Web Programlama
10. Hafta

25.11.2024

Mühendislik Fakültesi
Bilgisayar Mühendisliği

Hazırlayan: Dr. Ercan Ezin

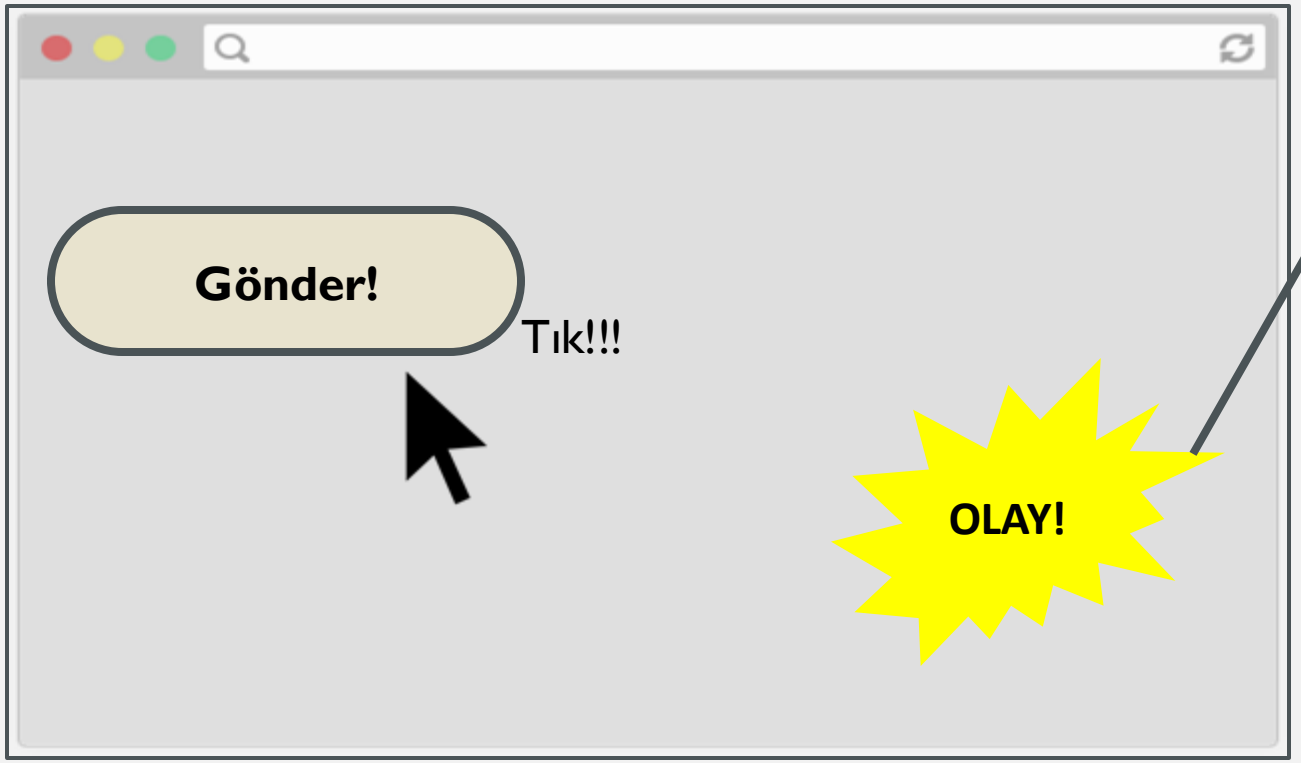
GİRİŞ

JavaScript Olaylar ve DOM yapısı, Console

Not: Bu dersin içeriği kaynakçada belirtilen materyallerde derlenerek üretilmiştir.
Credit: Content of the course has been picked from various generously shared sources listed in the credit section

JS: EVENTS

JS kodları genelde kullanıcı etkileşimi sonucu çalışır. Çok nadiren sayfa yüklendiğinde JS olayları devreye girer.




```
function onClick() {
   ...
}
```

JS olayları genelde **event handler** adında bir fonksiyon aracılığıyla çalışır.

