

JavaScript Becerilerini JSFiddle Kullanarak Geliştirme

Tahmini Süre: 60 dakika

Giriş:

Bu laboratuvar, JavaScript programlama becerilerinizi geliştirmek için pratik alıştırmalar sunmak amacıyla tasarlanmıştır. Odak noktası, gerçek dünya mantıksal problemlerini çözmekte, sizi etkili kod yazmaya ve bunları JS Fiddle'da çalıştırarak çıktıyi görmeye teşvik etmektedir. Bu laboratuvarın sonunda, çeşitli senaryolar için çözümler uygulama yeteneğinizi geliştirmiş olacak ve kodlama becerilerinizde güven kazanacaksınız.

Amaç:

- JavaScript kullanarak problem çözme becerilerini geliştirmek
- Mantıksal programlar yazma ve hata ayıklama pratiği yapmak
- Döngüler, fonksiyonlar ve koşullu mantık kullanarak gerçek dünya çözümlerini nasıl uygulayacağınızı anlamak
- JSFiddle gibi platformlarda kodlama uygulamalarını güçlendirmek

Alışturma 1: Toplam satış tutarını hesaplayın

Problem:

Bir çevrimiçi mağazada çalışıyorsunuz. Göreviniz, belirli bir satış işlemleri kümesi için toplam satış tutarını hesaplayan bir JavaScript kodu yazmaktır.

Girdi ayrıntıları:

- Satış işlemlerini temsil eden bir nesne dizisi. Her nesne aşağıdaki özelliklere sahiptir:
 - item: Ürünün adı (string)
 - quantity: Satılan birim sayısı (integer)
 - price: Birim fiyatı (float)

Cıktı ayrıntıları:

- Toplam satış tutarını temsil eden tek bir sayı

Uygulama adımları:

1. En az 3 örnek nesne içeren bir satış işlemleri dizisi tanımlayın
2. Bu diziyi girdi olarak alan calculateTotalSales adlı bir fonksiyon yazın
3. Diziyi döngü ile gerekli toplam satış tutarını hesaplayın
4. Toplam satış tutarını konsola yazdırın

▼ İpuçlarını görmek için buraya tıklayın

- sales dizisi, her biri item, quantity ve price özelliklerine sahip olan satış işlemlerini temsil eden nesneleri içerir.
- sales dizisindeki her nesneyi geçmek için bir döngü kullanın.
- Her nesne için, quantity ile price'ı çarpın ve o ürün için toplamı elde edin.
- Toplamları bir değişkende biriktirerek genel satış tutarını elde edin.
- Biriktirilmiş toplamı döndürün ve console.log kullanarak görüntüleyin.

▼ Çözüm kodunu görmek için buraya tıklayın

```
const sales = [
  { item: "Laptop", quantity: 2, price: 800 },
  { item: "Monitor", quantity: 1, price: 150 },
  { item: "Mouse", quantity: 4, price: 25 }
];
function calculateTotalSales(sales) {
  let total = 0;
  for (let i = 0; i < sales.length; i++) {
    total += sales[i].quantity * sales[i].price;
  }
}
```

```
        return total;
}
console.log("Toplam Satış Tutarı:", calculateTotalSales(sales));
```

Programı JSFiddle üzerinde yazın:

- [JSFiddle](#) adresine gidin
- Kodu JavaScript bölümüne yazın
- Programı çalıştırmak için Run butonuna tıklayın ve sonuçları konsol bölümünde kontrol edin

Kodun çıktısı aşağıdaki ekran görüntüsünde gösterildiği gibi görünmelidir.

The screenshot shows a JSFiddle interface with the following sections:

- HTML:** Contains the string "<div>Hello World!</div>"
- CSS:** Contains the string "div { background-color: #f0f0f0; }"
- JavaScript:** Contains the following code:

```
1 const sales = [
2     { item: "Laptop", quantity: 2, price: 800 },
3     { item: "Monitor", quantity: 1, price: 150 },
4     { item: "Mouse", quantity: 4, price: 25 }
5 ];
6
7 function calculateTotalSales(sales) {
8     let total = 0;
9     for (let i = 0; i < sales.length; i++) {
10         total += sales[i].quantity * sales[i].price;
11     }
12     return total;
13 }
14
15 console.log("Total Sales Amount:", calculateTotalSales(sales));
```
- Run Button:** A large orange button labeled "Run".
- Output Console:** Shows the output of the console.log statement: "Total Sales Amount:", 1850.

Aliştırma 2: Bir sipariş fişi oluşturun

Problem:

Bir müşterinin siparişi için bir fiş oluşturan bir JavaScript programı yazın. Fiş, her bir ürünün adı, miktarı, fiyatı ve toplam maliyetini içermelidir.

