CODE LAB · RAPOR

Soru Bankası
Internet Programlama

HTML, CSS, JavaScript ve Next.js konularını kapsayan tüm quiz soruları; seçenekleri, doğru cevapları ve açıklamalarıyla birlikte.

TOPLAM SORU	KONU	TARİH
101	8	08 Haziran 2026

Hüseyin Tunç · CodeLab

İçindekiler

JS Temelleri (19)

Soru 1 — q-js-1

```
let x = 10;
let y = "5";
console.log(x + y);
```

A) 15

B) "105"

C) NaN

D) Hata verir

Number + String işleminde JavaScript otomatik olarak sayıyı string'e çevirir. 10 + "5" = "105" (string birleştirme).

Soru 2 — q-js-2

```
console.log(typeof null);
```

A) "null"

B) "undefined"

C) "object"

D) "boolean"

typeof null "object" döner. Bu JavaScript'in bilinen bir hatasıdır (bug), dilden kaldırılamaz çünkü eski kodları bozar.

Soru 3 — q-js-3

```
const arr = [1, 2, 3];
arr.push(4);
console.log(arr.length);
```

A) 3

B) 4

C) Hata verir (const)

D) undefined

const ile tanımlanan dizi/objenin içeriği değiştirilebilir, sadece yeniden atama yapılamaz. push(4) sonrası uzunluk 4 olur.

Soru 4 — q-js-4

```
let a = "5" - 3;
console.log(a);
```

A) "53"

B) 2

C) NaN

D) Hata verir

Çıkarma operatörü (-) string'i otomatik olarak sayıya çevirir. "5" - 3 = 2.

Soru 5 — q-js-5

```
const topla = (a, b) => a + b;
console.log(topla(3));
```

A) 3

B) NaN

C) undefined

D) Hata verir

b parametresi verilmediği için undefined olur. 3 + undefined = NaN.

Soru 6 — q-arr-1

```
const ogrenci = {
ad: "Ali",
not: 85
};
console.log(ogrenci["ad"]);
```

A) "Ali"

B) undefined

C) Hata verir

D) "ad"

Köşeli parantez (bracket notation) ile obje özelliğine erişilir. ogrenci["ad"] = "Ali".

Soru 7 — q-arr-2

```
const dizi = [10, 20, 30];
console.log(dizi[3]);
```

A) 30

B) undefined

C) null

D) Hata verir

Dizi index'i 0'dan başlar. dizi[0]=10, dizi[1]=20, dizi[2]=30. dizi[3] mevcut değil 'undefined.

Soru 8 — q-arr-3

```
const matris = [[1,2],[3,4],[5,6]];
console.log(matris[1][0]);
```

A) 1

B) 2

C) 3

D) 4

matris[1] = [3,4]. matris[1][0] = 3. İlk index satırı, ikinci index elemanı seçer.

Soru 9 — q-js-6

```
let isim = "JavaScript";
console.log(isim.toUpperCase());
```

A) "javascript"

B) "JavaScript"

C) "JAVASCRIPT"

D) Hata verir

toUpperCase() tüm karakterleri büyük harfe çevirir. "JavaScript" ’"JAVASCRIPT".

Soru 10 — q-js-7

```
let metin = "Merhaba Dünya";
console.log(metin.includes("Dünya"));
```

A) true

B) false

C) "Dünya"

D) undefined

includes() metodu string içinde aranılan ifadeyi bulursa true döner. "Dünya" metin içinde var.

Soru 11 — q-js-8

```
console.log(parseInt("42px"));
```

A) NaN

B) 42

C) "42px"

D) Hata verir

parseInt() string'in başından itibaren sayı okur, sayı olmayan kısmı yok sayar. "42px" ’42.

Soru 12 — q-js-9

```
console.log(0.1 + 0.2 === 0.3);
```

A) true

B) false

C) 0.3

D) NaN

JavaScript'te ondalıklı sayılar (floating point) tam hassasiyetle saklanamaz. 0.1 + 0.2 = 0.30000000000000004, bu yüzden === 0.3 false döner.

Soru 13 — q-js-10

```
const selamla = (isim = "Dünya") => "Merhaba, " + isim;
console.log(selamla());
```

A) "Merhaba, undefined"

B) "Merhaba, Dünya"

C) Hata verir

D) "Merhaba, "

Varsayılan parametre değeri kullanılır. isim verilmediğinde "Dünya" olur '"Merhaba, Dünya".

Soru 14 — q-js-11

```
let x = "Merhaba";
console.log(x[0]);
```

A) "Merhaba"

B) "M"

C) 0

D) undefined

String'ler dizi gibi index ile erişilebilir. x[0] ilk karakteri döner: "M".

Soru 15 — q-js-12

```
const obj = { a: 1, b: 2, c: 3 };
console.log(Object.keys(obj).length);
```

A) 3

B) 6

C) undefined

D) Hata verir

Object.keys() objenin anahtarlarını dizi olarak döner: ["a", "b", "c"]. length = 3.

Soru 16 — q-js-13

```
console.log(Boolean(""));
console.log(Boolean("0"));
```

A) false false

B) true true

C) false true

D) true false

Boş string "" falsy 'false. "0" ise boş olmayan string, truthy 'true.

Soru 17 — q-js-14

```
let dizi = [1, 2, 3, 4, 5];
dizi.splice(1, 2);
console.log(dizi);
```

A) [1, 2, 3, 4, 5]

B) [1, 4, 5]

C) [2, 3]

D) [1, 2, 3]

splice(1, 2) index 1'den başlayarak 2 eleman siler. [1, 2, 3, 4, 5] '2 ve 3 silinir '[1, 4, 5].

