SAYFA 1

DOM DOM Document Object Model'i (DOM) Anlamak Tarayıcınızda bir web sayfası açtığınızı hayal edin. Tarayıcının size ne göstereceğini ve nasıl görünmesi gerektiğini bulması gerekir. Bunu yapmak için önce sayfanın HTML metnini okur ve işler. Bu, bir programın kodu ayrıştırmasına benzer. Tarayıcı daha sonra belgenin yapısının bir modelini oluşturur. Bu model, web sayfasını temsil eden bir veri yapısıdır. Bu temsile Document Object Model veya kısaca DOM denir. DOM'u iç içe geçmiş kutular veya bir ağaç gibi düşünün. Tüm web sayfası büyük bir kutu gibidir (`<html>` etiketi). `<html>` kutusunun içinde `<head>` ve `<body>` gibi daha küçük kutular bulunur. `<body>` kutusu, `<h1>` ve `<p>` gibi başka kutular içerir. • Bu küçük kutular daha da fazla kutu veya sadece metin içerebilir. DOM, JavaScript programlarının web sayfasıyla etkileşime girmesinin yoludur. Bu, "canlı" bir veri yapısıdır. Bu, JavaScript kullanarak DOM'da bir şeyi değiştirdiğinizde, tarayıcının bu değişiklikleri yansıtmak için ekranda gördüğünüzü hemen güncellediği anlamına gelir. DOM'a ve Yapısına (Ağaç) Erişme DOM'un yapısı, HTML'nizin yapısını takip eder. Bir ağaç gibi organize edilmiştir. • Belgenin her parçası bu ağaçta bir 'node' (düğüm)dur. • Node'lar (düğümler), 'children' (çocukları) olarak adlandırılan diğer node'lara başvurabilir. Bir node'un, içinde bulunduğu node'a geri işaret eden bir `parentNode` (ebeveyn node) olabilir. `<html>` etiketini temsil eden en üstteki node, ağacın köküdür. Ona `document.documentElement` kullanarak erişebilirsiniz. • JavaScript, `document` adlı global bir nesne aracılığıyla bu ağaç yapısına erişmenizi sağlar. `document.head`, `<head>` elementine ve `document.body`, `<body>` elementine işaret eder. 1

SAYFA 2

DOM DOM ağacındaki node'lar (düğümler) farklı tiplerde olabilir: Element node'ları (Öğe düğümleri): Bunlar `<p>` veya `<div>` gibi HTML etiketlerini temsil eder. `nodeType` kodları 1'dir (`Node.ELEMENT_NODE` olarak da mevcuttur). Element node'larının çocukları olabilir. • Metin node'ları (Metin düğümleri): Bunlar elementlerin içindeki metin içeriğini temsil eder. `nodeType` kodları 3'tür (`Node.TEXT_NODE` olarak da mevcuttur). Metin node'larının genellikle çocukları yoktur; ağaçtaki yapraklar gibidirler. Yorum node'ları (Yorum düğümleri): HTML yorumlarını (`<-- -->`) temsil ederler. `nodeType` kodları 8'dir (`Node.COMMENT_NODE`). Ağaçta gezinmek, aralarındaki bağlantıları kullanarak bir node'dan diğerine gitmek demektir. `parentNode`: Ağaçta mevcut olanın hemen üzerindeki node. `childNodes`: Bir node'un element, metin ve yorum node'ları dahil olmak üzere tüm doğrudan çocuklarını içeren dizi benzeri bir nesne. `children`: `childNodes`'a benzer, ancak yalnızca element node'larını (tip 1) içerir. Sadece HTML etiketleriyle ilgileniyorsanız bu genellikle daha kullanışlıdır. `firstChild` ve `lastChild`: En birinci ve en son çocuk node'lara işaret eder. `previousSibling` ve `nextSibling`: Mevcut node'un yanında, aynı ebeveyne sahip olan node'lara işaret eder. DOM arayüzü neden biraz tuhaf? DOM sadece JavaScript için tasarlanmadı; birçok farklı programlama dili tarafından kullanılmak üzere oluşturuldu. Bu, bazı kısımların JavaScript'te normalde yaptığınız şeylere kıyasla biraz garip veya daha az kullanışlı gelebileceği anlamına gelir. Örneğin, `childNodes` gibi koleksiyonlar standart JavaScript dizileri değildir, bu nedenle `slice` veya `map` gibi yerleşik metotlara sahip değillerdir. (Gerekirse `Array.from()` kullanarak onları gerçek bir diziye dönüştürebilirsiniz). Belirli Elementleri Bulma Ağaçta ebeveyn/çocuk bağlantılarını kullanarak dolaşabilseniz de, elementleri doğrudan bulmak genellikle çok daha kolaydır. Ağaçtaki tam konuma güvenmek

zor olabilir çünkü HTML'nizdeki etiketler arasındaki boşluklar gibi şeyler aslında DOM'da metin node'ları oluşturur. Elementleri bulmanın yaygın yolları şunlardır: 2

