WEB STANDARTLARINI KULLANARAK GELİŞTİRMEK

ÖNERİLER VE EN İYİ UYGULAMALAR

İçerik

1. Giriş

Bu döküman, web standartlarını kullanmanın, geliştiricinin daha az zaman ve para harcayarak üretebildiği yöntemle ve ziyaretçilerin daha iyi bir deneyim yaşayacağı web sayfaları oluşturmanızı nasıl ve neden sağlayacağını açıklamaktadır. Bunun yanısıra, olabildiğince çok kişi tarafından erişilebilir, yüksek kalitede web sayfaları üretmenin diğer yöntemlerini, prensiplerini ve en iyi uygulamalarını tartışmaktadır.

2. Tarihçe

Doksanların ikinci yarısında internet ve web ana akım haline geldiğinde, tarayıcı üreticileri, henüz, CSSi (Cascading Style Sheets) tasarımcıların bir HTML dökmanının sunumunu kontrol edebilmek amacıyla kullanabilmelerine yetecek kadar geliştirmemişlerdi. CSS Level 1'in belirtiminin

(specification) 1996'da, CSS Level 2'nin belirtiminin ise 1998'de yayınlandığı düşünülürse, bu geliştirmenin o zamanki eksikliği bir miktar anlaşılabilir.

Tarayıcılardaki CSS desteğinin eksikliği, basılı malzeme üzerinde çalışırken mümkün olabilen denetim düzeyine alışmış grafik tasarımcıların beklentileri ile birleştiğinde, HTML, web sayfalarının görünüşünü kontrol etmeyi olası hale getiren her türlü yöntemle suistimal edildi. Bu suistimallere en büyük örnek, tasarımcılar, `border="0"` özelliğini kullanarak tabloların sınırlarını gizleyebildiklerini, böylece sayfa düzenini kontrol edebilecekleri görünmez bir ızgara oluşturabildiklerini keşfettiklerinde oluşan akımdır. Diğer bir örnek ise yine sayfa düzenini kontrol etmeye yardımcı olan, saydam yani görünmez aralık (spacer) GIF'lerinin kullanımıdır.

HTML'in hedefi hiçbir zaman döküman görünüşünü kontrol etmek olmadığından, kısa yollar, geçersiz kodlar ve üreticiye özel elemanlar (etiket/tag), özellikler (attribute) kullanıldı. Geçerleme çok az kişi tarafından bilinen ya da kullanılan birşeydi. Bu tarz kod için en açıklayıcı isim etiket (tag) çorbasıdır.

Tarayıcıların yeni sürümleri çıktıkça, CSS desteği geliştirildi ve genişletildi ama bu gelişme ve genişlemenin hızı olması gerektiği kadar değildi. Tarayıcı üreticilerinin CSS'i geliştirmek konusunda bu kadar yavaş olmalarına rağmen, çok sayıda insanın makul düzeyde CSS desteği olan tarayıcıları kullanıyor olduğu öyle bir noktaya geldik ki artık HTML'i sadece gerçek amacıyla, yani bir dökümanın görünüşünü değil **yapısını** tanımlamak amacıyla **kullanmamamız** için hiçbir neden kalmadı. Bu yüzden, artık özellikle bu amaçla tasarlanmış olan CSS'i kullanabiliriz.

3. Web standartları

Web standartları nedir?

Web standartları, W3C ve diğer standart oluşumları tarafından bir araya getirilmiş olan, web tabanlı içeriğin yaratılmasında ve çözümlenmesinde kullanılan teknolojilerdir. Bu teknolojiler, gelecekte de doğru biçimde görüntülenebilecek dökümanların Web'de yayımlanmasını ve bu dökümanların olabildiğince çok insan tarafından erişilebilir olmasını sağlamak üzere tasarlanmışlardır.