EXTRA HTML ELEMENTS

Buttons:

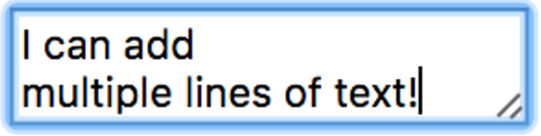
```
<button>Click me</button>
```

Single-line text input:

```
<input type="text" />
```

Multi-line text input:

```
<textarea></textarea>
```

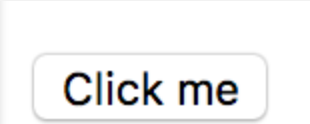


JS olayları için yukarıdaki elemanlarla etkileşimde sık kullanılan yöntemler vardır.

EXAMPLE

Konsola bir metin yazdırmak için event handler kullanabiliriz. Örneğin aşağıdaki gibi bir buton için:

```
<button>Click me</button>
```

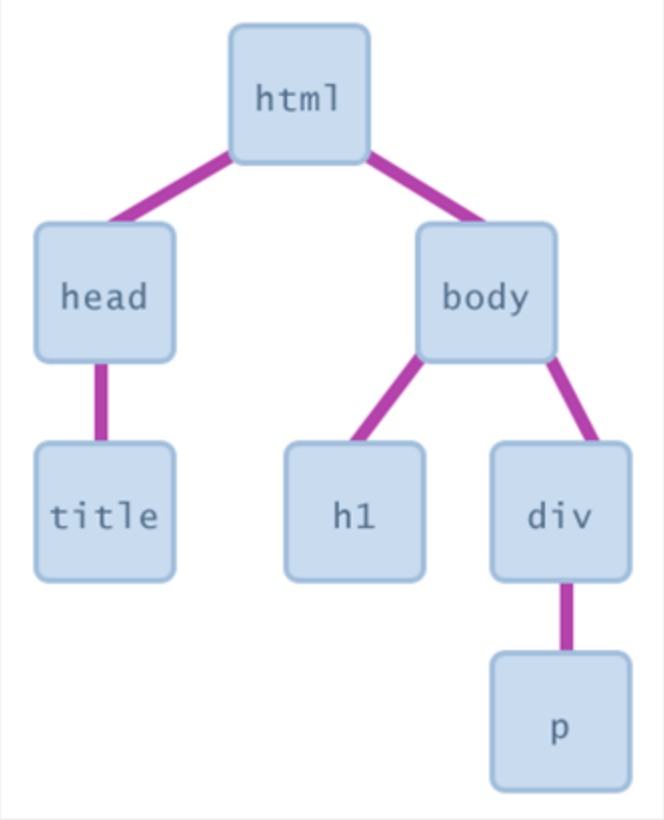


Soru: Nasıl bir event handler kullanabiliriz?

DOM(DOCUMENT OBJECT MODEL)

Html sayfasındaki her eleman taryıcı tarafından DOM ağaç yapısına çevrilir. Amaç içi içe geçen nesneleri ve viewport üzerindeki gösterimi düzenlemektir.

```
<html>
        <head>

        <title></title>
        </head>
        <body>
                <h1></h1>
                <div>

        <p></p>
                </div>
        </body>
</html>
```



DOM PROPERTIES

DOM, bir sayfadaki HTML elemanlarına karşılık gelen öğelerin bir **ağacıdır**.

• Bir elemanın durumunu görmek için bu öğeleri inceleyebilirsiniz.

(örneğin, bir metin kutusuna kullanıcının ne yazdığını almak için)

• JS kodu yazarak bir elemanın **en-boy** gibi özelliklerini değiştirmek için bu ağaç yapısındaki öğelerin özelliklerini kullanabilirsiniz.

(örneğin, bir stilin görünürlüğünü açmak veya kapatmak ya da bir `<h1>` etiketinin içeriğini değiştirmek için)

• JS kodu ile DOM'un oluşturduğu ağaç öğelerine yenilerini ekleyerek veya mevcutları çıkararak bir web sayfasına elemanlar ekleyebilir veya elemanları kaldırabilirsiniz.

