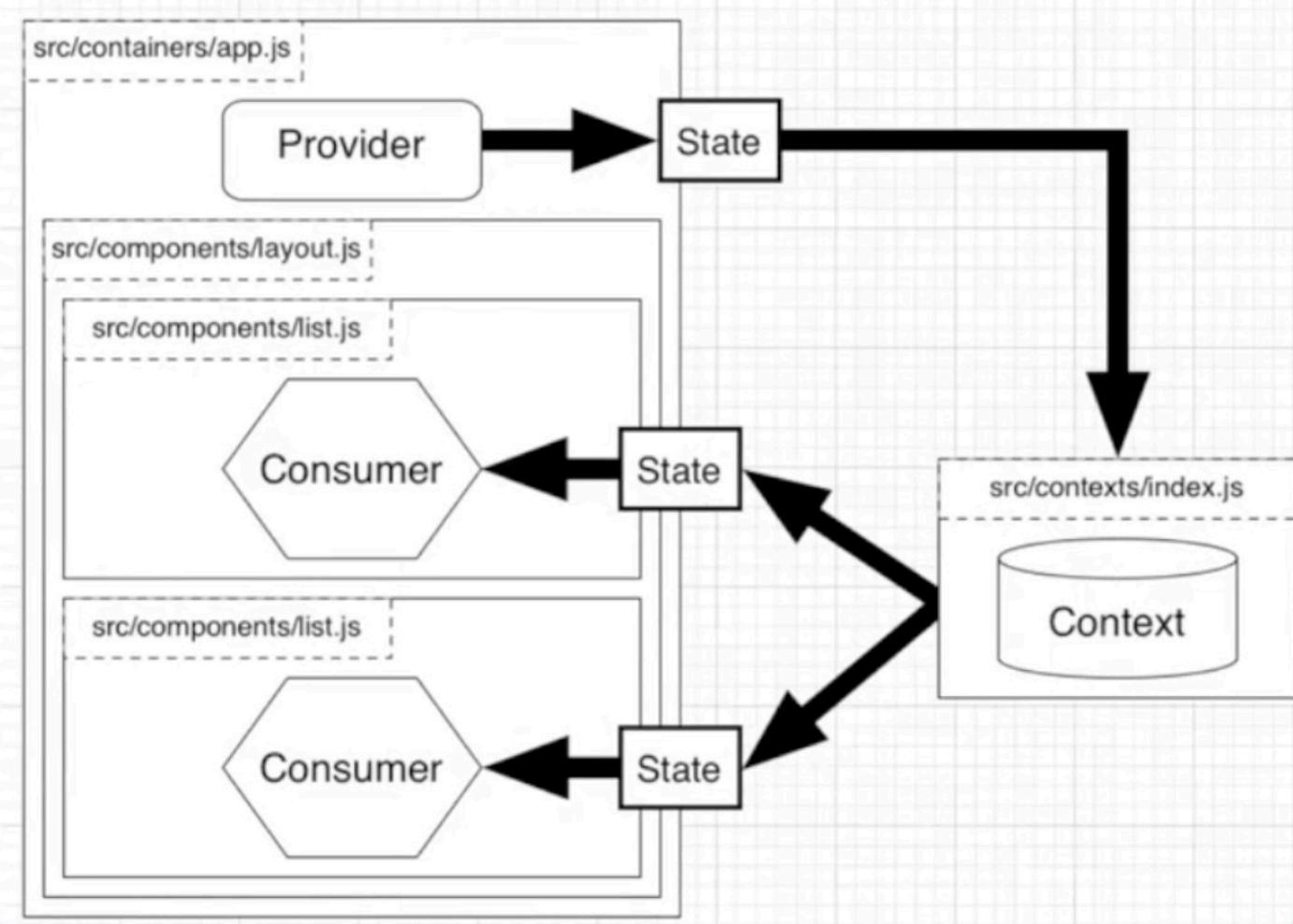


createContext

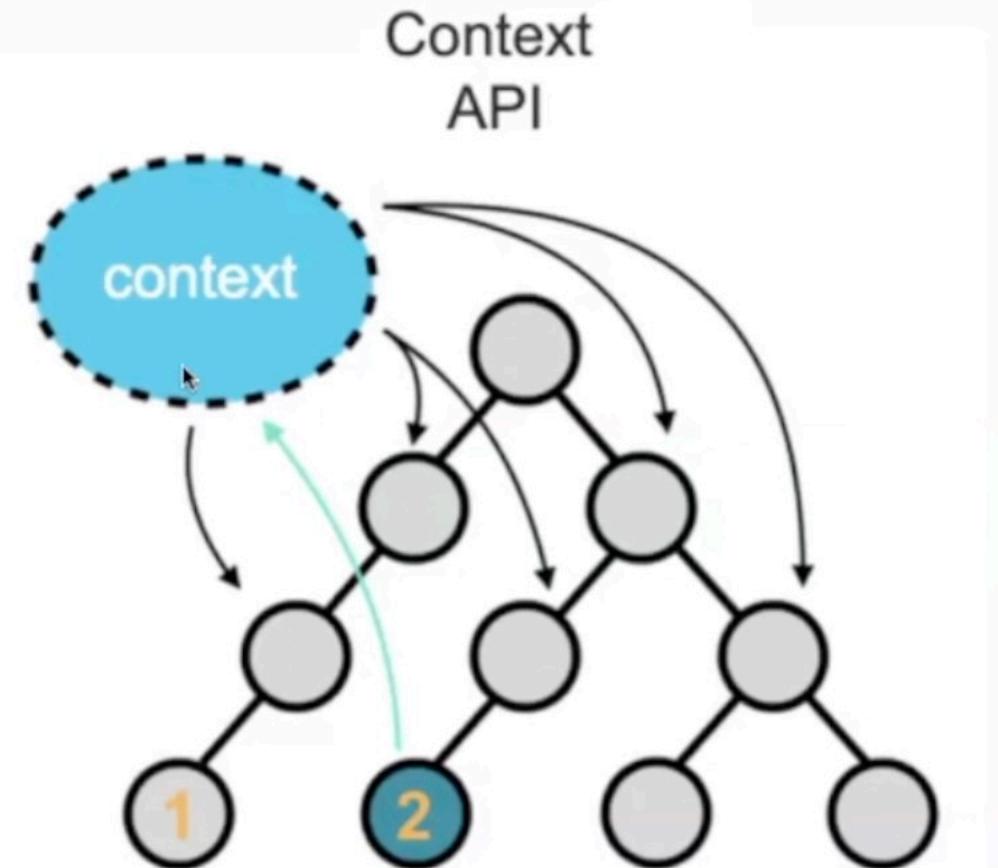
Context API içinde:
Provider ve **Consumer**
adında iki önemli
kavram vardır.

Provider ile
paylaştığımız state
değerlerine child
componentlerden
Consumer ile erişiriz.



Context API Nedir?

Context içerisinde
useState, **useReducer**,
localVariables kullanılabilir



Context API

Kurulum gerektirmez. React içerisinde gelir.