Girdi detayları:

- Sipariş edilen ürünler temsil eden bir nesne dizisi. Her nesne şunları içerir:
 - item: Ürünün adı (string)

- quantity: Sipariş edilen miktar (integer)
- price: Birim fiyatı (float)

Çıktı detayları:

- Her ürünün detaylarını ve toplam tutarı gösteren ayrıntılı bir fiş

Uygulama adımları:

1. En az 3 örnek girişi olan sipariş edilen ürünlerin bir dizisini tanımlayın
2. Bu diziyi girdi olarak alan bir `generateReceipt` fonksiyonu yazın
3. Ürünleri döngü ile geçerek her ürünün toplamını ve genel toplamı hesaplayın
4. Fişi formatta bir dize olarak yazdırın

- ▶ İpuçlarını görmek için buraya tıklayın
- ▶ Çözüm kodunu görmek için buraya tıklayın

Programı JSFiddle'da yazın:

- [JSFiddle](#) adresine gidin
- Kodu JavaScript bölümüne yazın
- Programı çalıştırmak için `Run` butonuna tıklayın ve sonuçları konsol bölümünde kontrol edin

Kodun çıktısı aşağıdaki ekran görüntüsünde gösterildiği gibi görünmelidir.

The screenshot shows a browser-based code editor interface with tabs for HTML, CSS, and JavaScript. The JavaScript tab is active, displaying the following code:

```
1 const orders = [
2     { item: "Espresso", quantity: 2, price: 3.5 },
3     { item: "Latte", quantity: 3, price: 4.0 },
4     { item: "Cappuccino", quantity: 1, price: 4.5 }
5 ];
6
7 function generateReceipt(orders) {
8     let grandTotal = 0;
9     console.log("Receipt:");
10    console.log("-----");
11    for (let i = 0; i < orders.length; i++) {
12        const itemTotal = orders[i].quantity * orders[i].price;
13        grandTotal += itemTotal;
14        console.log(`"${orders[i].item} - Quantity: ${orders[i].quantity}, Price: ${orders[i].price}, Total: ${itemTotal}`);
15    }
16    console.log("-----");
17    console.log(`Grand Total: ${grandTotal}`);
18 }
19
20 generateReceipt(orders);
```

The CSS tab is empty, showing only a line number 1.

The browser's developer tools are open, showing the JavaScript console output:

```
>_ Console ① 7 ① 0 △ 0 ① 0 Clear console Minimize
"Receipt:"
"-----"
"Espresso - Quantity: 2, Price: $3.5, Total: $7"
"Latte - Quantity: 3, Price: $4, Total: $12"
"Cappuccino - Quantity: 1, Price: $4.5, Total: $4.5"
"-----"
"Grand Total: $23.5"
```

Alıştırma 3: Şifreleri Doğrula

Problem:

Bir liste şifreyi doğrulamak için bir JavaScript programı yazın. Bir şifre geçerli ise:

- Sadece alfanümerik karakterler (harfler ve rakamlar) içermelidir
- En az 8 karakter uzunluğunda, en fazla 20 karakter uzunluğunda olmalıdır

Girdi detayları:

- Şifrelerin (string) bir dizisi

Cıktı detayları:

- Her bir şifrenin geçerli veya geçersiz olduğunu belirten bir mesaj

Uygulama adımları:

- Örnek şifrelerden oluşan bir dizi tanımlayın
- Bu diziyi girdi olarak alan validatePasswords adlı bir fonksiyon yazın
- Şifreleri döngü ile geçerek her birini doğrulama kriterlerine karşı kontrol edin
- Her bir şifrenin geçerli veya geçersiz olduğunu kaydedin

▼ İpuçlarını görmek için buraya tıklayın

- passwords dizisi, verilen kurallara göre doğrulanması gereken stringler içerir.
- Geçerli şifreler için deseni tanımlamak üzere bir düzenli ifade kullanın. Desen yalnızca alfanümerik karakterlere izin vermelidir ve 8 ile 20 karakter arasında olmalıdır.
- Her bir şifrenin düzenli ifadeyle eşleşmediğini kontrol etmek için test yöntemini kullanın.
- passwords dizisi üzerinde döngü yapın ve her bir şifreyi regex kontrolünün sonucuna göre geçerli veya geçersiz olarak kaydedin.

▼ Çözüm kodunu görmek için buraya tıklayın

```
const passwords = ["Password123", "short", "ValidPass123", "too_long_password_example", "12345"];
function validatePasswords(passwords) {
    const regex = /^[a-zA-Z0-9]{8,20}$/;
    for (let i = 0; i < passwords.length; i++) {
        if (regex.test(passwords[i])) {
            console.log(`${passwords[i]} geçerli.`);
        } else {
            console.log(`${passwords[i]} geçersiz.`);
        }
    }
}
validatePasswords(passwords);
```

Programı JSFiddle'da yazın:

- [JSFiddle](#) adresine gidin
- JavaScript bölümüne kodu yazın
- Programı çalıştırmak için Run butonuna tıklayın ve sonuçları konsol bölümünde kontrol edin

Kodun çıktısı aşağıdaki ekran görüntüsünde gösterildiği gibi görünmelidir.