ACCESS DOM OBJECTS

- querySelector fonksiyonu ile bir HTML elemanının DOM ağacındaki öğesine Javascript kodu yazarak ulaşabiliriz:

```
document.querySelector('css lokasyonu');
```

  - Yukarıdaki kod css lokasyonunun eşleştiği **ilk** elemana erişir.

- Ve querySelectorAll fonksiyonu ile:

```
document.querySelectorAll('css lokasyonu');
```

  - Css lokasyonunun eşleştiği **tüm** elemanlara erişebiliriz.

Örnek css lokasyonu:‘`a.link  h1`’

EXAMPLE SELECTORS

```
// Aşağıdaki kod ile DOM öğeleri içerisinde id’si button olan
ilk eleman seçilir, yoksa null değeri submitButton
değişkenine atanır.
let submitButton = document.querySelector('#button');


// Soru : Aşağıdaki kod neyi seçer?
let elemanListesi =
    document.querySelectorAll('.quote, .comment');

// Çözüm: quote veya comment sınıflarına sahip Dom öğelerinden oluşan
Bir liste seçilir ve elemanListesi değişkenine atanır.
```

DOM: ADDEVENTLISTENER

Her DOM nesnesinin/öğesine aşağıdaki gibi bir metodu ile olay dinleyici(event listener)tanımlayabilirsiniz:

`addEventListener(`***olay adi***`,` ***fonksiyon adi***`);`

- ***Olay adi:*** İlgili elemanın hangi olayı tetiklendiğinde bu olay dinleyicisinin çalıştırılacağını belirtir.
  - Örnek: `click` *yada* `focus`
- ***Fonksiyon adi:*** İlgili olay olduğunda yapmak istediğimiz algoritmanın yer aldığı JS fonksiyonunun adıdır.

HATIRLATMA: addEventListener ile bir dinleyici tanımlanmışsa silmek için `removeEventListener(`***olay adi***`,` ***fonksiyon adi***`);` metodunu kullanabilirsiniz.

EVENT LISTENER: EXAMPLE

```
<html>
▼<head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js"></script>
  </head>
▼<body>
    <button>Click Me!</button>
  </body>
</html>
```

JS PRATİK : Yandaki gibi tanımlanmış bir HTML kodunda kullanıcı butona bastığında konsola bir metin yazalım.

```
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

Hata verir. Neden ?

EVENT LISTENER: EXAMPLE

```html
<html>
▼<head>
    <meta charset="utf-8">
    <title>First JS Example</title>
    <script src="script.js"></script>
  </head>
▼<body>
    <button>Click Me!</button>
  </body>
</html>
```

Script dosyası içeriği ve tanımlaması sayfanın en üstündedir ve DOM daha oluşmadan EventListener eklemeye çalıştığından hata alınır.

ÇÖZÜM?

1- defer kullanmak. Bu sayede DOM oluşmadan JS çalıştırılmaz.

```
<script src="script.js" defer></script>
```

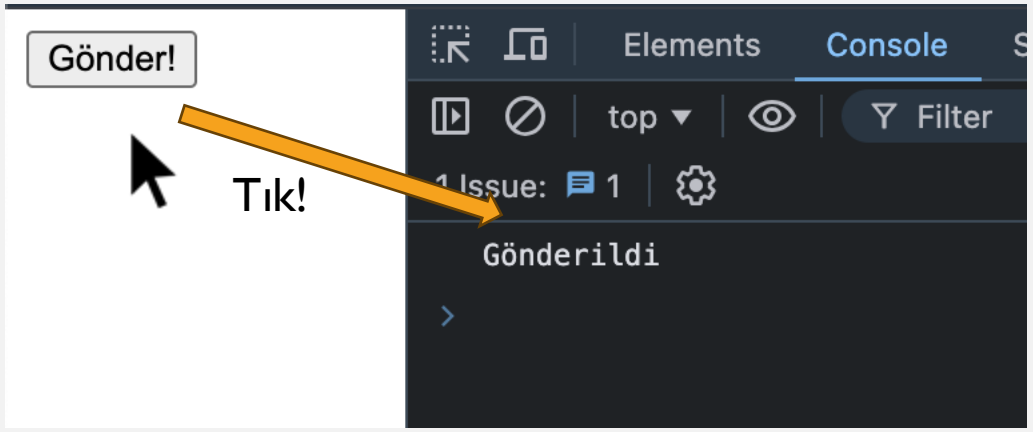
2- Javascript kodlarını sayfanın en altına eklemek ve DOM yüklemesini beklemek bu sayede DOM oluştuktan sonra JS çalışır.