Yapısı gereği kompleks veri yönetiminden ziyade, daha basit verileri yönetmek için tercih edilir.

Theme, Language, Currency gibi...

Context API

Kompleks data tiplerinde ise genellikle Redux gibi daha kapsamlı bir state management çözümü tercih edilir.

createContext

Önce bir Context oluşturalım

```
import { createContext } from "react";

export const myContext = createContext();

function App() {
  return <Main />;
}

export default App;
```

createContext

Artık bir **context** objemiz var fakat bu React component lerine henüz bağlanmadı. Bunu **Provider** ile sağlamalıyız.

Provider & Consumer

```
export const myContext = createContext();

function App() {
  const [userName, setUserName] = useState("");

  return <myContext.Provider value={userName}>
    <Main />
  </myContext.Provider>
}


```

```
return (
  <div className="layout">
    <myContext.Consumer>
      {(value) => <Header userName={value} />}
      <PageBody />
      <Footer />
    </myContext.Consumer>
  </div>
);
```

Provider

Value olarak paylaştığımız nesneleri child component lerden alabiliriz.

- **useState**: state, setter
- **useReducer**: state, dispatch
- **localVariables**: var, let, const

React içinde tanımlanan her şey Provider value içerisinde paylaşılabilir.

Provider

```
function App() {
  const [userName, setUserName] = useState("");
  const [students, dispatch] = useReducer(studentReducer, []);
  const PI = 3.14;
  let count = 0;

  return (
    <myContext.Provider
      value={{ userName, setUserName, students, dispatch, PI, count }}>
      <Main />;
    </myContext.Provider>
  );
}
```

Consumer

Provider ile sağlanan **value** değer(ler)i Consumer ile yakalanabilir.

Consumer

```
return (
  <div className="layout">
    <myContext.Consumer>
      {(value) => <h2>{value.userName}</h2>}
    </myContext.Consumer>
  </div>
);
```

```
<myContext.Provider
  value={{ userName, setUsername, PI, count,
  students, dispatch }}>
</myContext.Provider>
-----  

value = {
  userName,
  setUsername,
  PI,
  count,
  students,
  dispatch,
};
```

useContext

Provide edilen datanın daha pratik ve daha çok tercih edilen yöntemidir.

```
const Main = () => {  
  const { userName, setUserName, PI } = useContext(myContext);  
  ...  
}
```

children Prop

React içinde özel bir prop türüdür.

Bir component'in **Content** alanı içine yazılanlar o componentin içine children adında prop olarak gönderilir.

children Prop

```
export const Title = ({ children }) => {
  return <h1 className="page-title">
    {children}
  </h1>;
};
```

```
const LoginPage = ({ setUserName }) => {
  return (
    <div className="page">
      <Title>
        Kullanıcı Girişи
      </Title>
      ...
    );
}
```

Context Componenti Oluşturma

Context ile alakalı tüm tanımları bir dosyada toplayalım.

Önce **src** klasörü altına **src/context** adında bir klasör oluşturalım ve içine de **UserContextProvider.js** adında bir dosya oluşturalım.

Context Componenti Oluşturma

UserContext adında context nesnemizi oluşturalım.

```
import { createContext } from "react";
export const UserContext = createContext();
```

Context Componenti Oluşturma

Şimdi de **UserContextProvider** componentini oluşturalım ve **children** propu alınsın.

```
import { createContext } from "react";

export const UserContext = createContext();

export const UserContextProvider = ({ children }) => {
  return <UserContext.Provider value={}>
    {children}
  </UserContext.Provider>;
};
```

Context Componenti Oluşturma

Artık UserContext içinde paylaşmak istediğimiz tüm state değerlerini bu component içinde tanımlayabiliriz.

```
export const UserContextProvider = ({ children }) => {
  const [userName, setUserName] = useState("");
  const [email, setEmail] = useState("");

  return (
    <UserContext.Provider value={{ userName, setUserName, email, setEmail }}>
      {children}
    </UserContext.Provider>
  );
};
```

Context Componenti Oluşturma

UserContextProvider projeye uygulanabilir.

```
import { UserContextProvider } from "./context/userContextProvider";

function App() {
  return (
    <UserContextProvider>
      <Main />;
    </UserContextProvider>
  );
}
```

Authentication Nedir?

Authentication: Kimlik doğrulama

Gerçek kullanıcının DB deki kullanıcı kaydı ile eşleşmesi işlemi.

Authentication

Kullanıcı adı & Şifre bilgileri

Her seferinde bu bilgiler girilir mi?

Otomatik login işlemi nasıl gerçekleşir?

Otomatik Kullanıcı Girişи

Bunun için kullanıcının bilgisayarında **login credential**datalarını saklamamız gereklidir.

Mesela **localStorage** bu bilgileri saklamak için güzel bir yer.

Peki **kullanıcı adı & şifre** gibi hassas dataları localStorage da saklamamız doğru olur mu?

Authorization

Authorization: Yetkiledirme / Yetki denetimi

Kullanıcının kim olduğu biliniyor, fakat gerçekleştireceği işlem için yetkisi var mı yok mu sorusuna cevap veren mekanizmadır.

Authorization

E-ticaret sitesinde ürünü satın almamış bir kullanıcı puanlama ve yorum yapabilir mi?

Bir öğrenci dersten aldığı not bilgisini düzenleyebilir mi?

Bir işletim sisteminde misafir kullanıcı, yönetici kullanıcısının şifresini değiştirebilir mi?

Otomatik Kullanıcı Girişi

Bir sistemde **authenticate** olmamız için öncelikle bir forma **kullanıcı adı** ve **şifre** bilgilerini girmemiz gereklidir.

Girdiğimiz bilgiler **backend** uygulamasına gönderilir

Backend gelendatalar ile **DB** deki dataları karşılaştırır.

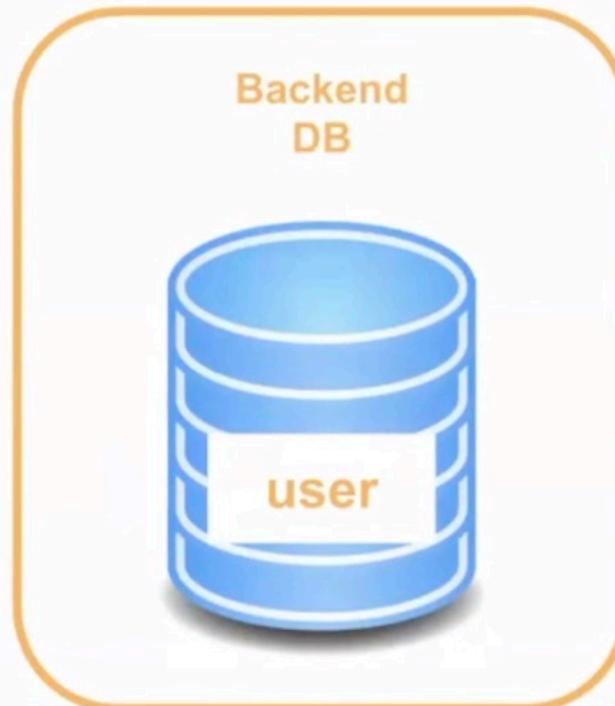
Otomatik Kullanıcı Girişİ

Kullanıcı **beni hatırla** seçeneğini seçtiyse?