Soru 18 — q-js-15

```
let a = [1, 2, 3];
let b = a;
b.push(4);
console.log(a.length);
```

A) 3

B) 4

C) undefined

D) Hata verir

Diziler referans tiplidir. b = a dediğimizde aynı diziyi paylaşırlar. b'ye push yapınca a da değişir. length = 4.

Soru 19 — q-js-16

```
const kisi = { ad: "Ali", yas: 25 };
const { ad, yas } = kisi;
console.log(ad, yas);
```

A) "Ali" 25

B) undefined undefined

C) Hata verir

D) {ad: "Ali", yas: 25}

Destructuring ile objenin özellikleri aynı isimli değişkenlere atanır. ad = "Ali", yas = 25.

Döngüler (9)

Soru 1 — q-loop-1

```
for (let i = 0; i < 5; i++) {
if (i === 3) break;
}
console.log("Bitti");
```

A) 0 1 2 Bitti

B) 0 1 2 3 Bitti

C) "Bitti"

D) Hata verir

Döngü içinde console.log yok, sadece döngü sonrası "Bitti" yazdırılır. break i=3'te döngüyü kırar ama çıktıyı etkilemez.

Soru 2 — q-loop-2

```
let toplam = 0;
for (let i = 1; i <= 3; i++) {
toplam += i;
}
console.log(toplam);
```

A) 3

B) 5

C) 6

D) 10

1 + 2 + 3 = 6. Döngü i=1,2,3 için çalışır.

Soru 3 — q-loop-3

```
const arr = ["a", "b", "c"];
for (let i = arr.length - 1; i >= 0; i--) {
console.log(arr[i]);
}
```

A) "a" "b" "c"

B) "c" "b" "a"

C) "c"

D) Hata verir

Döngü sondan başa (i=2,1,0) gider. arr[2]='c', arr[1]='b', arr[0]='a' sırasıyla yazdırılır.

Soru 4 — q-loop-4

```

```

```
let sonuc = "";
for (let i = 0; i < 3; i++) {
sonuc += i;
}
console.log(sonuc);
```

A) 3

B) 6

C) "012"

D) "123"

sonuc bir string olduğundan += ile birleştirme yapılır. "" + 0 + 1 + 2 = "012".

Soru 5 — q-loop-5

```
const arr = [10, 20, 30];
arr.forEach((val, i) => {
if (i === 1) return;
console.log(val);
});
```

A) 10 20 30

B) 10 30

C) 10

D) 20 30

forEach içinde return sadece o iterasyonu atlar (break gibi değil). i=1'de return ile 20 atlanır, 10 ve 30 yazdırılır.

Soru 6 — q-loop-6

```
let i = 5;
while (i > 0) {
i -= 2;
}
console.log(i);
```

A) 0

B) 1

C) -1

D) Sonsuz döngü

i: 5 '3 '1 '-1. i=-1 olunca i > 0 false olur ve döngü biter. Sonuç: -1.

Soru 7 — q-loop-7

```
const meyveler = ["elma", "armut", "çilek"];
for (const meyve of meyveler) {
if (meyve === "armut") continue;
```

```
console.log(meyve);
}
```

A) "elma" "armut" "çilek"

B) "elma" "çilek"

C) "armut"

D) "elma"

continue "armut" iterasyonunu atlar, diğerleri yazdırılır: "elma" ve "çilek".

Soru 8 — q-loop-8

```
const obj = { a: 1, b: 2, c: 3 };
for (const key in obj) {
console.log(key);
}
```

A) 1 2 3

B) "a" "b" "c"

C) "a:1" "b:2" "c:3"

D) Hata verir

for...in döngüsü objenin anahtarlarını (key) döner: "a", "b", "c". Değerlere obj[key] ile erişilir.

Soru 9 — q-loop-9

```
let sayac = 0;
do {
sayac++;
} while (sayac > 10);
console.log(sayac);
```

A) 0

B) 1

C) 10

D) 11

do...while döngüsü koşul kontrol edilmeden önce en az 1 kez çalışır. sayac 0'1 olur, 1 > 10 false 'döngü biter. Sonuç: 1.

If/Else & Ternary (6)

Soru 1 — q-cond-1

```
let x = 0;
if (x) {
console.log("truthy");
} else {
```

```
console.log("falsy");
}
```

A) "truthy"

B) "falsy"

C) 0

D) undefined

0 falsy bir değerdir. if(0) false olarak değerlendirilir, bu yüzden else bloğu çalışır.

Soru 2 — q-cond-2

```
let yas = 18;
let sonuc = yas >= 18 ? "Ehliyet alabilir" : "Alamaz";
console.log(sonuc);
```

A) "Ehliyet alabilir"

B) "Alamaz"

C) true

D) 18

yas >= 18 true olduğu için ternary operatörün ilk kısmı ("Ehliyet alabilir") döner.

Soru 3 — q-cond-3

```
let x = "5";
switch (x) {
case 5:
console.log("sayı");
break;
case "5":
console.log("string");
break;
default:
console.log("bilinmiyor");
}
```

A) "sayı"

B) "string"

C) "bilinmiyor"

D) Hata verir

switch strict comparison (===) kullanır. "5" === 5 false, "5" === "5" true olduğu için "string" yazdırılır.

Soru 4 — q-cond-4

```
let not = 75;
let harf;
if (not >= 90) harf = "AA";
```

```
else if (not >= 80) harf = "BA";
else if (not >= 70) harf = "BB";
else harf = "CC";
console.log(harf);
```

A) "AA"

B) "BA"

C) "BB"

D) "CC"

75 >= 90 false, 75 >= 80 false, 75 >= 70 true 'harf = "BB".

Soru 5 — q-cond-5

```
console.log(null == undefined);
console.log(null === undefined);
```

A) true true

B) false false

C) true false

D) false true

== (loose equality) ile null ve undefined eşittir 'true. === (strict equality) ile tip farklı olduğundan 'false.

