

Frontend Geliştiricileri İçin Prompt Yazma

[Open in app](#)[Sign up](#)[Sign in](#)

Medium



Berna Nur İlbaş

Follow



Share

Yapay zekâ ile çalışırken aldığınız cevabın kalitesi, çoğu zaman modelin zekâsıyla değil, sorunuzun kalitesiyle sınırlıdır. İyi haber: soru sormak öğrenilebilir bir beceri.

Yapay zekâ araçlarını kullanmaya başladığımda ilginç bir şey fark ettim: aynı aracı, aynı modeli kullanan iki geliştiriciden biri harika çıktılar alırken diğeri sürekli “bu ne alaka” dediği sonuçlarla boğuşuyordu. Uzun süre bunu modelin bir tuhaflığı sandım. Sonra anladım ki fark modelde değil, **prompt**'ta.

Prompt yazmak, bir frontend geliştirici için 2026'da tıpkı Git kullanmak ya da Chrome DevTools açmak gibi temel bir beceri haline geldi. Ve çoğu insan bunu ciddiye almıyor — editöre yarım ağız bir cümle yazıp “olmadı işte” diyor. Oysa iyi bir prompt, kafanızdaki net resmi makineye aktarmanın disiplini. Ne kadar net aktarırsanız, o kadar az düzeltme yaparsınız.

Bu yazıda, frontend işlerinde gerçekten işe yarayan prompt yazma tekniklerini, bol “kötü vs iyi” örneklerle anlatacağım. Amaç, editörünüze her yazdığınızda daha az uğraşıp daha çok iş çıkarmanız.

İyi Bir Frontend Prompt'unun Anatomisi

Kötü bir prompt genelde tek boyutludur: sadece görev. “Bir tablo yap.” İyi bir prompt ise dört katmandan oluşur:

1. Bağlam — hangi teknolojiler, hangi proje yapısı?

2. **Görev** — tam olarak ne istiyorum?

3. **Kısıtlar** — neye uymalı, neyi yapmamalı?

4. **Çıktı formatı** — sonucu nasıl istiyorum?

Bir örnekle görelim. Aynı isteği iki şekilde yazalım:

✗ Zayıf: “Bir arama kutusu yap.”

✓ Güçlü: “React + TypeScript ve Tailwind kullanıyorum. Bir SearchInput bileşeni yaz. Props: value (string), onChange (value: string) => void, placeholder (opsiyonel). Sol tarafında bir arama ikonu olsun, sağında değer varsa temizleme (x) butonu. Debounce YOK, sadece kontrollü input. Erişilebilir olsun: aria-label ekle. Sadece bileşen kodunu ver, açıklama yazma.”

İkincisi bağlamı (React/TS/Tailwind), görevi (SearchInput), kısıtları (debounce yok, erişilebilir) ve formatı (sadece kod) net veriyor. Sonuç, neredeyse hiç düzeltme gerektirmeyen bir bileşen olur:

tsx

```
import { Search, X } from “lucide-react”;
```

```
interface SearchInputProps {
```

```
  value: string;
```

```
  onChange: (value: string) => void;
```

```
  placeholder?: string;
```

```
}
```

```
export function SearchInput({ value, onChange, placeholder = “Ara...” }:  
  SearchInputProps) {
```

```
  return (
```

```
    <div className=”relative”>
```

```
      <Search className=”absolute left-3 top-1/2 h-4 w-4 -translate-y-1/2 text-gray-400” />
```

```
<input  
  
type="text"  
  
value={value}  
  
onChange={(e) => onChange(e.target.value)}  
  
placeholder={placeholder}  
  
aria-label="Arama"  
  
className="w-full rounded-lg border border-gray-200 py-2 pl-9 pr-9 focus:border-indigo-500 focus:outline-none"  
  
>  
  
{value && (  
  
<button  
  
type="button"  
  
onClick={() => onChange("")}  
  
aria-label="Aramayı temizle"  
  
className="absolute right-3 top-1/2 -translate-y-1/2 text-gray-400 hover:text-gray-600"  
  
>  
  
<X className="h-4 w-4" />  
  
</button>  
  
)}  
  
</div>  
  
);  
  
}
```

Zayıf prompt ise size class component mı fonksiyon mu, TypeScript mi JS mi, hangi stil kütüphanesi ile — hiçbirini bilmeden, kendi varsayımlarıyla dolu bir kod verir ve siz onu baştan uyarlırsınız.

Kural 1: Bağlamı Asla Varsaymayın

AI sizin kafanızdaki projeyi göremez. Hangi framework, hangi dil, hangi stil yaklaşımı, hangi state yönetimi kullandığınızı bilmez. Bunları söylemezseniz, tahmin eder — ve tahminleri sizin projenizle uyuşmaz.

