? Herkes web sitesi kurabiliyorsa (!) Front-End Developer'a neden ihtiyaç duyarız?

Front-End Developer projeye özel çözüm üretir, projenin ihtiyacını karşılayacak şeyleri geliştirir.

## 4. Google'lamak Konusu

Google'da arama yapıp doğru kaynaklara, dokümantasyonlara ulaşıp çözümler bulmak ayrı bir yetkinliktir. İyi bir Google'lama yeteneğiyle istediğiniz bilgiyi doğru kaynaklardan güncel versiyonlarını, daha kolay versiyonlarını, daha stabil versiyonlarını kolayca bulabilirsiniz. Bu

zamanla gelişen bir yetkinliktir. O yüzden Google'da arama yapmayı çok basit bir şey gibi düşünmeyin. Çünkü doğru kaynaklara ulaşmanız çok önemli.

## 5. Front-End Developer'ın iletişiminin neden kuvvetli olması gerekiyor?

Front-End Developer birçok departmanla çalışıyor.

Responsive websiteleri yapıyoruz, yani bir ekran birden fazla çözünürlükte çalışabiliyor. Bu noktada tasarımcı bize bir buton verdiğinde, bu butonu alıp koymak doğru ancak bu tablete geldiği zaman, mobile geldiği zaman nasıl bir aksiyonu olacak, kullanıcılar bununla etkileşime geçebilecek mi, tıklayabilecek mi gibi gibi konuları öngörmek ve bunu bildirmek önemli.

API'dan gelen verilerle kullanıcıya doğru mesajı göstermek gerekiyor. Örneğin, sepete tıkla butonuna tıkladınız hiçbir şey olmadı, neden? Belki işte o anda arkaplanda sistemsel bir sorun oldu, belki stok bitti, x oldu y oldu hiç fark etmez, Back-End Developer'la iletişime geçip "kullanıcı bir ürünü sepete ekleyemediği zaman kullanıcıya x kodunu dön ki ben de kullanıcıya bunun olmadığı bilgisini vereyim ya da anlık olarak bir sorun olduğunu kullanıcıya bir popup vs gibi bir şeyle bildireyim ki kullanıcı orada sadece boş boş tıklıyor olmasın" demek aslında çok önemli.

Front-End Developer merkez bir yerde. Tasarımcıyla, Back-End Developer'la birlikte çalışıyor. Bir taraf dokümante ediyor bir taraftan veriler akıyor, Front-End Developer'ın bunları kullanıcıya sorunsuz bir şekilde ulaştırması gerekiyor.

## 6. İngilizce neden önemli?

Kullandığımız dil ingilizce zaten çünkü HTML, CSS ve JavaScript'te İngilizce terimler kullanıyoruz. Bunların dokümantasyonları, kaynakları ve orijinalleri İngilizce. İngilizcenizi ne kadar geliştirirseniz yeni bir dil öğrenmeniz yeni bir framework öğrenmeniz o kadar kolay olur.

## 7. Büyük Bölünme Konusu

Eskiden yaygın olarak kullanılan sadece bir tane tarayıcı vardı o da Internet Explorer'dı (1995). Daha sonra yaygın olarak kullanılmaya başlanacak olan Firefox (2002) hayatımıza girdi. Firefox birtakım özellikler getirdi. Bu özellikler Internet Explorer'ın desteklemediği özelliklerdi. Örneğin, bir pencereye dropshadow koymak ya da bazı JavaScript fonksiyonlarını desteklemek. Bunları Internet Explorer yapmadığı için herkes bir anda Firefox'la geliştirme yapmaya başladı ama Internet Explorer kullanıcısı da vardı. Bu ikisi için web sitesi geliştirmek isterseniz iki kere kod yazmanız gerekiyordu. Zaman içinde bir de yaygın olarak kullanılmaya başlanacak olan Chrome (2008) çıktı ortaya. Böylece yaygın olarak kullanılan üç tane büyük tarayıcı olmuş oldu. Günümüze geldiğimizde ise tarayıcıların sayıları gittikçe artmış durumda.

