İçindekiler

JavaScript'e Giriş

Bakalım JavaScript nedir, ne yapılır ve hangi teknolojilerle birlikte çalışır.

JavaScript Nedir?

JavaScript, ilk başta *"web belgelerine canlılık"* getirmek için oluşturulmuştur.

Bu dilde yazılan kod kümelerine betik denir. Doğrudan HTML kodu içerisine yazılıp sayfa yüklendiğinde doğrudan çalışabilir.

Komutlar herhangi bir derleme ve hazırlığa gereksinim duymadan doğrudan çalışırlar.

Bu yönden bakınca JavaScript diğer dillere kıyasla oldukça farklıdır.

Neden JavaScript?

JavaScript ilk yazıldığında, başka bir adı vardı: "LiveScript". Ancak Java dili o dönemlerde çok ünlü olduğundan dolayı yeni bir dil ve "küçük kardeş" gibi görünmesi açısından JavaScript olarak değiştirildi.

Ancak JavaScript gelişerek kendince yönergeleri ECMAScript olan bağımsız bir dil haline geldi. Şu anda Java ile hiçbir ilgisi bulunmamaktadır.

Günümüzde JavaScript yalnızca ağ tarayıcıda değil, sunucuda veya JavaScript motoru olan her yerde çalışmaktadır.

Tarayıcılar bu JavaScript motoru gömülü bir biçimde gelirler. Bu ayrıca "JavaScript sanal makinesi" olarak da adlandırılır.

Bu JavaScript motorlarından bazıları şunlardır;

- V8 – Chrome ve Opera.
- SpiderMonkey – Firefox.
- Internet Explorer'ın "Trident", "Chakra" takma adlı motorlarının yanında Microsoft Edge için "ChakraCore" adında ayrı bir motoru bulunmaktadır. Safari ise "Nitro", "SquirrelFish" ve "SquirrelFish Extreme" gibi takma adlarla adlandırılan JavaScript motorunu kullanmaktadır.

Yukarıdaki terimleri aklınızda tutarsanız iyi olur, çünkü ileride şu tür tümcelerle karşılaşabilirsiniz: "V8'de A özelliğinin altyapısı", "Bu özelliğin altyapısının Chrome ve Opera'da bulunduğunu anlamanız gerekir."

JavaScript Motoru Nasıl Çalışır?

Motorlar çok karmaşık yapılardır. Ancak kolay ögelere dayanırlar.

1. Eğer bu motor tarayıcıya gömülmüş ise yazılan JavaScript kodlarını ayrıştırır.
2. Sonra bu kodları makine diline çevirir.
3. Makine bu kodları çok hızlı bir biçimde çalıştırır.

Motor bu sürecin her bir adımında iyileştirme yapar. Hatta derlenmiş ve çalışır durumda bulunan kodlardaki veri yapılarını inceler ve bunları iyileştirerek daha hızlı duruma getirir. Sonuç olarak yazılan bu kodlar çok hızlı bir biçimde çalışır.

Tarayıcı içerisindeki JavaScript neler yapabilir?

Günümüz JavaScript'i "güvenli" bir programlama dilidir. Düşük düzeydeki diller gibi bellek veya işlemciye doğrudan erişim sağlamaz. Tarayıcı için olduğundan dolayı böyle bir şeye gereksinim duymaz.

JavaScript'in yapabilecekleri büyük bir oranda ortama dayanır. Örneğin Node.JS, JavaScript işlevleri ile dosyaları okuma, yazma veya ağ üzerinden isteme işlemlerini yapabilir.

Tarayıcı içerisindeki JavaScript ise web sayfasında görsel değişikliklere ve kullanıcı ile sunucu arasındaki etkileşimle ilgili her şeyi yapabilir.

Örneğin tarayıcı içerisindeki JavaScript şunları yapabilir:

- Sayfaya yeni HTML kodları ekleme veya öncekileri değiştirme, stilleri değiştirme veya ekleme.
- Kullanıcının eylemlerine karşılık verme. Tıklama veya fare imlecinin hareketine göre işlem yaptırabilme.
- Ağ üzerinden talep gönderebilme. Dosya yükleme veya indirebilme (buna AJAX ve COMET teknolojileri denir)
- Tarayıcıdaki çerezleri silme, ekleme veya düzeltme işlemlerinin yapılması. İleti gösterilmesi.
- Kullanıcı tarafında verilerin saklanması ("local storage")

Tarayıcı içerisinde bulunan JavaScript ne yapamaz?