```javascript
function onClick() {
  console.log('clicked');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```

EVENT LISTENER: EXAMPLE

```html
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Buton Örneği</title>
    <script src="w10_script.js" defer></script>
  </head>
  <body>
    <button>Gönder!</button>
  </body>
</html>
```

```js
function onClick() {
    console.log('Gönderildi');
}

const button = document.querySelector('button');
button.addEventListener('click', onClick);
```



DOM: CSS CHANGE

Bir elemana uygulanmış CSS özelliği de aynı şekilde manipüle edilebilir. Bunun için `classList.add` and `classList.remove` kullanabilirsiniz.

```
const image = document.querySelector('img');
// "active” isimli bir sınıf ekleyelim.
image.classList.add('active');
//"hidden" isminde bir sınıf silelim.
image.classList.remove('hidden');
```

DOM: CURRENTTARGET

EventListener ile dinleyici eklediğimiz bir elemana tekrar erişmek istediğimizde, çağırdığımız metod/fonksiyon içinde `event.currentTarget` kullanırız. Böylelikle QuerySelector ile tekrar elemana erişmek zorunda kalmayız.

```
function openPresent(event) {
  const image = event.currentTarget;
  image.src = 'pasta.png';
  image.removeEventListener('click', openPresent);
}

const image = document.querySelector('img');
image.addEventListener('click', openPresent);
```

DOM: NODE PROPERTIES

DOM öğelerinde bulunan öntanımlı bazı değerler aşağıdaki gibidir.

Özellik	Açıklama
`id`	Elemanın id/kimlik değerini metin olarak tutar.
`innerHTML`	Elemanın ham şekilde içerisinde bulunan HTML içeriğini tutar.
`textContent`	Dom öğesinin ve içerisindeki elemanların metin içeriğini tutar.
`classList`	Öğeye uygulanmış sınıfların listesini tutar.

DOM:DYNAMIC CONTENT

```
<html>
  <head>
    <meta charset="utf-8">
    <title>JS Example</title>
  </head>
  <body>
    <h1>Bir dilek tutmaya hazır mısın?</h1>
    <img src="yildiz.png" width="498px" />
  </body>
<script type="text/javascript">
  function openPresent(event) {
  const image = event.currentTarget;
  image.src = 'kayan_yildiz.gif';

  const title = document.querySelector('h1');
  title.textContent = 'Bir Dilek Tut!!!';

  image.removeEventListener('click', openPresent);
  }
const image = document.querySelector('img');
image.addEventListener('click', openPresent);
</script>
</html>
```

Ders Sayfasından Ulaşabilirsiniz: Link

DOM: ADDING ELEMENT

Dom ağacına öğe ekleyerek sayfada değişiklik yapabiliriz. Bunu için DOM'da tanımlı `createElement` ve `appendChild` metodlarını kullanabiliriz.

```
document.createElement(tag string)
element.appendChild(element);
```

`element.innerHTML='</br> '` Şeklinde bir elemana eleman ekleyebilirsniz ancak bu güvenlik riski oluşturur

Aynı şekilde DOM'dan bir eleman silebiliriz. Bunu için `remove()` metodunu kullanabilirsiniz. Kullanımı şöyledir:

```
element.remove();
```

Veya `element.innerHTML=' '` ile bir elemanın içindeki herşeyi silebilirsiniz.

ADDING ELEMENT EXAMPLE

```
function openPresent(event) {
  const newHeader = document.createElement('h1');
  newHeader.textContent = 'Bir dilek tut!';
  const newImage = document.createElement('img');
  newImage.src = 'kayan_yildiz.gif';

  const container = document.querySelector('#container');
  container.innerHTML = '';
  container.appendChild(newHeader);
  container.appendChild(newImage);
}

const image = document.querySelector('img');
image.addEventListener('click', openPresent);
```

ÖDEV: Bir önceki örneği burdaki kodu modifiye ederek remove ve appendChild metodlarıyla yapınız.