Tarayıcı farklılığından kaynaklanan iş yükü çok arttığı için "front-end" diye bir dal yapalım arayüzle uğraşsın dediler bu kişiler.

Daha sonra işler biraz daha karışmaya başladı. Mobile geldi. Cep telefonları patladı. Cep telefonları popüler oldukça insanlar mobille daha çok site ziyaret etmeye başladı desktopa kıyasla. Bunun da getirisi şu oldu: Back-End Developer kendi özelinde ayrılıp database, veritabanları, sunucu işlemleri konusunda uzman olmasını getirdi. Böylelikle ayrım olmuş oldu.

Böylece Back-End Developer, siteye gelen yoğun ziyaretçi akınını handle edicek, veri akışı sağlayacak, sunuculardaki performanslı algoritmaları yazacak kesimi oluşturdu.

Front-End Developer da, bu kadar kullanıcının her ortamdan yani hem mobile'den hem masaüstünden, hem tabletlerden başka cihazlardan kolaylıkla siteyi gezmesi için uzmanlaştığı bir dal olmuş oldu.

Dünya nüfusunun yarısının elinde akıllı telefon var. Bu durum responsive tasarımı oldukça önemli bir hale getirdi. Sadece tarayıcı farklılığı desteği değil, bir ekranın her cihazda stabil olmasını sağlamak ekstra ekstra bir güç haline geldi. İphoneların ekran çözünürlüğü farklı androidlerin binlerce ekran çözünürlüğü var, tabletler masaüstleri var. O yüzden bu responsive çok önemli bir hale geldi zaman içerisinde.

## 8. Akıllı telefon, front-end tarafında responsive tasarım dışında başka neyi zorlaştırmış olabilir?

Yazdığımız JavaScript kodları, tarayıcı tarafından derleniyor. Bundan dolayı, telefonun özelliği ne kadar düşük olursa (işlemcisi, RAM'i vs.) sizin yazdığınız kodların çıktısını alması o kadar uzun sürüyor, yani ekranda kodların görülmesi. İşte bu noktada işlemciler de ramlerde bizim işimizin içine girmiş oluyor. Yazdığımız kodları optimize yazmanın önemi burada ortaya çıkıyor. Yani ne kadar kolay derlenebilir kod yazarsak web sitesi o kadar hızlı çalışıyor.

Ayrıca, mobilede ve tablette fareyi ve klavyeyi değil parmağımızı kullanıyoruz. Bundan dolayı parmak özelinde değişen tasarımlar da var. Örneğin çoğu sitede menü yoktur, hamburger menü vardır. Neden aslında mobilde kullanımı daha kolaydır. Yani sadece tasarım değişmiyor, yapısal değişiklik de var. O menü kayboluyor bambaşka bir menü geliyor. O buton yok oluyor bambaşka bir yapıda sticky (yapışık) bir form geliyor.

Bunlardan dolayı, front-end'in işyükü inanılmaz arttı. Çünkü tasarımı ekranda stabil bir şekilde çalıştırmak gerekiyor, bir yandan da çok kod yazmamak ya da optimize kod yazmak lazım ki her telefonda site açılabilsin ve performanslı çalışsın.

Biz şu anda Vue yazıcaz, Vue JavaScript demek zaten. Dolayısıyla Vue dediğimiz şey tarayıcıda çalışan bir şey. Eski bir bilgisayar kullanıyorsanız sizde site çok yavaştır, yeni bilgisayar kullanıyorsanız site sizde çok hızlıdır. Bunun Database'den veri gelmesi ya da tasarımın karmaşık olmasıyla alakası yok. Tamamen yazdığınız kodların tarayıcı üzerinde derlenmesiyle alakalı. Bu çok önemli bir kriter. Özellikle bu frameworklerle çalışıyorsanız.

Örneğin, Amazon'da yüzde birlik bir yavaşlama ki bu 100 milisaniyelik bir yavaşlamaya denk geliyor; 1.6 milyar dolar zarara mal olabiliyor. Çünkü insanlar yavaş bir site yerine hızlı bir siteye gitmeyi tercih ediyorlar. Bundan dolayı, Front-End'de sorumluluk iyice artıyor.