Sayfa yeniden açıldığında kullanıcının otomatik login olması gereklidir.

Bunun için **localStorage** kullanılabilir.

Otomatik Kullanıcı Girişи



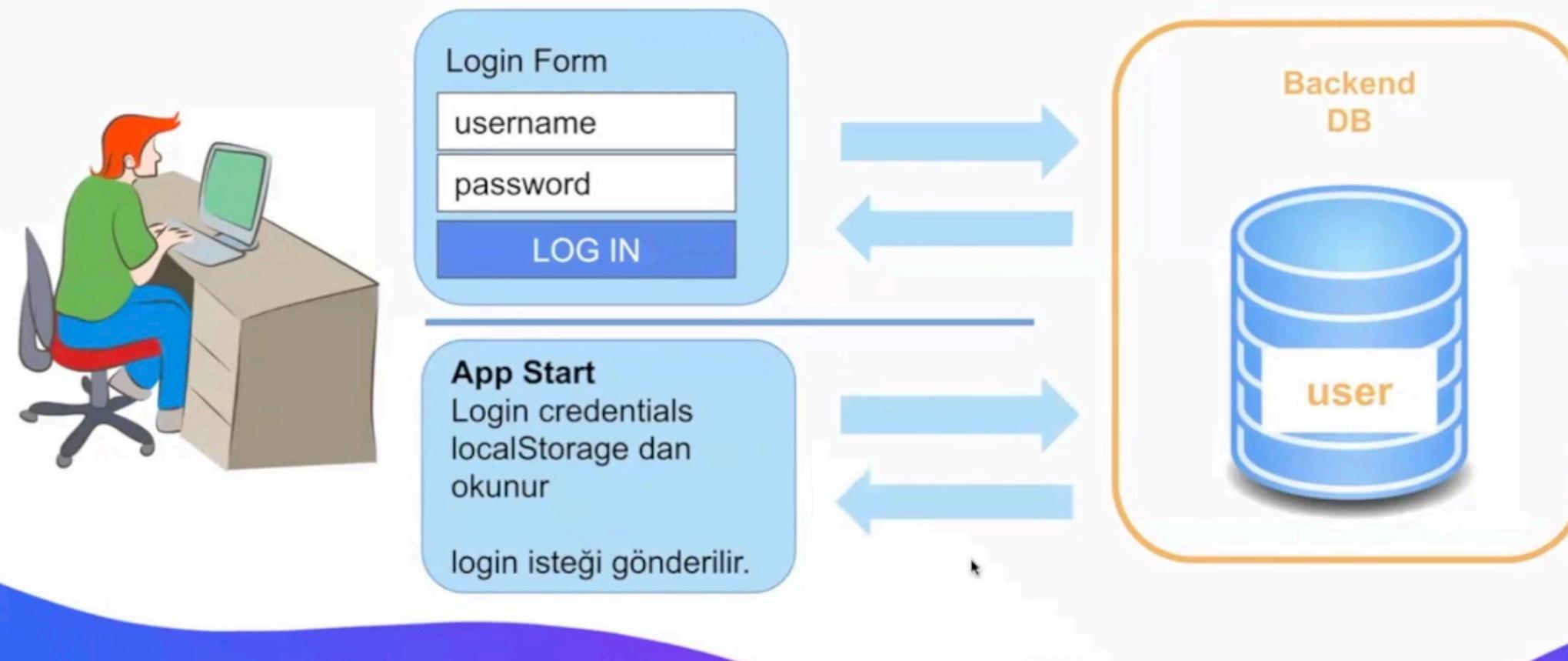
Otomatik Kullanıcı Girişİ

Kullanıcı önce login olmalı

Login credential lar localStorage a kaydedilmeli

Uygulama başladığında, localStorage içindeki credential datası ile login isteği gönderilmeli

Otomatik Kullanıcı Girişи



Otomatik Kullanıcı Girişİ

Peki kullanıcı adı ve şifresi gibi hassasdataları localStorage da kaydetmemiz bir güvenlik sorunu oluşturur mu?

Evet! Hassas veriler hiçbir zaman kullanıcı bilgisayarında kaydedilmemelidir.

Bu durumda bir kullanıcının kimliğini nasıl tespit edebiliriz?

Otomatik Kullanıcı Girişи

Kullanıcının hassas datalarını (kullanıcı adı, şifre) localStorage da kaydetmemeliyiz.

Eğer kullanıcı bilgilerini özel bir **ANAHTAR** ile şifrelersek bunu localStorage da saklayabiliriz.

Token

Geçici şifre: Expire date

Hassasdataları kullanmak yerine, özel bir **anahtar** ile **encrypt** edilen (şifrelenen) veridir.

```
user: {  
    email: "ali@abc.com",  
    expire: "12.02.2024",  
    IP: 24.97.173.11  
}
```



```
kd8hVmOk=WTY5Rh9NGgw-t  
ts7LQx6i-!zhFEBu64IpRQ2jqt  
DaiPpgMiqaNLa?qY,
```

Token

Otomatik Login işleminde kullanmak için uygundur.

Normal login işleminden sonra **backend** tarafından üretilip gönderilir.

Ardından **localStorage** a kaydedilir.

Bir daha ki açılışta bu geçici şifre **token** ile otomatik giriş yapılabilir.

Token



token localStorage a kaydedilir

App Start
token localStorage dan okunur
login isteği gönderilir.

Backend DB



Neden Korumalı Routing?

Sadece login olan kullanıcıya özel sayfalar

Ürün düzenleme

Tüm CRUD işlemleri

Korumalı Routing

Bir kullanıcının login olup olmadığını anlamamız gereklidir.

Bunu nasıl sağlayabiliriz?

Korumalı Routing Oluşturalım

Kullanıcı bir sayfayı açmak istediğiinde eğer login olmamışsa login sayfasına yönlendirmeliyiz.