DISPLAY

Bir elemanı DOM'dan silmeden `display:none;` CSS özelliği kullanarak içerisindeki elemanlarla beraber saklayabiliriz.

```
display: block;
display: inline;
display: inline-block;
display: flex;
display: none;
```

Tık!


```html
<html>
<head>
  <meta charset="UTF-8">
  <title>Display None Demo</title>
</head>
<body>
  <h1>Parolayı Görmek için butona tıkla</h1>
  <button id="toggleButton">Göster</button>
  <p id="hiddenText" style="display: none;">Parola Şafak!</p>
  <script>
    function mesajiGoster(event) {
      const text = document.getElementById("hiddenText");
      if (text.style.display === "none") {
        text.style.display = "block";
        button.textContent = "Sakla";
      } else {
        text.style.display = "none";
        button.textContent = "Göster";
      }
    }
    const button = document.getElementById("toggleButton");
    button.addEventListener('click', mesajiGoster);
  </script>
</body>
</html>
```

ASSIGNMENT/ÖDEV

Geliştirme Görevi: Tic-Tac-Toe Oyunu
Oyunun Kuralları:

- **Oyuncu Sayısı:** Oyun, iki oyuncu arasında oynanır.
- **Oyun Alanı:** 3x3 karelik bir ızgara (toplam 9 kare).
- **Amaç:** O veya X karakterlerinden birini seçen oyuncular, sırayla kareleri işaretler.
- **Kazanan:** Karakterlerini yatay, dikey veya çapraz olarak ilk üçlü sıralayan oyuncu oyunu kazanır.
- **Alternatif Mod:** Oyunu, iki oyuncunun sırayla oynayacağı şekilde veya rastgele hamle yapan bir bilgisayara karşı oynayacak şekilde geliştirebilirsiniz.

Proje Fikirleri:

1.Mevcut Örnek Uygulamayı Modifiye Edin:

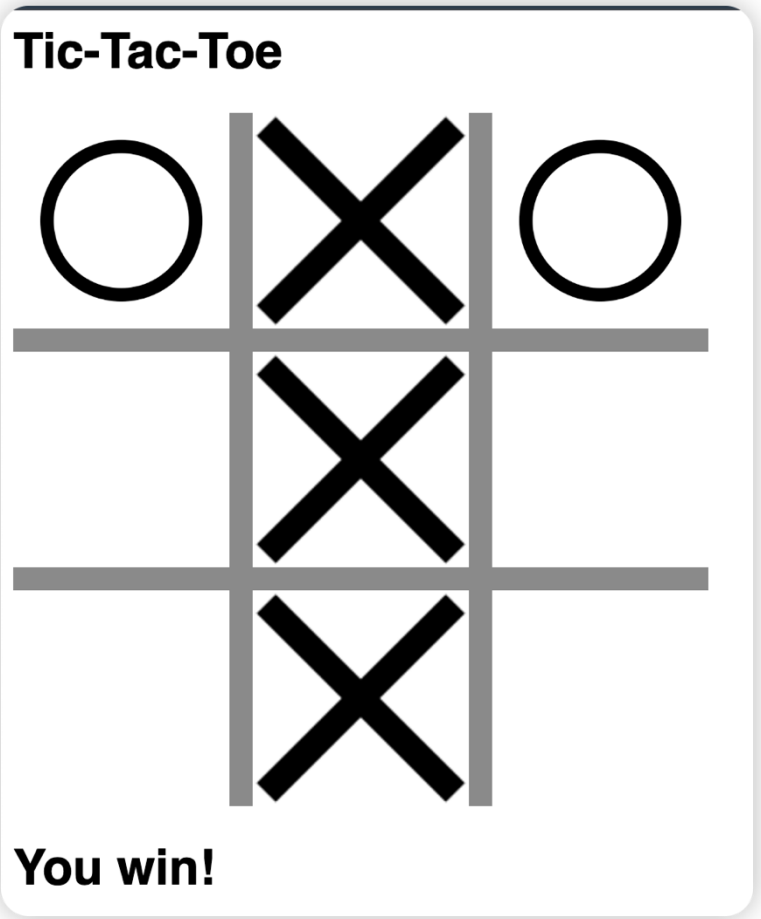
1. Ders sayfasında sunulan örnek kodu inceleyerek üzerinde değişiklikler yapabilirsiniz. Link