Soru 6 — q-cond-6

```
let x = 10;
let y = x > 5 ? x > 8 ? "büyük" : "orta" : "küçük";
console.log(y);
```

A) "küçük"

B) "orta"

C) "büyük"

D) Hata verir

İç içe ternary: x > 5 true 'x > 8 true '"büyük".

Filter & Map (8)

Soru 1 — q-fm-1

```
const sayilar = [1, 2, 3, 4, 5];
const sonuc = sayilar.filter(x => x > 3);
console.log(sonuc);
```

A) [4, 5]

B) [1, 2, 3]

C) [3, 4, 5]

D) 5

filter() koşulu sağlayan elemanları döner. x > 3 koşulunu sağlayanlar: 4 ve 5.

Soru 2 — q-fm-2

```
const isimler = ["Ali", "Ay", "Mehmet"];
const sonuc = isimler.map(x => x.length);
console.log(sonuc);
```

A) [3, 2, 6]

B) ["Ali", "Ay", "Mehmet"]

C) [3]

D) 6

map() her elemanı dönüştürür. Her ismin length'i alınır: Ali=3, Ay=2, Mehmet=6.

Soru 3 — q-fm-3

```
const sayilar = [1, 2, 3, 4];
const sonuc = sayilar
.filter(x => x % 2 === 0)
.map(x => x * 10);
console.log(sonuc);
```

A) [10, 20, 30, 40]

B) [20, 40]

C) [2, 4]

D) [10, 30]

Önce filter: çift sayılar [2, 4]. Sonra map: her biri 10 ile çarpılır '[20, 40].

Soru 4 — q-fm-4

```
const sayilar = [1, 2, 3];
const sonuc = sayilar.map(x => x * x);
console.log(sonuc);
```

A) [1, 4, 9]

B) [2, 4, 6]

C) [1, 2, 3]

D) 9

map() her elemanın karesini alır: 1*1=1, 2*2=4, 3*3=9 '[1, 4, 9].

Soru 5 — q-fm-5

```
const kelimeler = ["Ali", "Veli", "Can", "Ece"];
const sonuc = kelimeler.filter(k => k.length === 3);
console.log(sonuc);
```

A) ["Ali", "Can", "Ece"]

B) ["Veli"]

C) ["Ali", "Veli", "Can", "Ece"]

D) ["Can", "Ece"]

filter() 3 harfli olanları seçer: "Ali"(3), "Veli"(4-hayır), "Can"(3), "Ece"(3) '["Ali", "Can", "Ece"].

Soru 6 — q-fm-6

```
const arr = [5, 10, 15, 20];
const sonuc = arr.filter(x => x >= 10).map(x => x / 5);
console.log(sonuc);
```

A) [2, 3, 4]

B) [1, 2, 3, 4]

C) [10, 15, 20]

D) [2, 3]

filter: x >= 10 '[10, 15, 20]. map: her biri 5'e bölünür '[2, 3, 4].

Soru 7 — q-fm-7

```
const arr = [1, 2, 3, 4, 5];
const sonuc = arr.map(x => x * 2).filter(x => x > 6);
console.log(sonuc);
```

A) [8, 10]

B) [7, 8, 9, 10]

C) [4, 5]

D) [2, 4, 6, 8, 10]

Önce map: [2, 4, 6, 8, 10]. Sonra filter x > 6: [8, 10].

Soru 8 — q-fm-8

```
const sayilar = [3, 7, 2, 9, 1];
const enBuyuk = Math.max(...sayilar);
console.log(enBuyuk);
```

A) 1

B) 3

C) 9

D) 22

Spread (...) ile dizi elemanları Math.max'a tek tek parametre olarak verilir. En büyük değer: 9.

DOM Seçimi (7)

Soru 1 — q-dom-1

```
// HTML: <p id="mesaj">Merhaba</p>

const el = document.getElementById("mesaj");
el.textContent = "Selam";
console.log(el.textContent);
```

A) "Merhaba"

B) "Selam"

C) null

D) undefined

textContent ile içerik "Selam" olarak değiştirildi, sonra bu yeni değer okundu.

Soru 2 — q-dom-2

```
// HTML: <div class="kutu">A</div>
//       <div class="kutu">B</div>

const kutular = document.querySelectorAll(".kutu");
console.log(kutular.length);
```

A) 0

B) 1

C) 2

D) undefined

querySelectorAll ".kutu" sınıfına sahip tüm elemanları seçer. 2 adet div var, length = 2.

Soru 3 — q-dom-3

```
// HTML: <input id="isim" value="Ali">

const el = document.getElementById("isim");
el.value = "Veli";
console.log(el.value);
```

A) "Ali"

B) "Veli"

C) null

D) undefined

Input'un value'su "Veli" olarak değiştirildi, sonra bu yeni değer okundu.

Soru 4 — q-dom-4

```
// HTML: <ul id="liste">
//         <li>A</li>
//         <li>B</li>
//         <li>C</li>
//       </ul>

const liste = document.getElementById("liste");
console.log(liste.children.length);
```

A) 0

B) 1

C) 3

D) undefined

children özelliği doğrudan alt elemanları döner. 3 adet <li> var 'length = 3.

Soru 5 — q-dom-5

```
// HTML: <p id="metin">Eski Metin</p>

const el = document.getElementById("metin");
el.innerHTML = "<strong>Yeni</strong> Metin";
console.log(el.textContent);
```

A) "Yeni Metin"

B) "Yeni Metin"

C) "Eski Metin"

D) undefined

innerHTML HTML etiketlerini yorumlar. textContent ise sadece metin içeriğini döner (etiketler olmadan): "Yeni Metin".