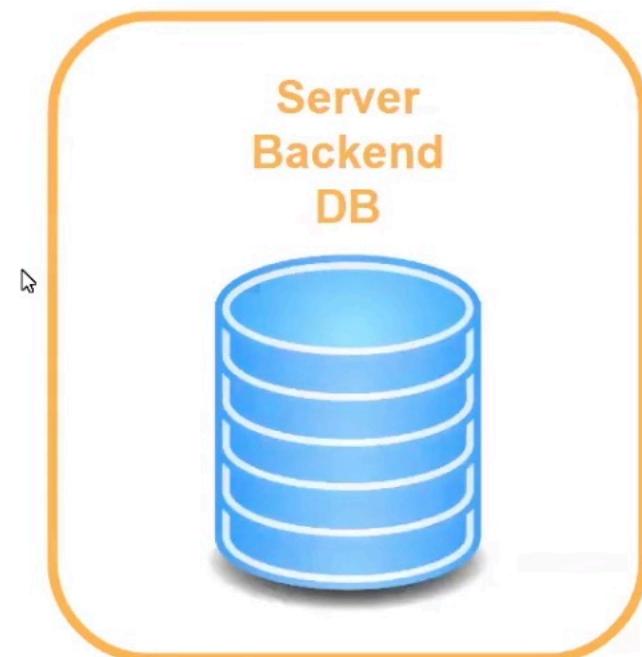
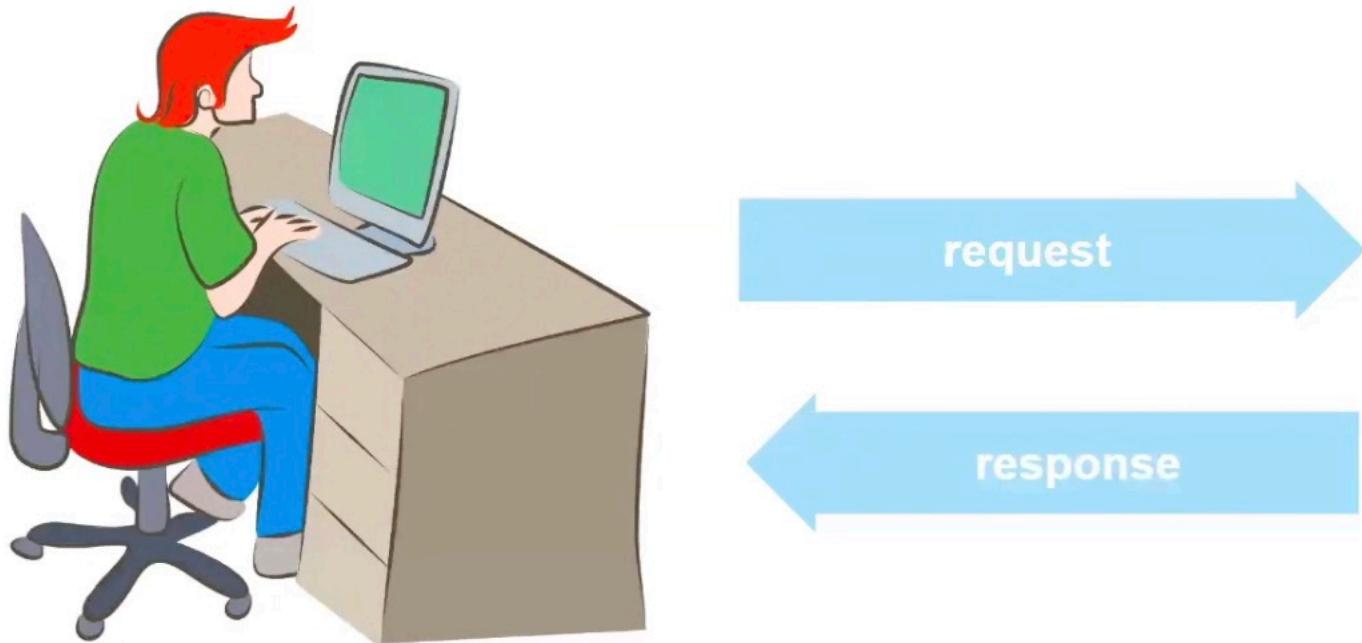
Bunu nasıl sağlayabiliriz?

Korumalı Routing Oluşturalım

```
import { Switch, Route, Redirect } from "react-router-dom";
...
<Route path="/sayac">
  {localStorage.getItem("token") ? (
    <SayacSayfa />
  ) : (
    <Redirect
      to={{
        pathname: "/login",
        state: { referrer: "/sayac" },
      }}
    />
  )}
</Route>
```

HTTP Request

Http Request'ler oluşturarak server ile iletişim kurarız.



HTTP Request

Bu iletişim ile

- Yeni data kaydedebiliriz **Create** **POST**
- Data okuyabiliriz **Read** **GET**
- Data güncelleyebiliriz **Update** **PUT**
- Data silebiliriz **Delete** **DELETE**



HTTP Request

Bu iletişim ile

CRUD İşlemleri

- Yeni data kaydedebiliriz **Create** **POST**
- Data okuyabiliriz **Read** **GET**
- Data güncelleyebiliriz **Update** **PUT**
- Data silebiliriz **Delete** **DELETE**

HTTP Request

JS'de **Http Request** gerçekleştiren birçok kütüphane - fonksiyon vardır

- axios
- fetch
- xmlHttpRequest 
- jQuery
- ajax

HTTP Request: GET

axios.get(url [, config])

```
// Read: GET
axios
  .get("https://workintech-fe-ecommerce.onrender.com/verify", {
    headers: {
      Authorization: token,
    },
  })
}
```

HTTP Request: POST

axios.post(url [, data [, config]])

```
// Create: POST
axios
  .post("https://workintech-fe-ecommerce.onrender.com/login", {
    email: formData.userName,
    password: formData.password,
  })
```

HTTP Request: PUT

```
axios.put(url [, data [, config ]])
```

```
// Update: PUT
axios
  .put(
    "https://620d69fb20ac3a4eedc05e3a.mockapi.io/api/products/" + product.id,
    product
  )
```

HTTP Request: DELETE

axios.delete(url [, config])

```
// Delete: DELETE
axios
.delete(
  "https://620d69fb20ac3a4eedc05e3a.mockapi.io/api/products/" + productId
)
```



HTTP Request

https://axios-http.com/docs/req_config

Döküman içinde bir axios (HTTP) request inin alabileceği konfigürasyon değerlerini inceleyebiliriz.

HTTP Request

Request Header alanı ile yapacağımız isteğe çeşitli parametreler ekleyebiliyoruz. Mesela **token** değeri bu parametrelerden birisi.

```
// Read: GET
axios
  .get("https://workintech-fe-ecommerce.onrender.com/verify", {
    headers: {
      Authorization: token,
    },
  })
```

Axios Instance Nedir?

baseURL değerini ve **header** gibi bazı konfigürasyonları her istekte verdiğimiziz düşünelim.

Bu kod tekrarına sebep olacaktır.

Kod tekrarını önlemek için ise bu özelliklere sahip boş bir axios örneği oluştururuz. Ve tüm isteklerimizi bu axios örneği ile gerçekleştiririz.

Axios Instance

src/api adında bir klasör oluşturalım
api.js adında bir dosya oluşturalım

```
export const createApiInstance = () => {
  const token = localStorage.getItem("token");

  return axios.create({
    baseURL: "https://620d69fb20ac3a4eedc05e3a.mockapi.io/api/",
    headers: {
      Authorization: token,
    },
  });
};

export let API = createApiInstance();
```

Artificial Intelligence (AI) Nedir?