## 9. Animasyon Konusu

Etkileşimi artırmak için sitede animasyonlar kullanılması gerekebilir. Awwwards, ödül veren bir site. Bu tarz görsellerin, animasyonların olduğu görselliğin önplana çıktığı dolayısıyla yine front-end'e büyük iş düşüyor. Bu da bambaşka bir uzmanlık getiriyor front-end'e. Animasyon yaparken "webGL" denilen teknolojiyi öğrenmek gerekiyor. animasyon alanında ilerlerken 3d artistlerle iletişimde olmak gerekiyor vs. ayrı bir front-end alanı.

## 10. JavaScript Konusu

Eskiden HTML ve CSS yapıyorduk. Ufak etkileşimler yapıyorduk. Ama zaman içerisinde JavaScript'in özellikleri ve yeni özellikleriyle birlikte ve tarayıcıların da desteklemesiyle birlikte JavaScript çok güçlü bir hale geldi. Peki nasıl çok güçlü bir hale geldi? Bunun sebebi web2.0 yani sosyal medya. Sosyal medyanın bu anlık feedleri, yani örneğin yeni twit attığınız zaman anlık gelmesi, instagramdaki likeların hemen yansıması gibi konular aslında tamamen JavaScript'le yapılmaya başlandı. Ve back-end tarafında da API'lerin gelişmesine neden oldu. Böylelikle, herkes bir API yapısı, bir JavaScript kullanma bir framework kullanma gibi bir yönelişe gitti. Burada mantık tamamen bir websitesini görünür yapmak değil, arka tarafta logic'in çalışması. Yani bir kullanıcı like'a bastıktan sonra sadece like gibi gözükebilir ama ne kadar önemli biliyorsunuz instagramda. Bu like'a tıkladıktan sonra bunun database'e kaydedilmesi ve herkeste bu etkileşimin gözükmesi gibi konulara odaklanmaya başladı. JavaScript frameworkleri de bu noktada işimizi kolaylaştırdı. Çünkü sadece JavaScriptle yazılacak kod miktarı arttıkça kişi sayısı artacağı için bu frameworkler aslında bizleri destekliyor. Biraz daha az kod yazmamızı ve onların yapılarını kullanarak bu sistemleri inşa etmemizi yardımcı oluyorlar. Ama hepsinin temeli JavaScript. Bugün işte React'ın, Vue'nun, Angular'ın vs.nin github repolarına baktığınız zaman girdiğiniz zaman binlerce milyonlarca satır kod görürsünüz, bunları her projede tekrar yazmak tabi ki yorucu olduğu için ve artık JavaScript de çok güçlü olduğu için herkes JavaScript frameworklerini kullanmayı tercih ediyor.

## 11. NodeJS Konusu

JavaScript dediğimiz şey salt bir metin dosyası, txt dosyası. Normal şartlarda bunu bilgisayarınızda çalıştıramazsınız. Tarayıcı içerisinde "v8" dediğimiz bir engine var, motor var. Bu v8 motorunun yaptığı şey, sizin yazdığınız JavaScript kodlarını derleyerek çalışır hale gelmesini sağlıyor. NodeJS, bunu yapan bu v8 engine'ni alıp bir uygulama haline getirdi

(masaüstü uygulaması gibi). Ve bu sayede JavaScript kodlarınızı sizin bilgisayarınızda yani terminal üzerinde çalıştırabilir hale getirmiş oldu. Yani tamamen tarayıcıdaki motoru söküp bilgisayarı koymak gibi düşünebilirsiniz. Bu şekilde, JavaScript sunucu içerisine de girmiş oldu (çünkü sunucu dediğimiz şey de bir bilgisayar ve terminali var). Böylece şu oldu, biz Front-Endçiler, artık sunucu içerisinde de bir şeyler yapabiliyoruz, API yazabiliyoruz, JavaScript yazabiliyoruz, Database'e bağlanabiliyoruz.