Soru 6 — q-dom-6

```
// HTML: <button id="btn">Tıkla</button>

const btn = document.getElementById("btn");
btn.onclick = () => { console.log("A"); };
btn.onclick = () => { console.log("B"); };
// butona tıklandığında ne yazdırılır?
```

A) "A"

B) "B"

C) "A" sonra "B"

D) Hiçbiri

onclick özelliği tek bir fonksiyon tutar. İkinci atama birincinin üzerine yazar. Sadece "B" yazdırılır.

Soru 7 — q-dom-7

```
// HTML: <div id="kutu" style="color: red;">Merhaba</div>

const el = document.getElementById("kutu");
el.style.color = "blue";
console.log(el.style.color);
```

A) "red"

B) "blue"

C) ""

D) undefined

el.style.color ile inline stil "blue" olarak değiştirildi, sonra bu yeni değer okundu.

HTML (6)

Soru 1 — q-html-1

```
<!-- Hangisi sırasız (unordered) liste oluşturur? -->

A) <ol><li>Eleman</li></ol>
B) <ul><li>Eleman</li></ul>
C) <dl><li>Eleman</li></dl>
D) <list><item>Eleman</item></list>
```

A) A - ol

B) B - ul

C) C - dl

D) D - list

 (unordered list) sırasız liste oluşturur. sıralı listedir. <dl> tanım listesidir. <list> geçerli bir HTML etiketi değildir.

Soru 2 — q-html-2

```
<a href="https://google.com" target="_blank">
Tıkla
</a>

<!-- target="_blank" ne yapar? -->
```

A) Linki devre dışı bırakır

B) Yeni sekmede açar

C) Aynı sekmede açar

D) Sayfayı yeniler

target="_blank" linki yeni bir sekmede/pencerede açar. Varsayılan (_self) aynı sekmede açar.

Soru 3 — q-html-3

```
<table>
<thead>
<tr>
<th>Ad</th>
<th>Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ali</td>
<td>85</td>
</tr>
</tbody>
</table>

<!-- <th> ile <td> arasındaki fark nedir? -->
```

A) Fark yoktur

B) th başlık, td veri hücresi

C) th veri, td başlık

D) th sadece thead içinde kullanılır

<th> (table header) başlık hücresidir, kalın ve ortalı görünür. <td> (table data) normal veri hücresidir.

Soru 4 — q-html-4

```
<form>
<input type="email" id="mail" placeholder="E-posta">
<select id="sehir">
<option value="">Seçiniz</option>
<option value="ist">İstanbul</option>
<option value="ank">Ankara</option>
</select>
<button type="submit">Gönder</button>
</form>

<!-- input type="email" ne sağlar? -->
```

A) Sadece görsel fark

B) E-posta formatı doğrulama

C) Otomatik e-posta gönderir

D) Şifre gibi gizler

type="email" tarayıcının yerleşik e-posta format doğrulamasını etkinleştirir. @ işareti olmadan form gönderilemez.

Soru 5 — q-html-5

```
<!-- Hangisi semantik HTML etiketi DEĞİLDİR? -->

A) <header>
B) <nav>
C) <div>
D) <article>
```

A) header

B) nav

C) div

D) article

 genel amaçlı bir container'dır, semantik anlam taşımaz. <header>, <nav>, <article> ise içeriğin anlamını belirten semantik etiketlerdir.

Soru 6 — q-html-6

```
<a href="#bolum2">Bölüm 2'ye Git</a>

<!-- ... sayfa içeriği ... -->

<div id="bolum2">
<h2>Bölüm 2</h2>
</div>

<!-- Bu link ne yapar? -->
```

A) Yeni sayfaya gider

B) Sayfada bolum2 id'li elemana atlar

C) Hata verir

D) Hiçbir şey yapmaz

href="#bolum2" sayfa içi çapa (anchor) bağlantısıdır. Sayfadaki id="bolum2" olan elemana kaydırır.

CSS (7)

Soru 1 — q-css-1

```
/* Hangisinin önceliği (specificity) en yüksektir? */

p { color: red; }
.metin { color: blue; }
#baslik { color: green; }
```

A) p (element)

B) .metin (class)

C) #baslik (id)

D) Hepsi eşit

CSS specificity sıralaması: element (0,0,1) < class (0,1,0) < id (1,0,0). #baslik en yüksek önceliğe sahiptir.

Soru 2 — q-css-2

```
.kutu {
width: 200px;
padding: 20px;
border: 5px solid black;
box-sizing: content-box;
}

/* Kutunun toplam genişliği kaç px? */
```

A) 200px

B) 240px

C) 245px

D) 250px

content-box'ta width sadece içerik alanıdır. Toplam = 200 + (20*2 padding) + (5*2 border) = 250px.

Soru 3 — q-css-3

```
.kutu {
width: 200px;
padding: 20px;
border: 5px solid black;
box-sizing: border-box;
}

/* Kutunun toplam genişliği kaç px? */
```

A) 200px

B) 240px

C) 250px

D) 150px

border-box'ta width toplam genişliktir (padding ve border dahil). Toplam her zaman 200px kalır. İçerik alanı = 200 - 40 - 10 = 150px.

Soru 4 — q-css-4

```
.container {
display: flex;
justify-content: space-between;
}
```

```
/* justify-content: space-between ne yapar? */
```

A) Elemanları ortalar

B) Elemanlar arası eşit boşluk, kenarlar yapışık

C) Elemanlar arası ve kenarlarda eşit boşluk

D) Elemanları sola yaslar

space-between: ilk eleman sola, son eleman sağa yapışır, aradaki boşluk eşit dağılır. space-evenly ise kenarlarda da eşit boşluk bırakır.

Soru 5 — q-css-5

```
.btn {
background: blue;
transition: all 0.3s ease;
}
.btn:hover {
background: red;
transform: scale(1.1);
}

/* transition ne yapar? */
```

A) Anında değiştirir

B) 0.3 saniyede yumuşak geçiş yapar

C) Animasyonu tekrarlar

D) Sadece rengi değiştirir

transition özellik değişikliklerini belirtilen sürede yumuşak geçişle uygular. 0.3s ease ile renk ve boyut değişimi akıcı olur.