Tarayıcı içerisinde bulunan JavaScript kullanıcı güvenliği amacıyla sınırlandırılmıştır. Amaç zararlı web sitelerinin özel bilgilere erişip kullanıcıya zarar vermesini engellemektir.

Bu engellemeleri şu biçimde sıralayabiliriz :

- Web sayfasında çalışan JavaScript dosyalara erişim sağlayamaz, saklama alanınızda bulunan programları kopyalayamaz veya çalıştıramaz. İşletim sisteminizin fonksiyonlarına doğrudan erişimi yoktur.

Günümüz tarayıcıları dosyalarla çalışmanıza izin verebilir. Ancak bu izin oldukça sınırlıdır. Örneğin, yalnızca dosyayı tarayıcıya taşıyıp bırakabilirsiniz veya <input> kullanarak dosyayı seçebilirsiniz.

Her zaman kullanıcıyla kamera veya mikrofon vasıtasıyla veya diğer aygıtlar aracılığıyla etkileşime geçebilirsiniz. Ancak kullanıcının kesin iznini almanız gerekir. Dolayısıyla bir web sayfası JavaScript ile gizliden sizin web kameranızı izleyemez veya çevrenizde bulunanlar hakkında bilgi alamaz. NSA

- Farklı sekmeler birbiri ile iletişime geçemez ve bilgi alışverişi yapamazlar. Bazı sitelerde aynı sekmeler iletişimde bulunabilir, örneğin bir sekmeden JavaScript ile diğer sekmeyi açabilirsiniz. Bu durumda bile, bir sayfa diğerinden farklı alan adı, kural veya kapılarda ise erişemez.

Bu olaya "Same Origin Policy" (Aynı kaynak kuralı) denir. Bunu çözmek için *her iki sayfa* özel bir JavaScript kodu ile birbirlerini onaylamalıdır. Bu engellemeler yine kullanıcının güvenliği içindir. Kullanıcının açtığı http://example.com sitesi diğer sekmede bulunan http://example.org sitesinden bilgi çalamamalıdır.

- JavaScript kolayca bulunduğu sayfadan veri alabilir. Ancak başka site veya alan adlarından veri alması sorunludur. Olanaklı olmasına karşın her iki yanın onayı gereklidir. Yine, bunun nedeni güvenlik sınırlarıdır diyebiliriz.

Bu sınırlar, tarayıcı dışında kullanıldığında ortadan kalkar. Örneğin, sunucular daha geniş yetkilere sahiptir.

JavaScript'i eşsiz yapan nedir ?

JavaScript'i eşsiz yapan en az 3 neden vardır:

- HTML/CSS ile tamamen bütünleşik çalışması.
- Basit şeyler basitçe yapılır.
- Tüm önemli tarayıcılarda çalışır ve ön tanımlı olarak etkindir.

JavaScript'ten başka bu üç özelliği taşıyan hiçbir tarayıcı teknolojisi yoktur.

JavaScript'in eşsiz olma nedeni budur ve bu yüzden web sayfaları geliştirmekte kullanılan en yaygın araçtır.

Yeni bir teknolojiyi öğrenmeye başlarken, sunacağı avantajlar için öngörü önemlidir. Bu sebeptendir ki, yeni diller ve tarayıcı yetkinlikleri içeren bu yönelimlere ayak uydurmalıyız.

JavaScript'e üstün diller

JavaScript'in söz dizimi ve yazımı herkese uymayabilir. Her yiğidin yoğurt yiyişi farklıdır.

Bu olağan bir durum, çünkü tasarımlar ve gereksinimler kişiden kişiye göre değişir.

Bundan dolayı yakın zamanda bir sürü yeni *transpiled* yani çevirilmiş diller türemiştir. Bu diller, çalıştırılmadan önce JavaScript'e çevriliyor. Günümüz araçları bu çeviri işini çok hızlı bir biçimde yapmaktadır. Gerçekte, doğrudan —siz yazarken bile— çevirme işini yapıp bu yeni dosyayı kullanılabilir duruma getirirler.