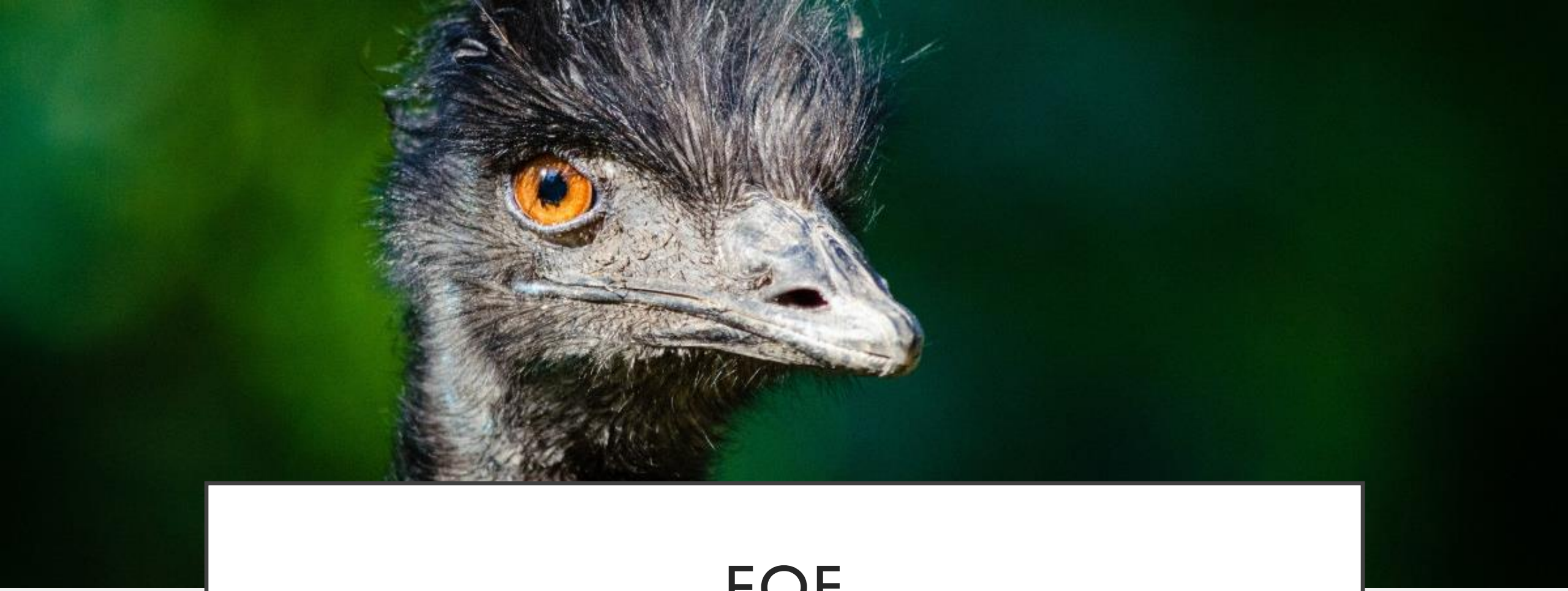
2.Kendi Sıfırdan Çözümünüzü Geliştirin:

1. Kendi mantığınız ve kodlama becerilerinizle tamamen yeni bir uygulama oluşturabilirsiniz.

3.Yapay Zeka Entegrasyonu:

1. Yapay Zeka API'leri kullanarak, rakip olarak daha zorlu bir bilgisayar oyuncusu geliştirebilirsiniz.





EOF

REFERENCES/CREDITS

1. Stepp M, Miller J, Kirst V. Web Programming Step by Step. Step by Step Publishing; 2012.
2. CS193X Web Programming Fundamentals Course Slides at Stanford Uni by Victoria Kirst
3. https://medium.com/swlh/an-introduction-to-git-and-github-22ecb4cb1256