İnsan davranışlarını simüle eden teknoloji.

insan öğrenmesi

algılaması

yaratıcılığı

karar verme
yeteneği

problem çözme
yeteneği

bağımsız hareket
etme yeteneği

Yapay Zeka

Nesneleri ayırt edebilir

İnsan'a ihtiyaç duymadan
harekete geçebilir

İnsan dilini anlayabilir ve
cevap verebilir

Yeni bilgi ve deneyimlerden
öğrenebilir

Yeni metinler, resimler,
videolar oluşturabilir

Yapay Zeka Nasıl Çalışır?

1950's

˲ Artificial intelligence (AI)

Human intelligence exhibited by machines

1980's

˲ Machine learning

AI systems that learn from historical data

2010's

˲ Deep learning

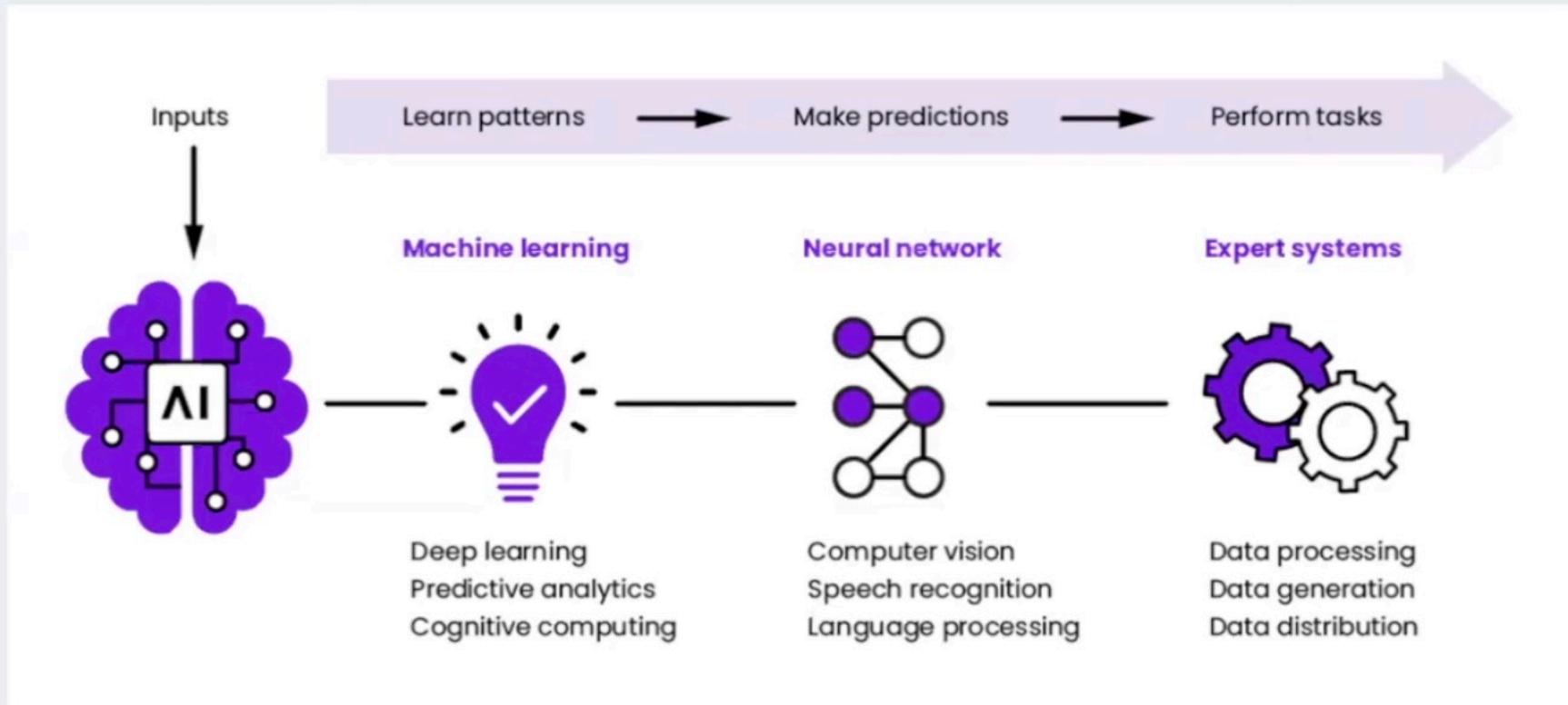
Machine learning models that mimic human brain function

2020's

˲ Generative AI (Gen AI)

Deep learning models (foundation models) that create original content

Yapay Zeka Nasıl Çalışır?



Yapay Zeka'nın Faydaları

Tekrarlayan görevlerin
otomasyonu

Verilerden daha hızlı ve
kapsamlı içgörüler alma

Gelişmiş karar alma
sureçleri

Daha az insan hatası

7/24 erişilebilirlik

Fiziksel risklerin azaltılması

Müşteri Deneyimi, Hizmet
ve Destek

Dolandırıcılık Tespiti

Kişiselleştirilmiş Pazarlama

Tahmine Dayalı Bakım

Uygulama Geliştirme ve
Modernizasyon

İnsan Kaynakları ve İşe Alım

Yapay Zeka'nın Problemleri ve Riskleri

Veri Riskleri

Model Riskleri

Operasyonel Riskler

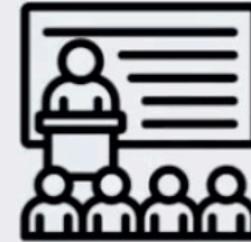
Etik ve Yasal Riskler



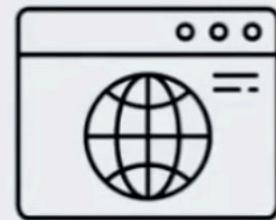
Kitap



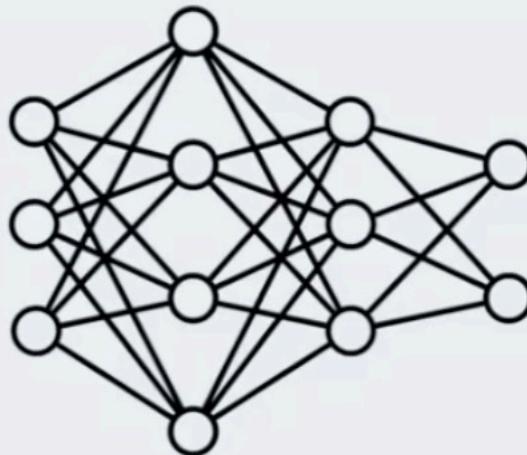
Dokümanlar



Eğitim Materyalleri



Web Sayfaları



Large Language Model (LLM)

Doğal dilleri(Natural Language) anlayabilen ve üretebilen yapay zeka (AI) sistemidir.