Bu dillere örnek vermek gerekirse:

- CofeeScript JavaScript için "şeker yazım" denebilecek bir dildir. Yazılımı daha kısadır ve daha temiz kod yazmaya yardımcı olur. Genellikle Ruby geliştiriciler bunu sever.
- Typescript durağan veri yapıları ile JavaScript yazılmasını sağlar. Karmaşık programlar geliştirmeyi kolaylaştırır. Microsoft tarafından geliştirilmiştir.
- Dart kendi başına ayrı bir dildir. Tarayıcı üzerinde veya telefon uygulamalarında kendi motoru üzerinden çalıştırılır. Google'ın tarayıcılarda JavaScript yerine Dart'ı önermiş olmasına karşın, bugünlerde JavaScript'e çeviri yapılarak kullanılmaktadır.

Bunlara daha fazla örnek eklenebilir. Yukarıdakileri bilseniz bile ne yaptığınızı tam olarak anlamak için JavaScript bilmelisiniz.

Özet

- JavaScript başlangıçta yalnızca ağ tarayıcılarında kullanılmak üzere geliştirilmiş bir dildi. Ancak günümüzde, birçok çevrede çalışabilir durumda.
- JavaScript şu anda HTML/CSS ile bütünleşik olmasından ve geniş uyumluluğundan dolayı benzersizdir.
- Birçok JavaScript'e çevirici dil bulunmaktadır. JavaScript'i iyi bir biçimde öğrendikten sonra bu dillere de bir bakmanızı öneririz.

Kılavuz ve Şartnameler

Bu kitap aslında bir *eğitim süreci*'dir. Amacı sizin kademeli olarak JavaScript öğrenmenizi sağlamaktır. Önce temellere alıştıktan sonra diğer kaynaklar üzerinde durulacaktır.

Şartname

ECMA-262 Şartnamesi JavaScript için olabilecek en derin bilgilerin bulunduğu kaynaktır. Dili tanımlar.

Fakat resmi bir dil kullanıldığından dolayı ilk seferde anlaması zordur. Eğer en güvenilir kaynak neredir diye soracak olursanız bunun cevabı **ECMA-262 Şartnamesi**'dir. Fakat her an gidip kolayca bilgi alabileceğiniz bir kaynak değildir.

Son taslağına https://tc39.es/ecma262/ adresinden erişebilirsiniz.

Daha geniş kitleler tarafından kullanılmayan yeni özelliklere ve önerilere https://github.com/tc39/proposals adresinden erişebilirsiniz.

Ayrıca, tarayıcı için geliştirme yapıyorsanız, ikinci bölümden farklı eğitimlere bakabilirsiniz.

Kılavuz

- **MDN (Mozilla) JavaScript Reference** örnek ve kılavuzların yeraldığı bir diğer kaynaktır. İstediğiniz konular derinlemesine incelemek için harika bir kaynaktır.

Buradan erişebilirsiniz: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

Google'da "MDN [term]" şeklinde aratırsanız aslında çok daha kolay erişebilirsiniz. Örneğin : parseInt'i aramak için https://google.com/search?q=MDN+parseInt kullanabilirsiniz.

- **MSDN** – Microsoft kılavuzu da çok fazla bilgi içermektedir. Buna JavaScript (JScript olarak da alandırılmakta) veya Internet Explorer gibi konular da dahildir; http://msdn.microsoft.com/ adresinden ulaşılabilir.

Ayrıca "RegExp MSDN" veya "RegExp MSDN jscript" şeklinde arayabilirsiniz.

Özelliklerin desteği

JavaScript çok hızlı gelişen bir dildir, sürekli olarak yeni özellikler eklenir.

Bunların tarayıcılarda desteklenip desteklenmediğini görmek için:

- http://caniuse.com – özellik bazlı tablo mevcuttur. Örneğin hangi JavaScript motorları cryptography özelliğini destekliyor sorusunun cevabını http://caniuse.com/#feat=cryptography adresinden bulabilirsiniz.
- https://kangax.github.io/compat-table – dil özelliğinin motorların hangisinde desteklenip hangisinde desteklenmediğini gösterir.

Bunların hepsi günlük hayatta işinize yarayacak kaynaklardır. Dil detayları ve bunların destekleri ile alakalı detaylar bulunmaktadır.

Lütfen belirli bir özelliği daha derinlemesine incelemek isterseniz bunları veya bu sayfayı hatırlayın.