SAYFA 3

DOM • `document.getElementById("birId")`: Benzersiz `id` özelliğine göre tek bir element bulur. ID'ler bir belgede benzersiz olmalıdır. • `element.getElementsByTagName("etiketAdı")`: Çağırdığınız elementin içinde (veya örneğin `document.body` üzerinde çağırırsanız tüm belgede) belirli bir HTML etiket adına (örneğin "p" veya "a") sahip tüm elementleri bulur. Dizi benzeri bir nesne döndürür. • `element.getElementsByClassName("sınıfAdı")`: `class` özelliğinde belirli bir sınıf adı (veya adları) listelenen tüm elementleri bulur. Çağrıldığı element içinde arama yapar. • `element.querySelector("cssSeçici")`: Bir CSS seçici dizesiyle eşleşen ilk elementi bulur. Seçiciler, belirli elementleri hedeflemek için CSS'de kullanılan küçük bir dildir. Örneğin, `"p"` tüm paragrafları, `".animal"` `animal` sınıfına sahip elementleri ve `"#gertrude"` `gertrude` ID'sine sahip elementi hedefler. Bu metot, `document` ve diğer element node'larında mevcuttur. İlk eşleşmeyi veya hiçbiri bulunamazsa `null` döndürür. • `element.querySelectorAll("cssSeçici")`: Bir CSS seçici dizesiyle eşleşen tüm elementleri bulur. Canlı olmayan (yani listeyi aldıktan sonra belge değişirse otomatik olarak güncellenmeyen) bir `NodeList` döndürür. Tıpkı `getElementsByTagName` gibi, bu listeyi gerçek bir diziye dönüştürmek için `Array.from()` kullanabilirsiniz. Belgeyi Değiştirme DOM'daki hemen hemen her şeyi değiştirebilirsiniz. JavaScript bu şekilde web sayfalarını etkileşimli ve dinamik hale getirir. • Node'ları kaldırma: Herhangi bir node nesnesi, onu belgeden çıkarmak için bir `remove()` metoduna sahiptir. • Node'ları ekleme: `parentElement.appendChild(newNode)`: `newNode`'u `parentElement`'in son çocuğu olarak ekler. `parentElement.insertBefore(newNode, referenceNode)`: `newNode`'u `parentElement` içine, `referenceNode`'un hemen öncesine ekler. • Önemli: Eklediğiniz node (`newNode`) zaten belgenin başka bir yerindeyse, onu yeni bir yere eklemek otomatik olarak eski yerinden kaldıracaktır. 3

SAYFA 4

DOM • Node'ları değiştirme: `parentElement.replaceChild(newNode, oldNode)`, `parentElement` içindeki `oldNode`'u `newNode` ile değiştirir. Yeni Node'lar Oluşturma Sayfaya yeni içerik eklemek için önce onun için DOM node'larını oluşturmanız gerekir. `document.createTextNode("bir metin")`: Belirtilen metin dizesini içeren bir metin node'u oluşturur. `document.createElement("tagName")`: Verilen etiket adına (örn. "p", "div") sahip yeni, boş bir element node'u oluşturur. Oluşturulduktan sonra, ona çocuklar veya nitelikler (attributes) ekleyebilirsiniz. Niteliklerle (Attributes) Çalışmak HTML elementlerinin `href`, `src`, `id` ve `class` gibi nitelikleri vardır. Birçok yaygın standart nitelik için, elementin DOM nesnesi üzerinde aynı ada sahip bir özelliği (property) doğrudan kullanarak onlara erişebilir ve değiştirebilirsiniz. Örneğin, `linkElement.href`. Kendi eklediğiniz özel nitelikler (`data-classified="secret"` gibi) veya standart bir niteliğin doğrudan bir özelliği yoksa, metotlar kullanırsınız: `element.getAttribute("attributeName")`: Bir niteliğin değerini okur. `element.setAttribute("attributeName", "newValue")`: Bir niteliğin değerini