Yapısal diller

- HTML (Hypertext Markup Language) 4.01
- XHTML (Extensible Hypertext Markup Language) 1.0
- XHTML 1.1
- XML (Extensible Markup Language) 1.0

Sunum dilleri

- CSS (Cascading Style Sheets) Level 1
- CSS Level 2 revision 1
- CSS Level 3 (in development)
- MathML (Mathematical Markup Language)
- SVG (Scalable Vector Graphics)

Nesne Modelleri

- DOM (Document Object Model) Level 1
- DOM Level 2
- DOM Level 3 Core

Betik (script) dilleri

- ECMAScript 262 (JavaScript'in standart hale getirilmiş biçimi)

Bu döküman, yapı için XHTML 1.0 Strict, sunum için CSS Level 1 and Level 2 ve betik (sript) yazma için ECMAScript 262 (betik yazma için çok fazla örnek olmasa da) üzerinde durmaktadır.

Bir dökümanın web standartlarına bağlı kalınarak oluşturulduğu söylendiğinde döküman yukardaki teknolojiler kullanılarak üretilmiş olmanın yanısıra aşağıdaki özellikleri de sergiliyor demektir:

- geçerli XHTML'den oluşur
- sayfa düzeni için tabloları değil CSS'i kullanır
- düzgün biçimde yapılandırılmış ve semantik olarak yazılmıştır
- ve tüm tarayıcılarda çalışır.

Çevirenin notu: Yazı boyunca *semantic* kelimesi *semantik* olarak çevrilmiştir. Tam karşılığı "anlam bilimsel"dir, ancak bu yazıda "doğru, mantıklı ve olması gerektiği gibi, anlamına uygun biçimde" anlamında kullanılmaktadır.

"Tüm tarayıcılarda çalışır" cümlesinin "tüm tarayıcılarda aynı görünür" demek olmadığına dikkat edin. Bir dökümanın tüm tarayıcılarda aynı görünmesini sağlamak neredeyse imkansızdır. Yalnızca resimden oluşsa bile bir web sayfasının heryerde aynı görünmesi sağlanamaz. Web'de sunulan dökümanlara, farklı işletim sistemlerinde çalışan çok fazla sayıda değişik tarayıcı araç kullanarak değişik boyutlardaki ve kalitelerdeki monitörler aracılığıyla (ya da hiç monitörsüz), tarayıcılarının varsayılan metin büyüklüğünü ve diğer seçeneklerini değiştirmiş kullanıcılar tarafından erişilir. Bunu kabullenmek hayatınızın çok daha az karışık olmasını sağlar. Web sayfaları hazırlayan herkes, kağıt üzerinde yayın çıkaranlar ya da televizyon için filmler üretenler gibi düşünülmesi gereken önkoşulların varolduğunu bilmelidir.

Neden web standartlarını kullanmalı?

Bazı web geliştiricileri ve tasarımcıları web standartlarını kullanmaya karşı direnç gösterirler. Sık görülen nedenler şunlardır: Çok zor, Nasıl olsa her şekilde çalışıyor ve Kullandığım araçlar geçersiz kod üretiyor, ne yapayım.

Duygusal olarak tepki göstermek ve bildiğiniz, kullanırken kendinizi rahat hissettiğiniz teknikleri bırakıp yeni birşey öğrenmeye direnç geliştirmek çok kolaydır. Ancak, duruma mantıklı olarak bakarsanız web standartlarını öğrenmenin ve kullanmanın bir çok getirisi vardır. Bunlara bir kaç örnek:

- **Daha kolay geliştirme ve bakım:** Daha semantik ve yapılandırılmış HTML kullanmak başka birisi tarafından yazılmış kodu anlamayı kolaylaştırır ve hızlandırır.
- **Gelecekte ortaya çıkacak tarayıcılarla uyumluluk:** Tanımlanmış standartları kullanır, geçerli kod yazarsanız, yeni çıkacak tarayıcıların dökümanlarınızı anlamaması riskini azaltır yani dökümanlarınızı gelecek uyumlu yapmış olursunuz.
- **Web sayfalarının daha hızlı indirilmesi ve çözümlenmesi:** Daha az HTML kodu daha küçük dosya boyutu ve daha hızlı indirilme demektir. Modern tarayıcılar, sayfaları, standart moddayken, geçmiş uyumlu moddayken olduğundan daha hızlı çözümlerler.