Soru 6 — q-css-6

```
.container {
display: flex;
flex-direction: column;
align-items: center;
}

/* Bu düzende elemanlar nasıl sıralanır? */
```

A) Yatay, sola yaslı

B) Yatay, ortalı

C) Dikey, sola yaslı

D) Dikey, ortalı

flex-direction: column elemanları dikey sıralar. align-items: center ise çapraz eksende (yatayda) ortalar.

Soru 7 — q-css-7

```
.spinner {
width: 40px;
height: 40px;
border: 4px solid #ccc;
border-top: 4px solid orange;
border-radius: 50%;
animation: don 1s linear infinite;
}

@keyframes don {
0% { transform: rotate(0deg); }
100% { transform: rotate(360deg); }
}

/* "infinite" ne anlama gelir? */
```

A) 1 kez döner

B) Sürekli döner

C) Çok hızlı döner

D) Tersten döner

infinite animasyonun sonsuz tekrarlanacağını belirtir. linear ise sabit hızda dönmesini sağlar.

Next.js (39)

Soru 1 — q-next-1

```
// Next.js nedir?
```

A) JavaScript'in yeni bir sürümü

B) React üzerine kurulmuş bir web framework'ü

C) Bir veritabanı yönetim sistemi

D) Sadece mobil uygulama geliştirme aracı

Next.js, React üzerine kurulmuş bir web framework'üdür. Yönlendirme (routing), sunucu tarafı render ve dosya bazlı yapı gibi özellikleri hazır olarak sunar.

Soru 2 — q-next-2

```
// app/iletisim/page.tsx

export default function Iletisim() {
return <h1>İletişim</h1>;
}

// Bu sayfaya hangi URL'den erişilir?
```

A) /

B) /iletisim

C) /page

D) /app/iletisim

Next.js dosya bazlı yönlendirme kullanır. app/iletisim/page.tsx dosyası /iletisim URL'ine karşılık gelir.

Soru 3 — q-next-4

```
"use client";

import { useState } from "react";

export default function Sayac() {
const [sayi, setSayi] = useState(0);
return <h1>{sayi}</h1>;
}

// "use client" ne işe yarar?
```

A) Bileşeni gizler

B) Bileşeni tarayıcı (client) tarafında çalıştırır; useState, onClick gibi etkileşimler kullanılabilir

C) Bileşeni sadece sunucuda çalıştırır

D) CSS dosyası yükler

Next.js'te bileşenler varsayılan olarak sunucuda çalışır. useState, useEffect, onClick gibi tarayıcı özelliklerini kullanmak için dosyanın en üstüne "use client" yönergesi eklenir.

Soru 4 — q-next-5

```
import Link from "next/link";

export default function Menu() {
return (
<>
<Link href="/hakkimda">Hakkımda (Link)</Link>
<a href="/hakkimda">Hakkımda (a)</a>
</>
);
}

// <Link> ile <a> arasındaki temel fark nedir?
```

A) Hiçbir fark yoktur, ikisi de aynıdır

B) <Link> sayfayı tamamen yeniden yüklemeden önceden belleğe alarak client-side geçiş yapar ve <a> ise sayfayı tamamen yeniden yükler

C) <Link> sadece dış sitelere gider

D) <a> her zaman daha hızlıdır

Next.js'in <Link> bileşeni sayfayı tamamen yenilemeden (client-side) geçiş yapar ve görünür haldeki hedef sayfaları arka planda önceden hazırlar. <a> ise sayfayı baştan yükler ve state'i sıfırlar.

Soru 5 — q-next-props-1

```
function Selam(props) {
return <h1>Merhaba {props.isim}</h1>;
}

// Kullanım:
<Selam isim="Ali" />

// Ekrana ne yazar?
```

A) "Merhaba props.isim"

B) "Merhaba {isim}"

C) "Merhaba Ali"

D) Hata verir

props bileşene dışarıdan veri geçirmek için kullanılır. JSX içinde {props.isim} ifadesi "Ali" değerini ekrana basar.

Soru 6 — q-next-state-1

```
"use client";
import { useState } from "react";

export default function Sayac() {
const [sayi, setSayi] = useState(0);
return (
<button onClick={() => setSayi(sayi + 1)}>
{sayi}
</button>
);
}

// useState ne işe yarar?
```

A) CSS state'ini yönetir

B) Bileşene state (durum) ekler; değer değiştiğinde yeniden render tetiklenir

C) Sunucudan veri çeker

D) Sayfayı tamamen yeniler

useState bir React hook'udur. Bileşene state ekler ve [değer, güncelleyici] döner. Güncelleyici çağrıldığında bileşen yeniden render olur.

Soru 7 — q-next-state-2

```

```

```
const [sayi, setSayi] = useState(5);

// useState(5) ifadesindeki 5 nedir?
```

A) Maksimum değer

B) Başlangıç (initial) değeri

C) State'in tipi

D) Render sayısı

useState'e verilen argüman state'in başlangıç değeridir. İlk render'da sayi = 5 olur.

Soru 8 — q-next-state-3

```
"use client";
import { useState } from "react";

export default function Form() {
const [ad, setAd] = useState("");
return (
<>
<input
value={ad}
onChange={(e) => setAd(e.target.value)}
/>
<p>Merhaba {ad}</p>
</>
);
}

// Input'a "Ali" yazılırsa ne olur?
```

A) Hiçbir şey değişmez

B) Ekranda "Merhaba Ali" yazar

C) Hata verir

D) Ekranda "Merhaba " yazar

onChange her tuş vuruşunda setAd ile state'i günceller. ad = "Ali" olunca bileşen yeniden render edilir ve "Merhaba Ali" görünür.