LLM'in Kullanım Alanları

Metin üretimi

İçerik özetleme

Yapay zeka asistanları

Yazılım kodu üretimi

Duygu analizi

Dil çevirisi

En İyi LLM Modelleri

BERT

Gemini 2.0 Flash

Gemma

GPT-4

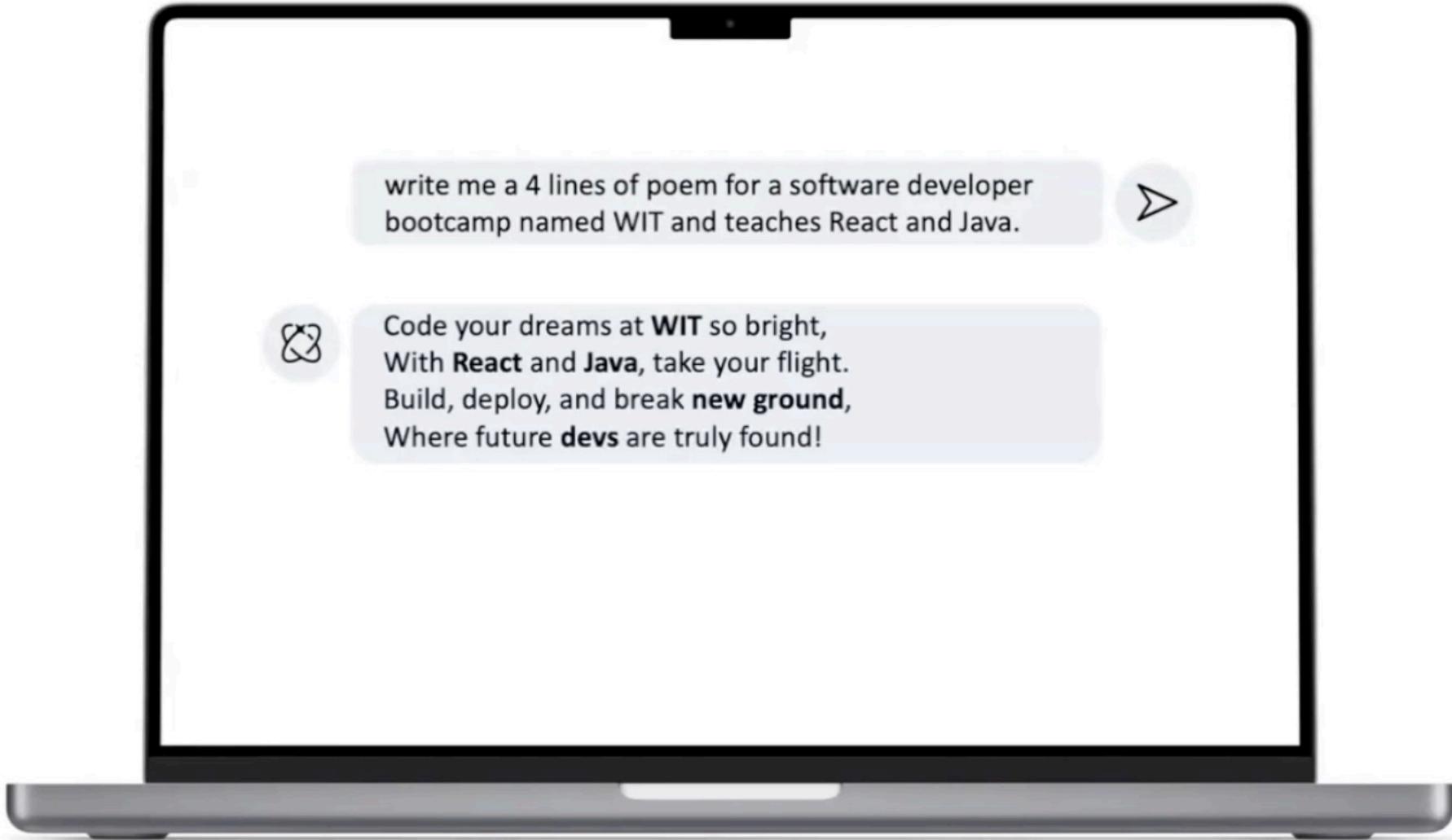
GPT-4o

Claude

DeepSeek-R1

Llama

Granite



write me a 4 lines of poem for a software developer bootcamp named WIT and teaches React and Java.



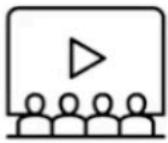
Code your dreams at **WIT** so bright,
With **React** and **Java**, take your flight.
Build, deploy, and break **new ground**,
Where future **devs** are truly found!

write me a 4 lines of poem for a software developer
bootcamp named WIT and teaches React and Java.

Prompt

Gen AI sistemleri, aldıkları istemlerin(**prompt**) kalitesine göre çıktı üretirler.

Nasıl Çalışır?



Gen AI, dil yapısını anlayan
Transformer mimarisi üzerine
inşa edilmiştir.



Prompt, AI modellerinin
verdiği çıktıları yönlendirir.
İterasyonlara dayanır.



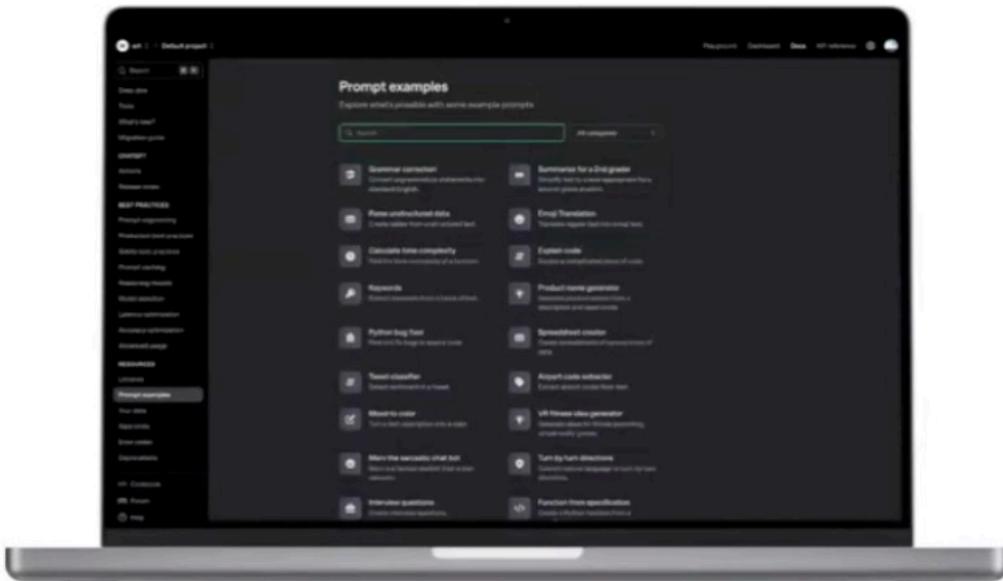
Model parametreleri ile AI
modelleri daha faydalı ve
alakalı çıktılar üretir.
Örn: token, temperature v.b.