- **Daha fazla erişilebilirlik:** Semantik HTML, yani yapının sunumdan ayrıldığı HTML, ekran okuyucuların ve alternatif tarayıcı araçlarının içeriği daha kolay algılamasını sağlar.
- **Daha yüksek arama motoru sıraları:** İçeriğin ve görünüşün ayrılması içeriğin sayfa boyutunun çok daha fazlasını temsil etmesi anlamına gelir. Semantik kodlarla birlikte bu, sayfanın arama motoru sırasını yükseltir.
- **Daha kolay uyum:** Semantik biçimde oluşturulmuş bir döküman, yalnızca farklı bir CSS dosyası ile ilişkilendirilerek yazdırma işlemi ya da el bilgisayarları veya cep telefonları gibi alternatif tarayıcı araçlara çok daha kolay uyumlu hale getirilebilir.

Web standartları, web sitesi yaratıcılarına para ve zaman kazandırır, ziyaretçilere ise çok daha güzel bir deneyim yaşatır. Bunun yanısıra, web standartları "gelecek"tir. Hali hazırda, web standartlarını kullanmıyorsanız şimdi başlamanın tam zamanıdır, aksi taktirde arkada kalma riskini almış olursunuz.

Daha fazla okuma:

- My Web site is standard! And yours?

  Web sitenizin kod kalitesini nasıl arttırabileceğiniz konusunda bir W3C dökümanı.

- Fighting for Standards

  Web Standartları Projesi'nin bildirisi.

- What are web standards and why should I use them?

  Web Standartları Projesi'nin web standartlarının ve onları kullanmanın neden iyi olduğunu derinlemesine anlatan dökümanı.

- The Business Benefits of Web Standards

  Bir şirketin web standartlarını kullanarak nasıl para kazanabileceği üstüne bir Netscape DevEdge makalesi.

- Web Standards for Business

Pazarlama, iletişim ve IT bölümlerinden proje paydaşlarına seslenen bir Web Standartları Projesi makalesi.

Geçerleme

Geçerleme, bir dökümanın, yazıldığı dilin kurallarına uyup uymadığını kontrol etme işlemidir. Bu işlemi, bir metnin yazım ve sözdizim kurallarına uyumunu kontrol etmeye benzetebilirsiniz.

Geçerleme, web sayfaları geliştirme sürecinin çok önemli bir parçasıdır. Farkedilmesi zor birçok hata geçerleme sırasında bulunur. Hatalar, yazım hataları kadar basit olabileceği gibi yanlış kullanılan bir elaman ya da özellik kadar ciddi de olabilir.

Ne yazık ki bir çok insan sayfalarını geçerlemez. Bazı insanların bundan haberleri bile yoktur, diğerleri ise bu işlemi unuturlar ve hatta bazıları bilerek ve isteyerek geçerlemeden sakınırlar. Bu durum için ağırlıkla tarayıcı üreticileri suçlanabilir. Birçok tarayıcı hata vermek yerine, geçersiz HTML kodlarını ellerinden geldiğince düzgün biçimde çözümlemeye ve yazan kişinin ne kastettiğini becerebildiği kadar kestirmeye çalışır. işte tarayıcıların bu davranışları bugün yazılmakta olan özensiz kodlara neden olmuştur. Bu tür kodlar kestirilemeyen sonuçlar ürettikleri ve web tarayıcıların hata işleme yöntemlerine güvendiklerinden dolayı sorunludurlar.

HTML ve CSS kodlarınızı geçerlememeniz için hiç bir neden yok. Aksine, bu sizin leyhinize.

Why we won't help you (Neden size yardım etmeyeceğiz?), Mark Pilgrim'in geçerlemenin avantajlarını anlattığı harika bir yazısıdır. Makale, ayrıca, yardım istemeden önce dökümanlarınızı geçerlemezseniz forumlarda ve e-posta gruplarında yardım bulmanızın neden daha zor olabileceğinden bahsetmektedir.