Kod Düzenleyiciler

Kod düzenleyiciler programcıların evi sayılabilir.

İki farklı düzenleyici bulunmaktadır. Bunlardan ilki IDE (Integrated Development Environment – Tümleşik Geliştirme Ortamı), ikincisi ise hafif düzenleyicilerdir. Genelde çoğu kişi her iki tipten de kendine uygun bir araç seçer.

TGO (Tümleşik Geliştirme Ortamı)

TGO (Tümleşik Geliştirme Ortamı) birçok özelliği içerisinde barındıran düzenleyici olarak adlandırılabilir. Genelde "tüm proje" üzerine işlemler yapılabilecek bir "geliştirme ortamı"dır.

TGO birçok dosyayı yükleyebilir ve bu dosyalar arasında geçişleri sağlar, projenin tamamında otomatik tamamlama özelliği sunar. Versiyon kontrol sistemleri ile entegre çalışır (Örneğin git), test ortamı olarak çalıştırılabilir ve diğer "proje-seviyesinde" işlemler yapılabilir.

Eğer henüz TGO seçmediyseniz aşağıdakilere göz atabilirsiniz:

- IntelliJ Düzenleyicileri: WebStorm ön yüz geliştirmek için PHPStorm (PHP), IDEA (Java), RubyMine (Ruby) ve diğer programlama dilleri için olanları bulunmaktadır.
- Eğer .NET Geliştiricisiyseniz Visual Studio iyi bir seçimdir. Ücretsiz versiyonunu Visual Studio Community adresinden indirebilirsiniz.
- Eclipse tabanlı ürünler; örneğin Aptana ve Zend Studio.
- Komodo IDE ücretsiz olan versiyonu sisteminize pek yük olmaz Komodo Edit.
- Netbeans.

Yukarıda bahsedilen tüm TGO'lar Windows ve macOS işletim sistemlerinde çalışmaktadır. Visual Studio haricindekiler ise Linux üzerinde de çalışabilmektedir.

Çoğu ücretli olmasına rağmen deneme süresi mevcuttur. Bahsedilen TGO'ların nitelikli bir geliştiricinin maaşına kıyasla görmezden gelinebilecek kadar az olduğu söylenebilir. Bundan dolayı size en uygun olanını seçmelisiniz.

Hafif Düzenleyiciler

"Hafif Düzenleyiciler" TGO'lar kadar güçlü olmasa da hızlı ve basittirler.

Genelde bir dosyayı hızlıca açıp düzenleme amacıyla kullanılırlar.

"Hafif Düzenleyici" ile TGO arasındaki ana fark TGO'nun proje seviyesinde çalışması ve daha fazla dosyayı başlangıçta yüklemesi, analiz etmesidir. Eğer tek dosya üzerinde çalışacaksanız "hafif düzenleyiciler" daha hızlı çalışacaktır.

Pratikte hafif düzenleyiciler birçok eklenti ile klasör bazında yazım, otomatik tamamlayıcı özelliklerine erişebilirler. Bundan dolayı TGO ile hafif düzenleyici arasındaki sınır çok belirgin değildir.

Aşağıdaki hafif düzenleyiciler ilginizi çekebilir:

- Visual Studio Code (tüm işletim sistemlerinde çalışır, ücretsiz).
- Atom (tüm işletim sistemlerinde çalışır, ücretsiz).
- Sublime Text (tüm işletim sistemlerinde çalışır, ücretli).
- Notepad++ (sadece Windows'ta çalışır, ücretsiz).
- Vim ve Emacs gibi düzenleyiciler de oldukça iyidir fakat öğrenme süresi diğerler hafif düzenleyicilere göre daha uzundur.

Geliştirici Konsolu

Kod yazmak hataya meyilli bir iştir. Bu süreç içerisinde muhtemelen hatalar yapacaksınız. Pardon! Kesinlikle hata yapacaksınız, en azından robot değilseniz.

Kullanıcı, tarayıcıda meydana gelen hataları göremez. Eğer yazdığınız kodda bir yanlışlık varsa, hatalı kısımları göremez ve bunu düzeltemezsiniz.

Hataları ve diğer kullanışlı bilgileri görebilmek için tarayıcılara entegre edilmiş "Geliştirici Araçlarını" kullanmalısınız.