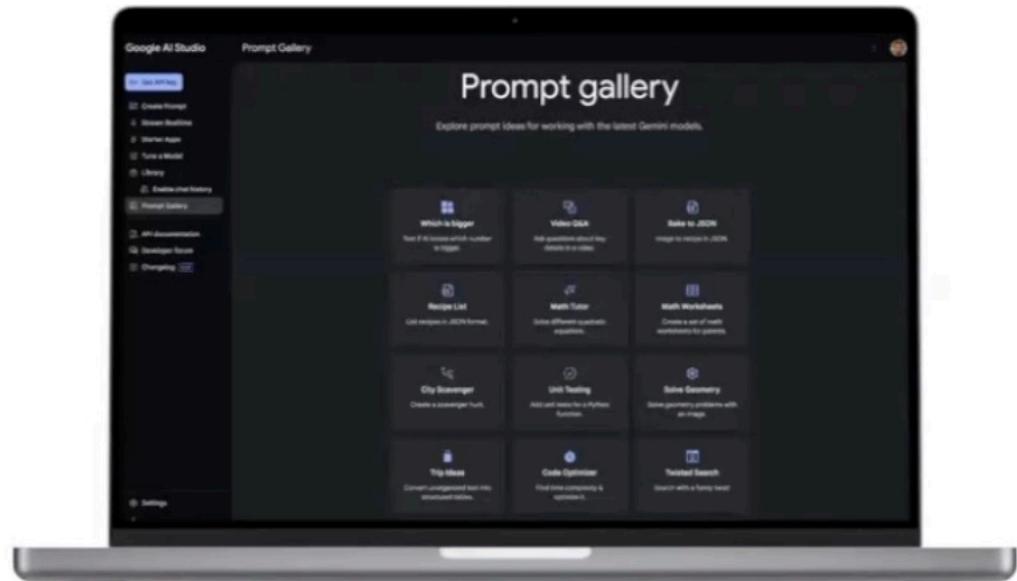
Prompt Engineer

- Büyük dil modellerine (LLM) aşinalık
- Güçlü iletişim becerileri
- Teknik kavramları açıklama yeteneği
- Programlama bilgisi (tercihen python)
- Veri yapıları ve algoritmalar konusunda sağlam bir temel
- Yaratıcılık ve yeni teknolojilerin avantajlarını ve risklerini değerlendirme yeteneği

Nasıl Öğrenirim



OpenAI Platform



Google AI Studio



AI API Oyun Alanları (Playground)

- Kodlamaya gerek olmadan denemeler yapmayı sağlar.
- Farklı modeller üzerinde çalışabiliriz.
- API ayarlarını test edebiliriz.
- Örneklerden öğrenebiliriz.
- API kullanımı için **Key** üretebiliriz.



Google AI Studio

Bilinmesi Gereken Parametreler

Token

1 token = ~4 karakter

Model

LLM model

Temperature

Yaratıcılık derecesi [0-2.0]

API Rollerleri ve Hiyerarşisi

Platform > Developer(System) > User > Assistant

AI API'sini kullanma

1. Gerekli kütüphaneyi kur

```
npm install @google/generative-ai
```

2. "GoogleGenerativeAI" metodunu import et.

```
import { GoogleGenerativeAI } from "@google/generative-ai";
```

AI API'sini kullanma

- Gen AI instance'sını oluştur.

```
const genAI = new GoogleGenerativeAI("YOUR_API_KEY");
```

- Model tanımla, prompt oluştur.

```
const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });
const prompt = "Explain how AI works";
```

- İstek yap(async) ve response'u bir değişkende sakla.

```
const result = await model.generateContent(prompt);
```

Response Daki

- Response'daki çıktıyı kullan. (Markdown formatında)

```
console.log(result.response.text());
```

Markdown

Element	Markdown Syntax
Heading	# H1 ## H2 ### H3
Bold	bold text
Italic	<i>italicized text</i>
Blockquote	<p style="padding-left: 2em;">↳ blockquote</p>
Ordered List	1. First item 2. Second item 3. Third item
Unordered List	- First item - Second item - Third item
Code	`code`
Horizontal Rule	---
Link	[title](https://www.example.com)
Image	![alt text](image.jpg)

AI API'sini kullanma

4. Gen AI instance'sını oluştur.

```
const genAI = new GoogleGenerativeAI("YOUR_API_KEY");
```

5. Model tanımla, prompt oluştur.

```
const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });
const prompt = "Explain how AI works";
```

6. İstek yap(async) ve response'u bir değişkende sakla.

```
const result = await model.generateContent(prompt);
```

7. Response'daki çıktıyı kullan. (Markdown formatında)

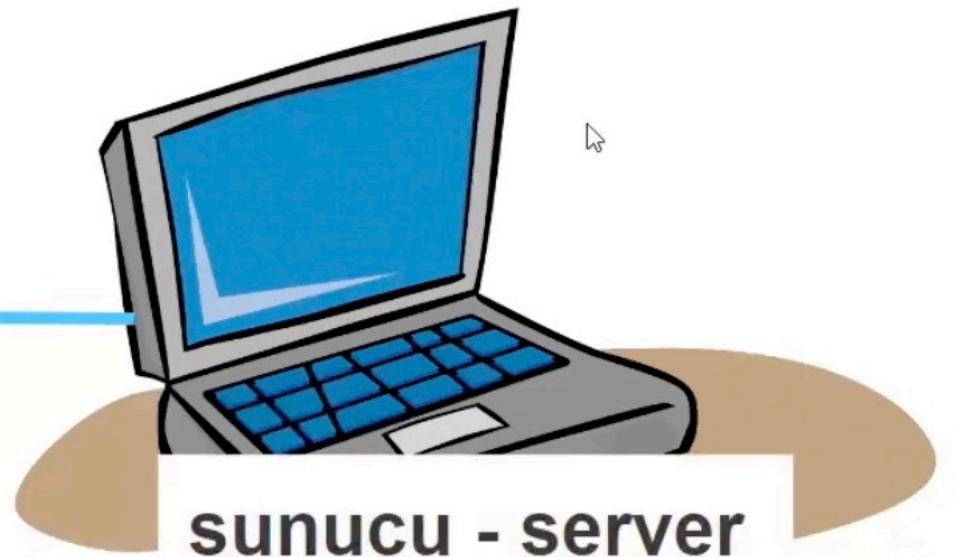
```
console.log(result.response.text());
```

Global Ağlar

En az iki bilgisayar arasında bir bağlantı kurulduğunda buna **network - ağ** denir

Veri isteyen: **istemci - client**

Veri sağlayan: **sunucu - server**



Global Ağlar

Ağ - Network daha büyük ölçeklerde de olabilir:

Resmi kurumlar arası
Okullar arası
Özel Şirketler arası

Dünyadaki tüm bilgisayarlar arası kurulan global ağa ise:

inter + network : **internet**

Global Ağlar: WWW Ağı

Bu çapta gelişmiş bir ağ bağlantısı içinde data paylaşmak için belirli prokoller - standartlar sağlanmalı.

Günlük hayatta kullandığımız internet aslında internet ağı içindeki **WWW** özel ağ çeşididir.

Bu derece büyük bir ağ, beraberinde büyük ölçekli **veri sağlayıcı - server** ihtiyaçları da getirdi.