Her prompt'un başına küçük bir “bağlam paketi” koymayı alışkanlık edinin:

“Stack: React 19, TypeScript, Tailwind, Redux Toolkit, React Router. Bileşenlerimiz fonksiyon bileşeni ve isimli export kullanıyor. Veri çekmeyi RTK Query ile yapıyoruz.”

Bunu her seferinde yazmak zahmetli geliyorsa — ki geliyor — bir sonraki bölümde bunun kalıcı çözümü var.

Kural 2: Bileşenin “Sözleşmesini” Tanımlayın

Frontend'de en değerli netlik, bileşenin API'sini önceden tanımlamaktır. Props neler, tipleri ne, hangi durumları (state) ele almalı, hangi olayları (event) yaymalı? Bunu net verdiğinizde AI tam istediğiniz şekli üretir.

✗ “Bir dropdown menü yap.”

✓ “Bir Dropdown bileşeni yaz. Props: options ({ label: string, value: string }[]), selected (string | null), onSelect ((value: string) => void), disabled (boolean). Dışarı tıklanınca kapanmalı, Escape tuşuyla kapanmalı, klavyeyle ok tuşları arasında gezilebilmeli.”

İkinci prompt, sadece “çalışan” değil, gerçekten kullanıma hazır bir bileşen çıkarır. Çünkü siz düşünme işini önceden yaptınız; AI sadece onu koda döktü.

Kural 3: Örnek Gösterin (Few-Shot)

Get Berna Nur İlbaş's stories in your inbox

Join Medium for free to get updates from this writer.

Enter your email

[Subscribe](#)☐ Remember me for faster sign in

AI kalıp taklidinde olağanüstü iyidir. Ona takip edeceği iyi bir kalıp vererseniz, ürettiği her şey o kalıba uyar. Bu, çıktının tasarım sisteminize oturması için en güçlü tekniktir.

“Aşağıda mevcut Button bileşenim var. AYNİ cva varyant yapısını, aynı prop desenini ve aynı isimlendirmeyi kullanarak bir IconButton bileşeni yaz:

tsx

[Button bileşeninizi buraya yapıştırın]

```

Bir örnek göstermek, on satır tarif yazmaktan daha etkilidir. “Şunun gibi ama şu farkla” kalıbı, frontend’de en çok işime yarayan prompt şablonlarından biri.

#### Kural 4: Kısıtları Açıkça Söyleyin

AI, söylemediğiniz şeyi bilemez. İstemediğiniz şeyleri de en az istediğiniz kadar net belirtin:

- “Ek kütüphane kurma, sadece React ve Tailwind kullan.”
- “Inline stil kullanma, sadece Tailwind sınıfları.”
- “Bu bileşeni erişilebilir yap: klavye navigasyonu ve ARIA nitelikleri ekle.”
- “Mobilde tek sütun, md breakpoint’ten sonra iki sütun olsun.”
- “any tipi kullanma, her şeyi tiple.”

Özellikle **erişilebilirlik ve responsive** davranış — AI’nın en sık atladığı iki şey. Bunları prompt’a baştan koymak, sonradan elle eklemekten çok daha hızlı.

#### Kural 5: Çıktı Formatını Kontrol Edin

AI’nın nasıl cevap vereceğini de yönlendirebilirsiniz, bu size zaman kazandırır:

- “Sadece kodu ver, açıklama yazma.” — Kopyala-yapıştır için ideal.

- “Önce yaklaşımı 2 cümleyle özetle, sonra kodu ver.” — Kritik bir bileşende, kararı anlamak için.
- “Değişiklikleri diff olarak göster.” — Mevcut kodu düzenlerken.
- “TypeScript, strict mode uyumlu olsun.” — Tip güvenliği için.

Ne istediğinizi söylemezseniz AI çoğu zaman uzun uzun açıklar, siz de kod parçasını o açıklamanın içinden ayıklarsınız. Küçük bir talimat, bu sürtünmeyi ortadan kaldırır.

## Kural 6: Tek Seferde Değil, Diyalog Halinde İlerleyin

En sık yapılan hatalardan biri, ilk çıktının mükemmel olmasını beklemek. Prompt yazmak monolog değil, diyalogdur. İlk taslağı alın, sonra konuşarak inceltin:

“Güzel. Şimdi bu kartı md breakpoint’te yatay layout’a çevir.” “Butona loading state ekle; yüklenirken spinner göster ve disable et.” “Bu bileşeni React.memo ile sarmalamak mantıklı mı? Neden?”

Her mesajda bağlam korunur, siz de küçük adımlarla tam istediğiniz yere gelirsiniz. Büyük ve muğlak tek bir prompt yerine, küçük ve net iterasyonlar her zaman daha iyi sonuç verir.

## Kalıcı Çözüm: Proje Kuralları (System Prompt / Rules)

Bağlamı her seferinde yazmak yorucu. Neyse ki modern AI editörleri (Cursor, Windsurf, Copilot vb.) proje geneli kalıcı talimatlar tanımlamanıza izin veriyor — genelde .cursorrules, .github/copilot-instructions.md ya da benzeri bir dosyayla. Bunu bir kez yazarsınız, her prompt’ta otomatik uygulanır.

Örnek bir kurallar dosyası şöyle görünür:

md

### # Proje Kuralları

- Stack: React 19 + TypeScript + Tailwind + Redux Toolkit
- Bileşenler: fonksiyon bileşeni, isimli export, PascalCase
- Stil: yalnızca Tailwind, inline stil yasak

- Tipler: `any` yasak, props her zaman `interface` ile tanımlanır
- Erişilebilirlik: interaktif her öğe klavyeyle kullanılabilir olmalı
- Olay işleyicileri `handle`, prop fonksiyonları `on` ön ekiyle
- Yeni bileşen üretirken mevcut `@/components/ui` desenlerine uy

Bu dosya, AI için kalıcı bir “biz burada böyle yapıyoruz” rehberidir. Bir kez kurduğunuzda, ürettiği her şey ekibinizin standartlarına uyar ve tekrar tekrar aynı bağlamı yazmaktan kurtulursunuz. Bence bir projede yapılabilecek en yüksek getirili tek AI yatırımı bu.

## Farklı İşler İçin Farklı Prompt Kalıpları

Prompt yazmak sadece “bileşen üret” değil. Frontend’de günlük olarak karşılaştığım üç senaryo ve kalıpları:

**Hata ayıklama (debugging):** Hatayı, ilgili kodu ve beklenen davranışı birlikte verin.

“Bu bileşen ilk render’da iki kez API çağrısı yapıyor. İşte kod: [...]. useEffect bağımlılıklarında sorun mu var, yoksa StrictMode etkisi mi? Adım adım açıkla.”

**Refactor:** Mevcut kodu ve hedefi net verin.

“Bu 200 satırlık bileşeni daha küçük parçalara böl. Veri çekme mantığını bir custom hook’a taşı. Davranışı değiştirme, sadece yapıyı temizle.”

**Öğrenme:** Cevabı değil, anlayışı isteyin.

“Bu kodda neden useCallback kullanılmış? Kaldırırsam ne olur, hangi durumda gerçekten gerekli? Basit bir örnekle açıkla.”

Fark ediyorsanız, her birinde ortak bir tema var: **ne istediğinizi netleştirmek**. Kalıp değişse de prensip aynı.

## Sık Yapılan Prompt Hataları

Son olarak, verimliliğinizi baltalayan yaygın anti-kalıpları toparlayalım:

1. **Çok muğlak olmak.** “İyi bir form yap” — “iyi” nedir? Kaç alan, hangi validasyon, hangi stil? Spesifik olun.

2. **Çok fazla şeyi tek prompt'a sığdırmak.** “Tüm dashboard’u, grafiklerle, filtrelerle, dark mode’la yap.” Parçalayın.
3. **Bağlam vermemek.** Framework, dil, stil yaklaşımını söylemeden iyi çıktı beklemek.
4. **Kısıtları unutmak.** Erişilebilirlik ve responsive’i sonradan hatırlamak yerine baştan istemek.
5. **İlk çıktıyı sorgusuz kabul etmek.** “Çalışıyor gibi” ile “doğru” farklı şeyler; iterasyon yapın, gözden geçirin.
6. **AI’nın uydurduğunu doğrulamamak.** Var olmayan bir prop veya fonksiyon önerebilir; şüphede dokümana bakın.

Prompt yazmak, aslında bir düşünme disiplini. İyi bir prompt yazabilmek için önce ne istediğinizi kendinizin netleştirmiş olması gerekir. Bu yüzden prompt yazmayı öğrenmek, aynı zamanda problemi daha net düşünmeyi öğrenmektir — ve bu, AI olsun olmasın işinize yarar.

Aklınızda tutmanız gereken tek şey varsa o da şu: AI’a düşünmeyi değil, yazmayı devredin. Bağlamı, kısıtları, bileşenin sözleşmesini, iyi çıktının neye benzediğini bilmek sizin işiniz. Onu net bir dille aktarmak, prompt yazma becerisi. Bu ikisini birleştirdiğinizde, AI gerçekten sizi hızlandıran bir ortağa dönüşür — muğlak isteklerinize muğlak cevaplar veren bir oyuncak değil.

İyi bir prompt, iyi bir soru gibidir: cevabın yarısını içinde taşır.

*Net sorular, net kodlamalar.* 🙌

Web Development Html



Follow

**Written by Berna Nur İlbaş**

0 followers · 1 following