BBEdit ve Homesite gibi birçok HTML editörü, yerleşik geçerleme araçlarına sahiptir. Geliştirme aracınızın yerleşik geçerleme yöntemleri yoksa W3C'nin çevrimiçi erişilebilen geçerleme serviserinden faydalanabilirsiniz:

- (X)HTML: W3C MarkUp Geçerleme Servisi
- CSS: W3C CSS Geçerleme Servisi

Geçerleme araçlarının ürettikleri hata mesajlarını anlamak bir miktar ustalık isteyebilir. Hata listesinin başlarında yeralan bir hata altınaki birçok hataya neden oluyor olabilir. Bu hataya neden olan kodu düzeltir ve dökümanı yeniden geçerlerseniz hata listesini beklediğinizden fazla kısaltırsınız. Sıkça görülen hata mesajlarının açıklamalarını Black Widow Web Design'ın Common XHTML Validation Errors sayfasında bulabilirsiniz.

Kodunuzu tamamıyla geçerlenmiş hale getirmek çok iyi bir fikirdir ama bazı geçerleme hatalarından kurtulmak oldukça zor olabilir. Buna en iyi örnek sayfaya Flash ya da eklenti (plugin) isteyen başka bir içerik gömmek olabilir. Flash'la ilgili sorunlar hakkında daha detaylı bilgiye Flash Satay: Embedding Flash While Supporting Standards (Web Standartlarını Desteklerken Sayfaya Gömülü Flash Kullanmak) ve Embedding flash without <embed> (<embed> Kullanmadan Sayfanıza Flash Gömmek) sayfalarından erişebilirsiniz.

4. Yapı ve sunum

Web standartlarından sözedilirken sıkça değinilen konulardan biri yapıyı ve sunumu belirleyen kodun birbirinden ayrılmasıdır. Özellikle dökümanın semantik yapısını düşünmeye alışkın değilseniz başlangıçta yapı ve sunumu belirleyen kodun farkını anlamak zor olabilir. Ama, bunu anlamak çok önemlidir. Çünkü bu ikisi birbirinden ayrıldığında bir dökümanın sunumunu CSS ile konrtrol etmek daha kolaydır.

Yapı bir dökümanın olmazsa olmaz parçaları ve içeriğinin semantik ve yapısal kodudur.

Sunum ise içeriğe verdiğiniz stildir. Birçok durumda sunum dökümanın nasıl göründüğü ile ilgilidir ancak herkes grafik bir tarayıcı kullanmadığından bazen dökümanın nasıl duyulduğunu da belirler.

Yapıyı ve sunumu mümkün olduğunca birbirinden ayırın. İdeal olarak, sadece yapıyı ve içeriği kapsayan bir HTML dökümanınız ve sunumu kontrol etmenizi sağlayan bir de CSS dökümanınız olmalıdır.

Yapının ve sunumun birbirinden ayrılması günümüz web tasarımında yaygın değildir. Birçok web sitesinde HTML kodu hem yapısal hem de semantik olarak eksiktir.

Sayfa düzeni için tablo kullanmak

Yapıyı ve sunumu birbirinden ayırmak için dökümanın sunumunu kontrol etmek amacıyla tabloları kullanmak yerine CSS kullanmalısınız. Sayfa düzeni için tablo kullanmaya alıştığınızda bu söylediğimi yapmak size zor gelebilir ama düşündüğünüz kadar zor değildir. Bu konuda internette çok miktarda yardım bulabilirsiniz. Ayrıca size katacağı avantajları da çok fazladır. Bu yüzden bu farklı düşünüş tarzını öğrenmeye zaman ayırmaya değecektir.

Daha fazla okuma:

- Why tables for layout is stupid (Sayfa düzeni için tablo kullanmak neden aptalcadır)

  Seybold 2003'de yapılan bir sunumun slaytları.

Semantik HTML

Yapıyı sunumdan ayırmanın diğer bir önemli parçası dökümanın yapısını kodlamak için semantik HTML kullanmaktır. Dökümanın o bölümünde kullanmaya uygun yapısal anlamı olan bir xhtml etiketi varsa onun yerine başka bir şey kullanmak için hiçbir nedeniniz olamaz. Diğer bir deyişle bir HTML etiketini başka bir HTML etiketi gibi göstermek için CSS kullanmayın. Örneğin; bir başlığı kodlamak için <h1> yerine elemanını kullanmayın.