Soru 9 — q-next-effect-1

```
"use client";
import { useEffect } from "react";

export default function Sayfa() {
useEffect(() => {
console.log("Çalıştı");
}, []);

return <h1>Merhaba</h1>;
}
```

```
// Boş bağımlılık dizisi [] ne anlama gelir?
```

A) Effect hiç çalışmaz

B) Effect her render'da çalışır

C) Effect sadece bileşen ilk yüklendiğinde 1 kez çalışır

D) Effect sonsuz döner

useEffect'in ikinci parametresi bağımlılık dizisidir. Boş dizi [] verildiğinde effect yalnızca ilk render'da (mount) bir kez çalışır.

Soru 10 — q-next-effect-2

```
"use client";
import { useState, useEffect } from "react";

export default function Hata() {
const [sayi, setSayi] = useState(0);

useEffect(() => {
setSayi(sayi + 1);
}); // dikkat: bağımlılık dizisi YOK

return <p>{sayi}</p>;
}

// Bu kodda ne olur?
```

A) sayi 1 olur, sorun yoktur

B) Effect hiç çalışmaz

C) Sonsuz döngü oluşur: her render effect'i tetikler, effect state'i değiştirir, state değişince yeniden render olur

D) Sayfa derlenmez, hata fırlatır

Bağımlılık dizisi verilmediğinde useEffect her render'dan sonra çalışır. İçindeki setSayi state'i değiştirir, bu da yeniden render'a yol açar, render tekrar effect'i çalıştırır 'sonsuz döngü.

Soru 11 — q-next-effect-3

```
// useEffect ne için kullanılır?
```

A) Sadece CSS yazmak için

B) Yan etkileri (veri çekme, abonelik, zamanlayıcı vb.) yönetmek için

C) Sadece state tanımlamak için

D) HTML elementi oluşturmak için

useEffect, bileşen render edildikten sonra yan etkileri (API çağrısı, zamanlayıcı, abonelik gibi) çalıştırmak için kullanılır.

Soru 12 — q-next-map-1

```
```

```
export default function Liste() {
const meyveler = ["elma", "armut", "çilek"];
return (
<ul>
{meyveler.map((m) => (
<li key={m}>{m}</li>
))}
</ul>
);
}

// Ekrana kaç <li> elemanı yazılır?
```

A) 1

B) 2

C) 3

D) 0

map() dizinin her elemanı için bir <li> üretir. 3 elemanlı dizi '3 adet <li>.

Soru 13 — q-next-map-2

```
const isimler = ["Ali", "Veli"];
return (
<ul>
{isimler.map((i) => (
<li>{i}</li>
))}
</ul>
);

// Bu kodda eksik olan nedir?
```

A) return ifadesi eksik

B) Her <li> elemanına benzersiz bir key prop'u verilmeli

C) map yerine forEach kullanılmalı

D) Hiçbir eksiklik yok

JSX içinde map ile liste oluştururken her elemana benzersiz bir key prop'u verilmelidir. React, listeyi verimli güncelleyebilmek için bu key'i kullanır. Örn: <li key={i}>{i}.

Soru 14 — q-next-g-2

```
// Bir Counter (Sayaç) uygulamasında sayının değerini her
// tıklamada 1 artırmak için buton üzerinde hangi olay (event)
// tetiklenmelidir?
```

A) onKeyUp

B) onChange

C) onClick

D) onSubmit

Buton gibi elementlere fare ile tıklandığında gerçekleşecek eylemleri yönetmek için onClick olayı kullanılır. onKeyUp klavye, onChange ise form girdileri içindir.

Soru 15 — q-next-g-5

```
// Next.js (App Router) mimarisinde, dosya bazlı yönlendirme
// sistemine göre bir sayfa oluşturmak için dosya adı ne olmalıdır?
```

A) index.js

B) page.js

C) route.js

D) layout.js

Next.js App Router'da bir klasörün rota olabilmesi için içinde mutlaka page.js dosyası barındırması gerekir. route.js ise API uç noktaları için kullanılır.

Soru 16 — q-next-g-6

```
// Next.js'te veri yüklenirken kullanıcıya otomatik olarak
// bir yükleniyor arayüzü göstermek için App Router içinde hangi
// özel dosya ismi kullanılır?
```

A) spinner.js

B) error.js

C) loading.js

D) fallback.js

Next.js, bir sayfa yüklenirken React Suspense altyapısını kullanarak otomatik olarak loading.js dosyasındaki içeriği ekrana getirir.

Soru 17 — q-next-g-7

```
// Modern tarayıcılarda entegre olarak bulunan ve bir adresten (URL)
// asenkron olarak veri çekmemizi sağlayan fonksiyon hangisidir?
```

A) fetch

B) axios

C) request

D) get

fetch API'si, ağ üzerinden kaynakları almak için tarayıcılarda yerleşik olarak sunulan modern bir JavaScript fonksiyonudur. axios ise harici bir kütüphanedir.

Soru 18 — q-next-g-8

```
```

```
// Next.js'te dinamik bir rota (dynamic route) oluşturmak için
// klasör ismi nasıl tanımlanmalıdır?
```

A) :id

B) [id]

C) {id}

D) (id)

Next.js dosya sisteminde bir klasör adını köşeli parantez içine almak ([id]), o klasörün dinamik bir rota (parametre) olduğunu belirtir.

Soru 19 — q-next-g-10

```
// API'lerden veri alırken veya veri gönderirken en yaygın kullanılan,
// insan tarafından okunabilen hafif veri değişim formatı hangisidir?
```

A) XML

B) HTML

C) JSON

D) TXT

JSON (JavaScript Object Notation), JavaScript nesne yapısına dayanan, günümüzde API veri alışverişlerinde endüstri standardı haline gelmiş hafif bir formattır.