The screenshot shows a browser-based code editor interface. At the top, there's a toolbar with icons for file operations, a save button, and a "Run" button. Below the toolbar, there are three tabs: "HTML", "CSS", and "JavaScript". The "JavaScript" tab is currently selected and contains the following code:

```
1 const passwords = ["Password123", "short", "ValidPass123", "too_long_password_example", "12345"];
2
3 function validatePasswords(passwords) {
4     const regex = /^[a-zA-Z0-9]{8,20}$/;
5     for (let i = 0; i < passwords.length; i++) {
6         if (regex.test(passwords[i])) {
7             console.log(`[${passwords[i]}] is valid.`);
8         } else {
9             console.log(`[${passwords[i]}] is invalid.`);
10        }
11    }
12 }
13 validatePasswords(passwords);
```

To the right of the code editor is a "Console" window showing the output of the script's execution:

```
>_ Console ① 5 ① 0 △ 0 ① 0
  "Running fiddle"
  "Password123 is valid."
  "short is invalid."
  "ValidPass123 is valid."
  "too_long_password_example is invalid."
  "12345 is invalid."
>_
```

The console output shows that "Password123" and "ValidPass123" are valid, while "short", "too_long_password_example", and "12345" are invalid.

Ahşurma 4: Ürün stok seviyelerini takip et

Problem:

Bir çevrimiçi perakende şirketinde çalışırsınız. Göreviniz, envanterdeki çeşitli ürünlerin stok seviyelerini takip eden bir JavaScript programı yazmaktır. Program, bir ürünün stokta olup olmadığını kontrol etmeli ve uygun bir mesaj kaydetmelidir.

Girdi detayları:

- Ürünleri temsil eden bir nesne dizisi. Her nesne şunları içerir:

- product: Ürünün adı (string)
- stock: Stokta mevcut birim sayısı (integer)

Çıktı detayları:

- Her bir ürün için ürünün stokta olup olmadığını belirten bir mesaj.

Uygulama adımları:

- En az 3 örnek ürün ile ürün nesnelerinden oluşan bir dizi tanımlayın
- Bu diziyi girdi olarak alan checkStockLevels adında bir fonksiyon yazın
- Diziyi döngü ile dolaşarak her ürünün stok seviyesini kontrol edin
- Ürünün "Stokta" mı yoksa "Stokta Yok" mu olduğunu belirten bir mesaj yazdırın

▼ İpuçlarını görmek için buraya tıklayın

- products dizisi product ve stock özelliklerine sahip nesneleri içerir.
- products dizisini dolaşmak ve her ürünün stok seviyesini kontrol etmek için bir döngü kullanın.
- Stok 0'dan büyükse, "Stokta" yazdırın; aksi takdirde "Stokta Yok" yazdırın.

▼ Çözüm kodunu görmek için buraya tıklayın

```
const products = [
  { product: "Laptop", stock: 5 },
  { product: "Headphones", stock: 0 },
  { product: "Smartphone", stock: 3 }
];
function checkStockLevels(products) {
  for (let i = 0; i < products.length; i++) {
    if (products[i].stock > 0) {
      console.log(`${products[i].product} Stokta.`);
    } else {
      console.log(` ${products[i].product} Stokta Yok.`);
    }
  }
}
checkStockLevels(products);
```

Programı JSFiddle'da yazın:

- [JSFiddle](#) adresine gidin
- Kodu JavaScript bölümüne yazın
- Programı çalıştırmak için Run butonuna tıklayın ve sonuçları konsol bölümünde kontrol edin

Kodun çıktısı aşağıdaki ekran görüntüsünde gösterildiği gibi görünmelidir.

The screenshot shows a browser-based code editor interface. At the top, there's a header with icons for file operations, a title 'Untitled fiddle', and a 'Run' button. Below the header are three tabs: 'HTML', 'CSS', and 'JavaScript'. The 'JavaScript' tab is currently selected, displaying the following code:

```
1 const products = [
2     { product: "Laptop", stock: 5 },
3     { product: "Headphones", stock: 0 },
4     { product: "Smartphone", stock: 3 }
5 ];
6
7 function checkStockLevels(products) {
8     for (let i = 0; i < products.length; i++) {
9         if (products[i].stock > 0) {
10             console.log(`${products[i].product} is In Stock.`);
11         } else {
12             console.log(`${products[i].product} is Out of Stock.`);
13         }
14     }
15 }
16
17 checkStockLevels(products);
```

To the right of the code editor is a 'Console' panel. It shows the output of the `console.log` statements from the script. The output is:

- "Running fiddle"
- "Laptop is In Stock."
- "Headphones is Out of Stock."
- "Smartphone is In Stock."

Sonuç:

Bu alıştırmalar sayesinde, JavaScript kullanarak orta seviyedeki problemleri çözme pratiği yaptınız. Her görev, doğrulamadan string manipülasyonuna kadar mantıksal düşünmenin farklı yönlerine odaklandı. Programlama becerilerinizi ve güveninizi artırmak için benzer zorluklarla pratik yapmaya devam edin.

Yazar

[Rajashree Patil](#)



Değişiklik Günlüğü

Tarih	Versiyon	Değiştiren	Değişiklik Açıklaması
2024-27-12	——	Prashant Juyal	QA düzenlemeleri
2024-13-12	1.0	Rajashree Patil	İlk versiyon oluşturuldu