## 12. Front-End'in kolu daha ne kadar uzamış olabilir?

Masaüstü uygulaması yazabiliyoruz (bundan dolayı, front-end'in kolu masaüstü uygulamalarına kadar uzamış durumda). HTML, CSS ve JavaScriptle yazılmış birçok masaüstü uygulaması var. Bunların içerisinde özellikle "Slack" ve "VS Code" masaüstü uygulamaları HTML, CSS ve JavaScript'le yazılmıştır.

React Native'in yaptığı şey mobil uygulama çıkarmak.

İşte Front-End aslında böyle dallandı dallandı büyüdü, televizyonlara kadar girdi. Televizyonların içerisindeki netflix uygulamasında bile HTML, CSS, JavaScripten oluşan bir yapı var. Hatta netflixteki filmlerin dizilerin kapak görselleri AVF teknolojisi denilen bir teknolojiyle yapılıyor. AVF teknolojisi, tarayıcılara yeni gelen bir özellik. Yani o netflixte gördüğünüz dizi film kapakları web teknolojisiyle gelen kısımlar. Bunun gibi birçok web teknolojisi var. (bu arada şunu söyleyeyim Netflix, Airbnb, Spotify'ın engineering bloglarını sayfalarını takip edin, çünkü oradaki tüm teknolojiler bizim web'e yön veriyor.

## 13. Erişilebilirlik Konusu

Erişilebilirlikle kastedilen şu: kolunuz kırıldı ya da tek kolunuzu kullanamadığınız bir durumda ya da geçici bir göz kaybı yaşadınız, bu gibi durumlarda size seslerle, görsel farklılıklarla yön gösteren ya da tek kolunuzla bile kullanmanıza yardımcı olan front-end'in konusudur.

## 14. Front-End Alanında Yaygın Olarak Kullanılan Frameworkler (veya Libraryler)

Günümüzde front-end alanında yaygın olarak kullanılan JavaScript frameworkleri React, Vue, Angular ve Svelte'dir. Bu frameworklerden biri projenin ihtiyacına göre tercih edilebilir.

## 15. jQuery Konusu

jQuery, bir JavaScript kütüphanesidir ve hala kullanılmaktadır. jQuery, eski bir teknoloji olmasına rağmen Türkiye'deki ve dünyadaki web sitelerinde hala vardır.

## 16. Web Components nedir?

Web Components şudur: bugün kullandığımız React, Vue, Angular'ın native hali. Yani artık tarayıcı kendi içinde size React gibi bir özellik veriyor. Siz oturup bir framework kullanmanız gerekmiyor. Bunu Web Components yaparak React gibi uygulamalar oluşturabiliyorsunuz küçük küçük React'la aynı mantıkta. Web Components'i inceleyin. Çünkü Web Components'i inceledikçe aa React'a da böyle yazabilirim dediğiniz konular da olacak. Web Components daha yeni yeni geliyor. İleride, React yazarken Web Components teknolojisine geçmek gerektiği zaman aa zaten aynı mantık, hiçbir şey değişmeyecek ben bunu biliyorum demeniz, sizin için çok büyük bir artı olacak. Bunu da unutmayın.

## 17. Jamstack Nedir?

Jam'in açılımı şudur: JavaScript API Markup bu üç kelimenin baş harflerinden "jam"i oluşturmuşlar. Stack de bildiğimiz stack. Mesela bir firmanın kullandığı teknolojilerin tümüne tech stack denir.