Soru 20 — q-next-g-12

```
// Kullanıcı bir input alanına yazı yazarken klavyeden bir tuşu
// elinden çektiği anı yakalamak için hangi event listener kullanılır?
```

A) onKeyDown

B) onKeyPress

C) onKeyUp

D) onClick

onKeyUp olayı, kullanıcı klavyedeki bir tuşa basıp parmağını tuştan kaldırdığı anda tetiklenir. onKeyDown ise tuşa basıldığı anı yakalar.

Soru 21 — q-next-g-13

```
// React'ta props verileri hakkında aşağıdakilerden hangisi doğrudur?
```

A) Alt bileşen (child) tarafından doğrudan değiştirilebilirler.

B) Yalnızca sayılardan oluşmak zorundadırlar.

C) Üst bileşenden alt bileşene akar.

D) Kullanılabilmesi için mutlaka useEffect içinde tanımlanmalıdır.

Props verileri tek yönlü veri akışına (one-way data binding) uygundur ve alt bileşen için salt okunurdur, doğrudan manipüle edilemezler.

Soru 22 — q-next-g-15

```
// Bir web API'sine istek attıktan sonra gelen JSON yanıtını
// JavaScript nesnesine dönüştürmek için hangi metot asenkron olarak çağrılır?
```

A) response.toString()

B) JSON.stringify(response)

C) response.json()

D) response.parse()

fetch ile gelen yanıt nesnesi (Response) üzerindeki .json() metodu, gelen ham veriyi asenkron olarak ayrıştırıp JavaScript nesnesine dönüştürür.

Soru 23 — q-next-g-16

```
// app/users/[id]/page.js
// URL /users/123 olduğunda params.id değeri ne olur?
```

A) [id]

B) undefined

C) 123

D) page

URL'deki /users/ kısmından sonra gelen dinamik değer 123 olduğu için Next.js bunu yakalar ve params.id değerine eşitler.

Soru 24 — q-next-g-17

```
// React'ta bir state güncellendiğinde aşağıdakilerden hangisi gerçekleşir?
```

A) Sayfa tamamen yenilenir (F5 atılmış gibi olur).

B) Bileşen yeniden render edilir (re-render).

C) Bütün props'lar silinir.

D) Uygulama hata vererek durur.

State değişimi React'a o bileşenin arayüzünün güncellenmesi gerektiğini söyler ve DOM'u güncellemek için bileşeni yeniden render (re-render) eder.

Soru 25 — q-next-g-18

```
// Next.js App Router yapısında tüm sayfaların etrafını saran,
// header ve footer gibi ortak bileşenleri bir kez tanımlamamızı
// sağlayan özel dosya hangisidir?
```

A) page.js

B) layout.js

C) loading.js

D) global.css

layout.js, altındaki sayfaların veya alt klasörlerin ortak yerleşim planını ve şablonunu belirlemek için kullanılır, durumunu sayfa geçişlerinde korur.

Soru 26 — q-next-g-20

```
// React'ta "Counter" uygulamasında state doğrudan
// count = count + 1 şeklinde güncellenirse ne olur?
```

A) Sayım değeri başarılı bir şekilde artar ve ekranda güncellenir.

B) Değişkenin değeri arka planda değişebilir ama bileşen re-render olmadığı için ekrandaki görüntü değişmez.

C) Uygulama hemen bilgisayarı kapatır.

D) Props verileri otomatik olarak güncellenir.

State doğrudan mutasyona uğratıldığında React bu değişikliği fark edemez ve arayüzü yeniden render etmez. Bu yüzden her zaman setCount gibi güncelleyici fonksiyonlar kullanılmalıdır.

Soru 27 — q-next-g-22

```
// Aşağıdakilerden hangisi geçerli bir useState başlangıç değeri olamaz?
```

A) Bir fonksiyon tanımı

B) Hiçbiri; her türlü geçerli JavaScript veri tipi başlangıç değeri olabilir.

C) Bir dizi veya nesne

D) null veya undefined

useState; string, sayı, boolean, nesne, dizi, null, undefined veya fonksiyon dahil tüm geçerli JavaScript veri tiplerini başlangıç değeri olarak kabul eder.

Soru 28 — q-next-g-23

```
<button onClick={handleClick()}>Tıkla</button>
// Bu tanımdaki hata nedir?
```

A) onClick kelimesi tamamen küçük harflerle yazılmalıdır.

B) Fonksiyon parantezlerle çağrıldığı için buton tıklanmadan, bileşen render edilir edilmez fonksiyon doğrudan çalışır.

C) Buton elementi React'ta onClick olayını desteklemez.

D) Süslü parantez yerine çift tırnak kullanılmalıdır.

Parantez koymak fonksiyonu hemen yürütür. Doğru kullanım parantez olmadan sadece {handleClick} şeklinde fonksiyonun referansını vermektir.

Soru 29 — q-next-g-27

```
// JSON formatındaki bir veride aşağıdakilerden hangisi
// geçerli bir veri türü olarak kullanılamaz?
```

A) undefined

B) String (Metin)

C) Number (Sayı)

D) Boolean (true/false)

JSON standardında undefined tanımı yoktur. Eksik veya tanımsız değerler için null kullanılır veya anahtar nesneye hiç eklenmez.

Soru 30 — q-next-g-33

```
// React'ta bir input elementindeki metin her değiştiğinde
// bu değişimi yakalayıp state'e aktarmak için hangi event listener
// kullanılır?
```

A) onClick

B) onChange

C) onSubmit

D) onHover

onChange olayı, form elemanlarının (input, select, textarea) değeri kullanıcı tarafından her değiştirildiğinde tetiklenir.