Genelde geliştiriciler Chrome veya Firefox'a yoğunlaşmaktadırlar çünkü ikisinin de geliştirme aracı çok iyidir. Diğer tarayıcılar da geliştirme araçları sunarlar, bazen daha farklı özelliklerle bile olsa genelde amaçları Chrome veya Firefox'u yakalamaktır. Bundan dolayı çoğu kişi "favori" tarayıcıya sahiptir ve eğer tarayıcı tabanlı bir problemle karşılaşırsa diğer tarayıcıyı kontrol eder.

Geliştirici araçları gerçekten güçlüdür ve birçok farklı özelliği bünyesinde barındırır. Öncelikle bu araçların nasıl açılacağını ve hataları nasıl inceleyeceğimizi göreceğiz. Tabi bunlar için JavaScript kodları da çalıştıracağız.

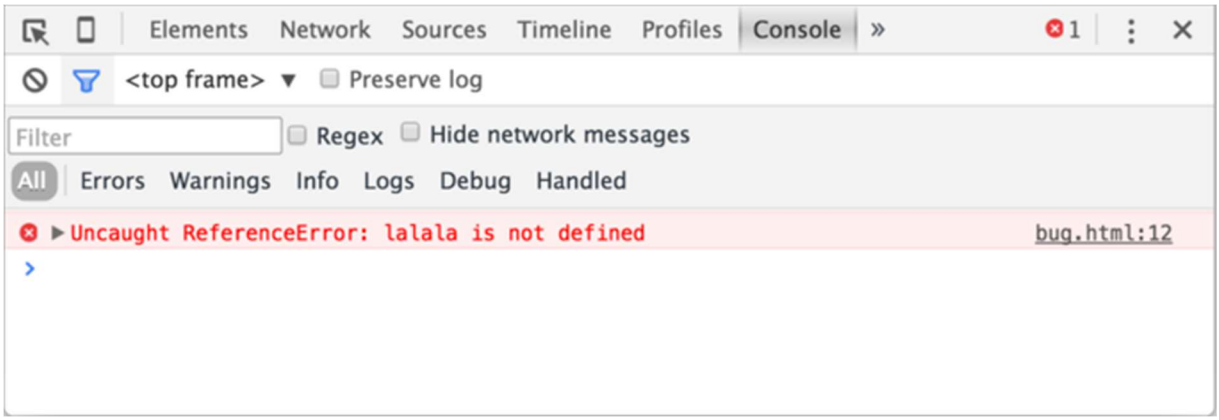
Google Chrome

bug.html sayfasını açın.

Bu sayfada bulunan JavaScript kodunda bir hata var. Kullanıcı bunu göremiyor, öyleyse geliştirici araçlarından bakıp bu hatayı tanımlayabilirsiniz.

F12'ye veya Cmd+Opt+J'ye basarak geliştirici araçlarını açabilirsiniz.

Geliştirici araçları konsol paneliyle açılacaktır. Aşağıdaki ekranda ilk hatanızı göreceksiniz:



Chrome'un geliştirme aracı versiyona göre değişiklik gösterecektir. Fakat genel hatları itibariyle şu anda gördüğünüze benzeyecektir.

- Konsol panelinde kırmızı renk ile hatayı görebilirsiniz. Bu durumda kodunuz bilinmeyen "lalala" komutunda hata vermiş.
- Sağ tarafında hatanın hangi satırda olduğunu görebilirsiniz. Bu alan tıklanabilirdir. Şu anda hata bug.html:12'de bulunmaktadır.

Hatanın altında > sembolünü görebilirsiniz. Bu, "komut satırı"'nı işaret eder. Komutunuzu yazdıktan sonra Enter'a basarak o satırdaki komutu çalıştırabilirsiniz. Birden fazla satır kod yazabilmek için ise Shift+Enter tuş kombinasyonunu kullanabilirsiniz.

Başlangıç için hataları görmek yeterli olacaktır. Daha sonra geliştirme aracını Chrome ile Hata Ayıklama bölümünde derinlemesine öğreneceksiniz.