ayarlar veya değiştirir. `class` niteliği özel bir durumdur çünkü `class`, JavaScript'te ayrılmış (reserved) bir kelimedir. Ona bir özellik olarak erişmek için `element.className` kullanırsınız. Yine de `getAttribute("class")` ve `setAttribute("class", value)` kullanabilirsiniz. Gelecekteki standart niteliklerle çakışmaları önlemek için kendi özel niteliklerinizin adlarını `data` ile başlatmak iyi bir uygulamadır. Düzen (Layout) ve Stil (Styling) (Nasıl Göründüğü) Tarayıcılar, her elementin ekranda nerede ve ne kadar büyük olması gerektiğini hesaplar. Buna düzen (layout) denir. • Paragraflar (`<p>`) ve başlıklar (`<h1>`) gibi bazı elementler genellikle kendi satırlarında gösterilir ve mevcut tüm genişliği kaplar. Bunlar `block` (blok) elementlerdir.

SAYFA 5

DOM Bağlantılar (`<a>`) veya `<strong>` gibi diğer elementler, metin içinde aynı satırda gösterilir. Bunlar `inline` (satır içi) elementlerdir. Bunu stil kullanarak değiştirebilirsiniz. Bir elementin boyutu ve konumu hakkında bilgi alabilirsiniz: • `element.offsetWidth` ve `element.offsetHeight`: Elementin piksel (ekrandaki temel ölçü birimi) cinsinden toplam genişliğini ve yüksekliğini verir. • `element.clientWidth` ve `element.clientHeight`: Kenarlıklar hariç, elementin içindeki boşluğun genişliğini ve yüksekliğini verir. • `element.getBoundingClientRect()`: Elementin ekranın sol üst köşesine göre hassas konumunu verir. Önemli: Düzeni hesaplamak çaba gerektirir. Tarayıcılar bunu çok sık yapmaktan kaçınmaya çalışır. JavaScript kodunuz sürekli olarak DOM'u değiştirir ve ardından düzen bilgisi (`offsetHeight` gibi) isterse, tarayıcının düzeni tekrar tekrar hesaplaması gerekir, bu da kodunuzun çok yavaş çalışmasına neden olabilir. Stil (Styling), elementlerin renk, yazı tipi, boyut vb. gibi görsel görünümlerini kontrol eder. Bu genellikle CSS (Cascading Style Sheets) kullanılarak yapılır. • Stiller, HTML'deki `style` niteliği kullanılarak doğrudan bir elemente uygulanabilir. Örneğin, `<a style="color: green;">`. Bir `style` niteliği, noktalı virgülle ayrılmış bir veya daha fazla bildirim (`color: green` gibi) içerir. • JavaScript, bir elementin stillerini doğrudan `style` özelliği aracılığıyla kontrol edebilir. Bu özellik, tek tek stil özelliklerini ayarlayabileceğiniz bir nesnedir. Örneğin, `element.style.color = "red"`: • CSS'de tire içeren özellik adları (`font-family` gibi) JavaScript'te `camelCase` (`element.style.fontFamily`) olur. Ayrıca HTML'de bir `<style>` etiketi içinde veya ayrı CSS dosyalarında stil kuralları tanımlayabilirsiniz. Bu kurallar, seçiciler (tüm paragraflar için `p`, o sınıfa sahip elementler için `.my-class` veya o ID'ye sahip element için `#my-ld` gibi) kullanarak elementleri hedefler. CSS'deki 'cascading' (basamaklı) kelimesi, aynı elemente birden fazla stil kuralının uygulanabileceği ve tarayıcının hangisinin kazanacağına karar vermek için kuralları olduğu (özgüllük ve sıraya göre) anlamına gelir. `style` niteliği kullanılarak doğrudan bir elemente ayarlanan stiller en yüksek önceliğe sahiptir. 5

SAYFA 6

DOM `display` stil özelliği güçlüdür; bir blok elementi satır içine, bir satır içi elementi bloğa dönüştürebilir veya bir elementi tamamen gizleyebilir (`display: none`). Gizlemek kullanışlıdır çünkü daha sonra kolayca tekrar gösterebilirsiniz. Kısacası DOM, tarayıcınızın web sayfanızın yapısı ve içeriğinin bir ağaç gibi organize edilmiş dahili haritasıdır. JavaScript

bu haritayı okuyabilir, içinde belirli noktaları bulabilir ve değiştirebilir; bu da kullanıcının ekranda gördüğünü otomatik olarak günceller. Ayrıca, CSS stil sistemine bağlanan `style` özelliğini kullanarak elementlerin nasıl görüneceğini de kontrol edebilirsiniz. 6