Soru 31 — q-next-g-35

```
useEffect(() => {
// ...
}, [search]);

// Bu efekt ne zaman çalışır?
```

A) Sadece sayfa ilk yüklendiğinde

B) İlk yüklendiğinde VE search değişkeninin değeri her değiştiğinde

C) Hiçbir zaman çalışmaz.

D) Sadece search değişkeni sıfırlandığında

Bağımlılık dizisine yazılan elemanlar React tarafından izlenir. İlk yüklemeden sonra, bu dizideki herhangi bir eleman güncellendikçe efekt fonksiyonu yeniden yürütülür.

Soru 32 — q-next-g-37

```
```

```
// Aşağıdaki kod parçalarından hangisi başarılı bir şekilde
// 1'den başlayan bir sayaç durum (state) yapısı tanımlar?
```

A) const count = 1;

B) const [count, setCount] = useState(1);

C) const [setCount, count] = useState(1);

D) const count = useState(1);

Doğru useState kullanımı dizi parçalama yöntemiyle ilk eleman olarak değişkeni, ikinci eleman olarak güncelleyici fonksiyonu alır. Parantez içi de başlangıç değeridir.

Soru 33 — q-next-g-39

```
// Bir web uygulamasında API (Application Programming Interface)
// temel olarak hangi amaca hizmet eder?
```

A) Sayfaların CSS stillerini güzelleştirmek

B) Farklı yazılımların veya istemci ile sunucunun birbiriyle güvenli bir şekilde veri alışverişi yapmasını sağlamak

C) Kullanıcının bilgisayarındaki dosyaları silmek

D) İnternet hızını artırmak

API'ler uygulamalar arası iletişimi ve veri transfer köprülerini oluşturan, belirli kuralları olan standart yazılım arayüzleridir.

Soru 34 — q-next-g-41

```
// React'ta bir fonksiyonel bileşen (Functional Component)
// temelde ne döndürmek (return) zorundadır?
```

A) Hiçbir şey döndürmez.

B) JSX (HTML benzeri arayüz yapısı) veya null

C) Sadece düz bir text stringi

D) Büyük bir JavaScript nesnesi

React fonksiyonel bileşenleri ekranda ne gösterileceğini belirten JSX ifadeleri döndürür. Eğer arayüzde bir şey çizilmeyecekse açıkça null döndürülmelidir.

Soru 35 — q-next-g-49

```
// Bir JSON verisi stringe çevrildiğinde veya okunurken en dışta
// hangi karakterlerin kullanımı zorunludur?
```

A) Köşeli Parantezler [ ] veya Süslü Parantezler { }

B) Normal Parantezler ( )

C) Açılı Parantezler < >

D) Yıldız işaretleri * *

Geçerli bir JSON dokümanı en dışta ya bir nesneyi (object) temsil eden {} ya da bir listeyi (array) temsil eden [] ile sarmalanmak zorundadır.

Soru 36 — q-next-ternary-1

```
"use client";
import { useState } from "react";

export default function Profil() {
const [girisYapildi, setGirisYapildi] = useState(false);

return (
<div>
{girisYapildi ? <h1>Hoş geldin!</h1> : <h1>Lütfen giriş yap</h1>}
</div>
);
}

// JSX içindeki { ... ? ... : ... } yapısının adı ve görevi nedir?
```

A) if/else bloğudur; JSX içinde aynen kullanılabilir.

B) Ternary (üçlü) operatörüdür; koşula göre iki farklı JSX'ten birini render etmek için kullanılır.

C) switch-case yapısıdır; sadece sayıları karşılaştırır.

D) Spread operatörüdür; iki bileşeni birleştirir.

JSX içine doğrudan if yazılamaz, ifade (expression) yazılır. Ternary operatörü (kosul ? a : b) tek bir ifade olduğu için JSX'in süslü parantezleri içinde kullanılabilir ve koşullu render için en yaygın yöntemdir.

Soru 37 — q-next-ternary-2

```
const yas = 17;
const mesaj = yas >= 18 ? "Reşit" : "Reşit değil";
console.log(mesaj);

// Ekrana ne yazar?
```

A) "Reşit"

B) "Reşit değil"

C) true

D) false

yas >= 18 ifadesi 17 >= 18 olduğundan false döner. Ternary operatörü false durumda iki nokta üst üsteden sonraki değeri ("Reşit değil") seçer.

Soru 38 — q-next-func-1

```
// Aşağıdakilerden hangisi geçerli bir React functional component
// tanımıdır?
```

A) function selam() { return <h1>Merhaba</h1>; }

B) function Selam() { return <h1>Merhaba</h1>; }

C) function Selam() { <h1>Merhaba</h1>; }

D) const Selam = <h1>Merhaba</h1>;

React bileşen isimleri büyük harfle başlamalıdır (PascalCase) — küçük harfle başlayan tag'leri React HTML elementi sanır. Ayrıca bileşen JSX'i return ile döndürmelidir; sadece yazmak yeterli değildir.

Soru 39 — q-next-func-2

```
// Aşağıdaki iki tanım arasındaki fark nedir?

// 1) function Karsila({ isim }) {
//      return <h1>Merhaba {isim}</h1>;
//    }

// 2) const Karsila = ({ isim }) => <h1>Merhaba {isim}</h1>;
```

A) İkinci tanım React'ta çalışmaz, sadece birinci geçerlidir.

B) Aralarında işlevsel bir fark yoktur; ikisi de aynı bileşeni tanımlar (function declaration vs. arrow function).

C) Birinci tanım state tutamaz, ikinci tanım tutar.

D) İkinci tanım otomatik olarak Client Component olur.

React'ta bileşenler hem function declaration (function Karsila() { ... }) hem de arrow function (const Karsila = () => ...) ile yazılabilir. Her ikisi de JSX döndürdüğü sürece aynı şekilde çalışır; sadece yazım stili farkıdır.