Server

Fakat veri isteyen - client rolündeki bilgisayarların sayısı arttıkça, basit bir bilgisayar ve basit bir bağlantı server görevi için yeterli olmamaya başladı.



Server

Server - Sunucular

- Güçlü işlem gücü
- Yüksek kapasiteli ve hızlı bellek
- Sürekli ve güçlü ağ bağlantısı
- Sabit bir ortam sıcaklığı

Server Room

Gerekli koşulları sağlayıp **server** hizmeti veren özel şirketler vardır: Vercel, goDaddy...



02:05



Server / Hosting

Bir web sayfasının **server** içinde barınmasına / yayınlanmasına **hosting** denir.

Hosting hizmeti aldığımız zaman web sayfamızın erişim adresi ve belirli bir alan bize tanımlanır.

Bu adres IP'dir. Ör: **104.233.38.44**



Server / Domain

Eğer bir web sayfasının IP adresini bilirsek o adres ile de erişim sağlayabiliriz.

Mesela browser'a : **40.114.177.156** IP adresini girelim

<https://duckduckgo.com/> reklamsız arama motoru adresine dönüştüğünü farkettiniz mi?

Server / Domain

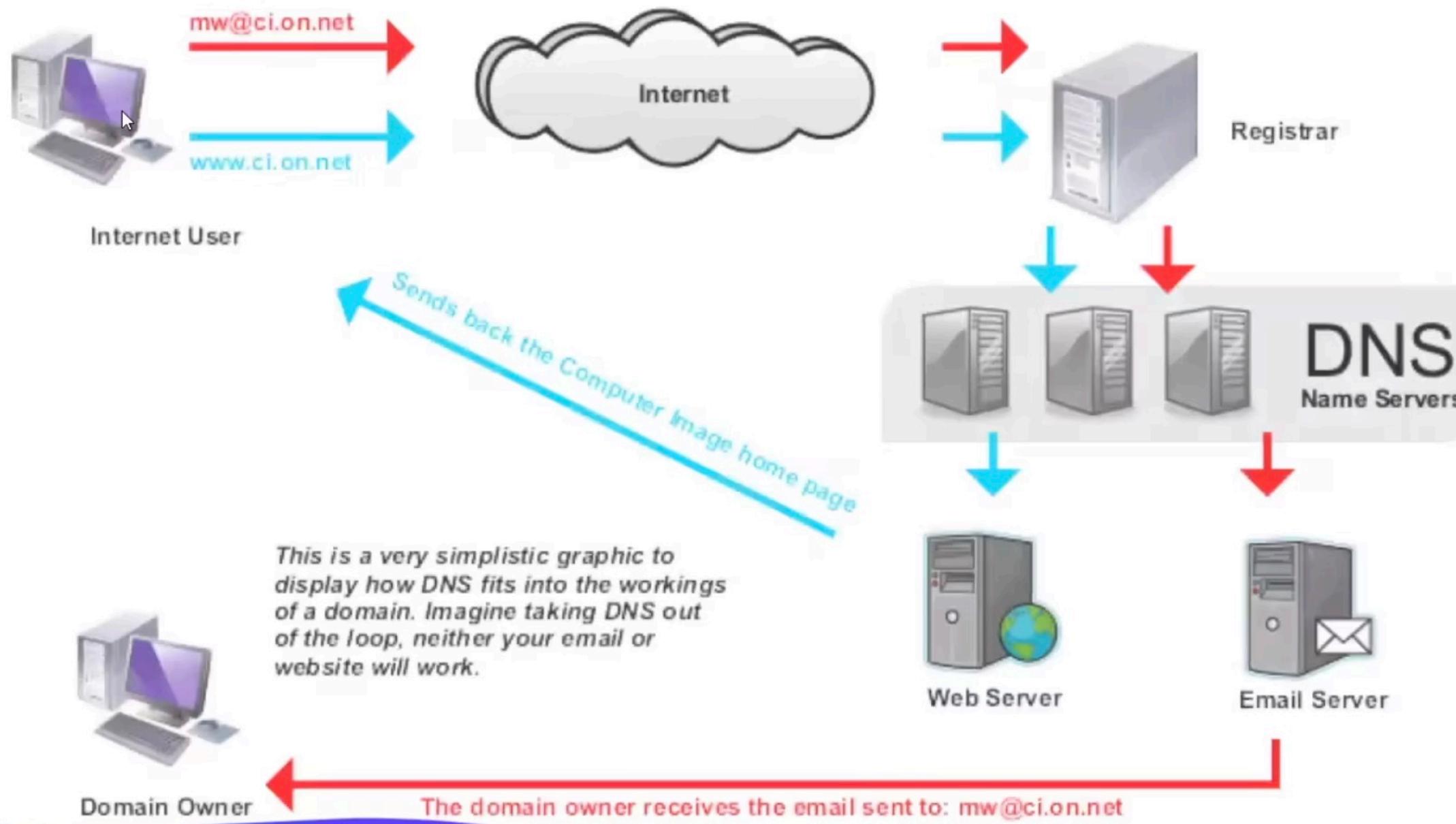
Bizim telefon numaralarını ezbere bilmediğimiz, rehbere isim ile kaydettiğimiz gibi

Web sayfalarının da isminin kaydedildiği uluslararası rehber ağı vardır.

Buna **Domain Name Service** denir. Bu hizmeti veren özel kuruluşlar vardır.

 **Hosting** hizmeti aldığımız gibi **Domain** hizmeti de almalıyız.

Bir Web Sayfası Nasıl Açıılır?



Bir Web Sayfası Nasıl Açıılır?

Birisini telefon ile arayacakken ne yaparız?

Önce rehberi açar, orada o kişinin ismini buluruz.
Sonra o isme kayıtlı numaraya erişiriz.
Ve o numarayı kullanarak aramayı başlatırız.



Tarayıcıya herhangi bir web sayfasının adını yazdığımızda da benzer bir süreç işler.

Deploy Nedir?

Yaptığımız web sayfasının kodlarını server'a yükleme işlemidir.

Server hizmeti satın aldığımız şirket bize bir cloud hesabı tanımlar. O hesabı kullanarak arayüz üzerinden veya FTP programları ile sayfamızın kodlarını server daki harddisk alanımıza kopyalayabiliriz.

Peki React Uygulaması deploy etmeye hazır mıdır?

React Uygulamasını Deploy Etmek

React uygulaması localimizde development'a uygun bir setup ile çalışır ve bu haliyle deploy edilmez.

Deploy etmeye uygun bir hale getirmemiz için

```
> npm run build
```

kodunu çalıştırılmamız gereklidir.

React Uygulamasını Deploy Etmek

Build yaptığımızda **build** adında bir folder oluşturulur

Projede kullanılan kodlar bir araya getirilerek bir tane JS ve bir tane CSS dosyasına dönüştürülür.

Ve tüm kodlar minify edilerek boyut olarak düşürülür.

Bu dosyalar public içindeki HTML template ine tanımlanır.