Semantik HTML kullanarak bir dökümanın değişik bölümlerinin tüm web tarayıcıları için anlamlı olmasını sağlarsınız. Bu tarayıcı, modern bir PC'deki en son grafik tabanlı web tarayıcı, CSS'i işleyemeyen eski bir tarayacı ya da Unix üzerinde metin tabanlı bir tarayıcı olsa da farketmez.

Başlıklar

Başlıkları kodlamak için başlığın seviyesine göre <h1>, <h2>, <h3>, <h4>, <h5> ya da <h6> kullanın. <h1> en üst seviye başlık için kullanılır.

Örnekler:

```
<h1>Döküman başlığı</h1>
<h2>Alt başlık</h2>
```

Döküman başlığı

Alt başlık

Paragraflar

Paragrafları kodlamak için <p> elamanını kullanın. Paragraflar arasında boşluk yaratmak için
 elemanını kullanmayın. Paragraflar arasındaki aralıklar (margin) CSS tarafından belirlenir; bu da paragrafların doğru biçimde kodlanmasını gerektirir.

Örnek:

```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing
elit. Donec risus. Ed rhoncus sodales metus. Donec adipiscing
mollis neque. Aliquam in nulla.</p>
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec risus. Sed rhoncus sodales metus. Donec adipiscing mollis neque. Aliquam in nulla.

Listeler

Birşeylerin listesini kodlayacaksanız liste yapın. XHTML'de üç farklı liste türü vardır: sırasız listeler, sıralı listeler ve tanım listeleri.

İmli listeler olarak da bilinen sırasız listeler <ul> ile başlar ve ile sonlanır. Her liste elemanı <li> elamanı içinde yazılır.

Sıralı listeler ise <ol> ile başlayıp ile sonlanır.

Tanım listeleri biraz daha farklıdır. Terimler ve anlamları/açıklamaları gibi bölümleri kodlamada kullanılırlar. Tanım listeleri <dl> ile başlayıp </dl> sonlanır. Açıklanacak her terim <dt> elemanı, terimin anlamı/açıklaması da <dd> elemanı içinde yazılır.

Örnekler:

```
<ul>
    <li>Öge 1</li>
    <li>Öge 2</li>
    <li>Öge 3</li>
</ul>
```

- Öge 1

- Öge 2
- Öge 3

```
<ol>
    <li>Öge 1</li>
    <li>Öge 2</li>
    <li>Öge 3</li>
</ol>
```

1. Öge 1
2. Öge 2
3. Öge 3

```
<dl>
    <dt>web sitesi</dt>
    <dd>Bir kuruma ya da kişiye ait birbiri ile bağlantılı  web
sayfalarının

       toplamı.</dd>
    <dt>web sayfası</dt>
    <dd>Metin, grafik ve diğer ortam bileşenlerini içerebilen,
internet