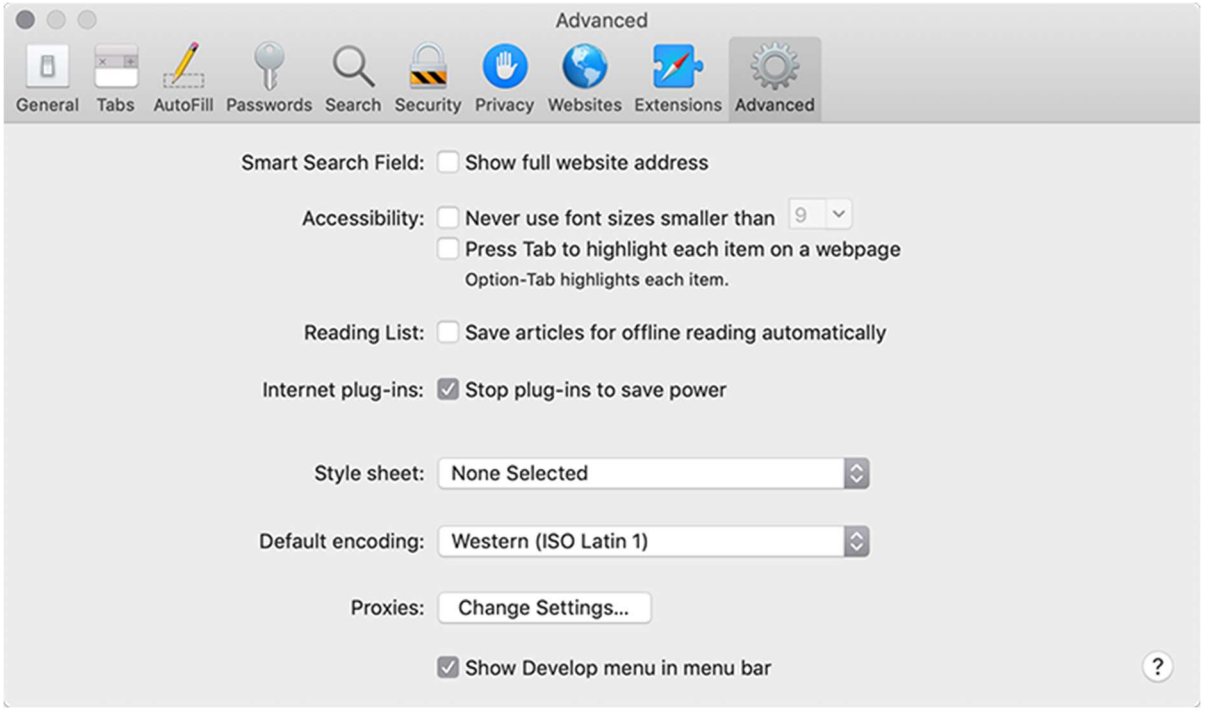
Firefox, Edge ve diğerleri

Çoğu tarayıcı geliştirme aracını F12 tuşu ile açar.

Görüntü ve kullanım olarak çoğu birbirine benzer. Bir tanesini öğrendiğinizde diğerine geçişiniz oldukça kolay olur.

Safari

Safari (sadece macOS için desteklenmektedir) biraz özeldir. Geliştirici araçlarını kullanabilmek için önce "Geliştirici Menüsü"nü aktif hale getirmeniz gerekmektedir. Bunun için özellikler sayfasını açıp "Gelişmiş" panelinden aşağıdaki gibi "Show Develop menu in menu bar"ı işaretlemelisiniz.



Bu işlemi yaptıktan sonra Cmd+Opt+C ile geliştirici konsolunu açıp kapatabilirsiniz. Ayrıca dikkat ederseniz üst menüde "Develop" adında yeni bir başlık göreceksiniz. Buradan da birçok komutu çalıştırabilirsiniz.

Multi-line input

Genelde konsol ekranında Enter yaparsanız bulunduğu satırı çalıştırır. Birden fazla satırı yazmak istiyorsanız Shift+Enter kullanabilirsiniz.

Özet

- Geliştirici araçları hataları görmenizi, komutları çalıştırmanızı, değişkenleri takip etmenizi sağlar.
- Windows işletim sisteminde f12 tuşu ile açılır (Çoğu tarayıcıda bu tuş çalışır). macOS işletim sistemi için ise Google Chrome: Cmd+Opt+J ile Safari ise: Cmd+Opt+C tuşu ile açılır (Safari'de geliştirici modunu açmanız gerekmekte).

Artık çalışma ortamınızı da ayarladığınıza göre JavaScript öğrenmeye başlayabilirsiniz.