Şimdi bu Jamstack nasıl çıktı onu söyleyeyim: ilk başta bugün bir React, Angular veya Vue kütüphanesiyle yazdığımız bir projenin aslında hiçbir HTML çıktısı olmuyor. Yani tamamen JavaScript üzerinde yazdığımız için bunları, tamamen tarayıcı üzerinde derlenerekten o örüntü tarayıcıda yansımış oluyor ama içeriğine baktığımız zaman bomboş bir index.html dosyası görürüz. Şimdi Google da arama sonuçlarını listelerken şunu yapıyor; ben senin index.html dosyana girerim bakarım ne varsa onları indexlerim diyor. Örneğin bir websitemiz olduğunu düşünün kodluyoruz diyelim. Google robotu kodluyoruzun sitesinde girdiği zaman arayüze bakmaz, kodlarına bakar ve der ki sayfa içerisinde metinler var mı, ne var işte kodluyoruz var ekip var bootcamp var, bunları indexlerim der. Ama React, Vue, Angular ile yaptığımız projelerde bunların hiçbirisinin çıktısı o HTML'de olmaz. Bunların hepsi JavaScriptle oluştuğu için Google da şunu diyor: ya benim zamanım çok değerli, ben oturup sizin JavaScript kodlarınızı derleyemem, o yüzden index.html in içerisinde ne varsa ben onu alırım, gerisine karışmam diyor. Eğer index.html in içerisinde hiçbir şey yoksa ben senin sayfanı indekslemem der. Çünkü Google her gün milyonlarca websitesini tarıyor her gün ve bu milyonlarca websitesinin içinde JavaScriptleri çalıştırması ya da bunları render etmesi ayrı bir maliyet ve bu maliyete girmemek için aslında sadece index.html kısmına bakıyor. HTML kısmına bakıyor dediğimiz şey bildiğimiz siteye gidin kaynağı görüntüle diyin orda ne çıkarsa ben onu indekslerim diyor. Durum böyle olunca ve bu frameworkler de çok popüler olunca (React, Vue, Angular gibi), buna bir alternatif üretildi. İşte bu alternatif Jamstack'tir. İşte bu yüzden, Jamstack çıktı. Jamstack'in yaptığı aslında çok basit, sizin JavaScriptle yazıp ortaya çıkardığınız o dosyayı o görüntüyü HTML olarak kaydediyor. Yani baya bildiğimiz 20 sene önceki gibi her sayfanın bir HTML'i var. Ana sayfanız mı var, index.html, hakkınızda sayfanız mı var about-us.html, iletişim sayfanız mı var contact.html şeklinde ve hepsinin çıktısını verdiği için Google diyor ki bu sefer aa evet bunların çıktısı var ve ben bu HTML'i görebiliyorum diyor. Yani JavaScript aslında tamamen ortadan kalkmış oluyor. JavaScript sadece sizin geliştirme

ortamınızda var. Siz yine JavaScript yazıyorsunuz, React yazıyorsunuz, HTML'leri oradan oluşturuyorsunuz ama bu Jamstack teknolojisiyle aslında JavaScriptler derlenip HTML olarak çıkmış oluyor ve Google'a uyumlu hale ancak böyle gelebiliyor. Onun dışında React, Vue, Angular kesinlikle SEO (Search Engine Optimization-Arama Motoru Optimizasyonu) uyumlu değildir. Bunun için mutlaka bir teknoloji kullanılması gerekiyor Jamstack gibi. Bu arada Jamstack bir teknoloji değil, bir metodoloji, sadece bu sistemi biraraya getiren bir ekosistem aslında öyle diyebilirim. Yani React'ta yazdığınız şeyin HTML çıktısını almanıza veya Vue'da yazdığınız şeyin HTML çıktısını almanıza yardımcı olan bir yapı. Belki duymuşsunuzdur; next.js, nuxt.js, gatsby. Bu teknolojiler işte Jamstack teknolojileridir. Bunların yaptığı şey React'la yaptığınız şeyleri tamamen HTML'e dökmek için olan teknolojiler. Ayrıca bu Jamstack teknolojilerinin şöyle güzel bir avantajı var: bu teknolojilerin çıktısı HTML olduğu için JavaScript derlemesi gerekmiyor, bundan dolayı siteler, telefonlarda işlemcisi ve RAM'i düşük cihazlarda daha stabil ve daha performanslı çalışıyor böylece.

Not: Jamstack'e ilgi duyuyorsanız Jamstack.istanbul'u takip etmenizi öneririm.

## 18. Yararlı Linkler / Kaynaklar

https://www.awwwards.com/

https://Jamstack.istanbul/

https://www.11ty.dev/

https://www.smashingmagazine.com/

https://css-tricks.com/

https://speckyboy.com/

https://tympanus.net/codrops/

https://dev.to/

https://dev.to/devdevcharlie

https://hashnode.com/

https://vitejs.dev/