       üzerindeki temel bilgi birimi.</dd>
</dl>
```

web sitesi

Bir kuruma ya da kişiye ait birbiri ile bağlantılı web sayfalarının toplamı.

web sayfası

Metin, grafik ve diğer ortam bileşenlerini içerebilen, internet üzerindeki temel bilgi birimi.

Listenin içeriğinin alışılmış bir liste gibi görünmesini istemediğiniz de bile CSS liste kullanımını olanaklı kılar. Bağlantıların bir listesi olan yol bulma (navigation) barı buna iyi bir örnek olabilir. Bir menüyü liste gibi kullanmak menünün CSS desteklemeyen tarayıcılarda da anlamlı görünmesini sağlar.

Alıntılar

<q> elemanı, kısa, satır içi (inline) alıntılar için kullanılmalıdır. Web tarayıcıları <q> elamanının içeriğinden önce ve sonra otomatik olarak tırnak imleri ekler. Ne yazık ki, Internet Explorer bunu yapmaz ve hatta bazı durumlarda <q> elemanı erişim sorunlarına dahi neden olabilir. Bu nedenle kimileri <q> kullanmaktan kaçınmanızı onun yerine tırnak imlerini sizin eklemenizi salık verir. Satır içi (inline) alıntıları uygun sınıflarla elemanları ile kullanmak alıntıları CSS ile biçimlendirmenizi sağlar ancak bu kodun semantik olarak hiçbir değeri yoktur. <q> elemanı hakkındaki sorunlarla ilgili daha ayrıntılı bilgi için Mark Pilgrim'in The Q tag (Q etiketi) adlı makalesini okuyabilirsiniz.

Daha uzun, bir iki paragraflık alıntılar için <blockquote> elemanı kullanılmalıdır. O zaman alıntıyı biçimlendirmek için CSS kullanabilirsiniz. <blockquote> elemanının içine doğrudan metin yazılamayacağına dikkat edin – içine yazılacak metin bir elemanın içinde olmalı; örneğin <p>.

cite özelliği, alıntının kaynağını belirtmek için hem <q> ile hem de <blockquote> ile kullanılabilir. Dikkat edin; eğer satıriçi alıntılar için <q> yerine kullanırsanız cite özelliğini kullanamazsınız.

Örnekler:

```
<p>W3C'ye göre <q cite="http://www.w3.org/TR/REC-html40/
struct/text.html#h-9.2.1">İfade elamanlarının görüntüleniş biçimi
kullanıcı

aracına bağlıdır.</q>.</p>
```

W3C'ye göre İfade elamanlarının görüntüleniş biçimi kullanıcı aracına bağlıdır.

```
<p>W3C'ye göre <span class="quote">&#8220;İfade elamanlarının
görüntüleniş

biçimi kullanıcı aracına bağlıdır.&#8221;</span>.</p>
```

W3C'ye göre *"İfade elamanlarının görüntüleniş biçimi kullanıcı aracına bağlıdır."*.

```
<blockquote cite="http://www.w3.org/TR/1999/REC-html401-19991224/
struct/text.html">
    <p>&#8220;İzleyen bölümler metinleri yapılandırma konuları

       üzerinde duruyor. Metin görünümleri ile ilgili elemanlar
(hizalama

       elemanları, font elemanları, biçem sayfaları, vb.) belirtimde

       bulunmaktadır. Karakterler hakkında bilgi için döküman karakter

       seti ile ilgili bölüme başvurunuz.&#8221;</p>
</blockquote>
```

"İzleyen bölümler metinleri yapılandırma konuları üzerinde duruyor. Metin görünümleri ile ilgili elemanlar (alignment elemanları, font elemanları, biçem sayfaları, vb.) belirtimde bulunmaktadır. Karakterler hakkında bilgi için döküman karakter seti ile ilgili bölüme başvurunuz."

Vurgu

 vurgu için, ise daha güçlü vurgu için kullanılır. Birçok tarayıcı *vurgulanmış* metni italik,**daha güçlü vurgulanmış** metni kalın görüntüler. Ancak bu bir zorunluluk olmadığından vurgulanmış metnin nasıl görüntüleneceğinden emin olmak için görünümlerini CSS ile belirlemek en doğrusu olacaktır. Yapmak istediğiniz sadece görsel bir efekt ise vurgu elemanı kullanmayın.

Örnek:

```
<p><em>Vurgulanmış</em> metin normalde italik olarak,
<strong>daha güçlü vurgulanmış</strong> metin ise kalın
görüntülenir.</p>
```

Vurgulanmış metin normalde italik olarak, **daha güçlü vurgulanmış** metin ise kalın görüntülenir.

Tablolar

XHTML tabloları sayfa düzeni için kullanılmamalıdır. Ama, tablo verilerini göstermek için tabii ki tablolar kullanılmalıdır. Veri tablolarını olabildiğince erişilebilir kılmak için tabloyu oluşturmakta

kullanılabilen birçok farklı bileşeni bilmek ve kullanmak gerekir. Bunlara örnek, tablo başlıkları (<th>), özetler (summary özelliği), ve etiketlerdir (<caption> elemanı).

Örnek:

```
<table class="example" summary="Bu tablo İsveç'teki

1999-2003 yılları arası nüfus artışını göstermektedir.">
    <caption>İsveç'teki yıllık nüfus artışı, 1999-2003</caption>
    <thead>
        <tr>
            <td> </td>
            <th scope="col">1999</th>
            <th scope="col">2000</th>
            <th scope="col">2001</th>
            <th scope="col">2002</th>
            <th scope="col">2003</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <th>Nüfus</th>
            <td scope="row">8 861 426</td>
            <td scope="row">8 882 792</td>
            <td scope="row">8 909 128</td>
            <td scope="row">8 940 788</td>
            <td scope="row">8 975 670</td>
        </tr>
        <tr>
            <th>Artış</th>
            <td scope="row">7 104</td>
            <td scope="row">21 366</td>
            <td scope="row">26 368</td>
            <td scope="row">31 884</td>
            <td scope="row">34 882</td>
        </tr>
    </tbody>
</table>
```

İsveç'teki yıllık nüfus artışı, 1999–2003

	1999	2000	2001	2002	2003
Nüfus	8 861 426	8 882 792	8 909 128	8 940 788	8 975 670
Artış	7 104	21 366	26 368	31 884	34 882

Tablolar ve kullanımları hakkında daha fazla bilgi için, Tables in HTML documents (HTML dökümanlarında tablolar) ve HTML Techniques for Web Content Accessibility Guidelines 1.0 (Web İçeriği Erişilebilirlik Yöntemleri için HTML Teknikleri 1.0) yazılarından faydalanabilirsiniz.

Daha fazla okuma:

- SimpleQuiz

  Semantik yöntemle nasıl kodlama yapılacağını öğreten harika bir kaynak.

5. (X)HTML

HTML 4.01 kullanarak modern, yapılandırılmış ve standartlara uyumlu web sayfaları oluşturmak mümkündür. Ancak, temiz, semantik kodlamaya geçiş ve XML gibi diğer gelecek kodlama dillerine daha hazırlıklı olmak için yeni oluşturulacak web sayfaları için XHTML 1.0 Strict kullanılmasını öneririm. Bu dökümandaki örnekler de XHTML 1.0 Strict kullanılarak hazırlanmıştır.

XHTML 1.0, HTML 4'ün XML 1.0 içinde yeniden formule edilmiş ve HTML'in yerine kullanılmak için geliştirilmiş halidir. Kullanılmasını önerdiğim XHTML 1.0 Strict sunuma yönelik kodlama yapılmasına izin vermez (HTML 4.01 de buna izin vermez ama burada XHTML'e odaklanmak istiyorum). Bu yüzden, XHTML 1.0 Strict yapıyı sunumdan ayırmaya zorlar.

HTML'in son srümü olan XHTML 1.1 kullanımı teknik açıdan biraz daha zordur. Çünkü belirtim, XHTML 1.1 dökümanlarının MIME türü olarak `application/xhtml+xml` **kullanılması** ve `text/html` olarak **sunulmaması** gerektiğini belirtir. `text/html` kullanmak yasaklanmamıştır ama kullanılması önerilmez. Öte yandan, `application/xhtml+xml` **kullanması gereken** XHTML 1.0, HTML uyumlu ise MIME türü olarak `text/html` kullanabilir. XHTML Media Types (XHTML Medya Türleri) başlıklı W3C notu W3C tarafından önerilen MIME türlerine genel bir bakış içerir.

Ne yazık ki, bazı eski tarayıcılar ve Internet Explorer `application/xhtml+xml` MIME türünü algılamaz ve böyle bir durumda ya kaynak kodu gösterir ya da dökümanı hiç görüntülemez.

`application/xhtml+xml` kullanmak isterseniz, sunucunun, dökümanı isteyen tarayıcının MIME türünü destekleyip desteklemediğini kontrol etmesini ve destekliyorsa kullanmasını aksi taktirde `text/html` kullanmasını sağlamalısınız.

Sunucu tarafı betikleri için PHP kullanıyorsanız içerik üzerine anlaşma betiği, farklı tarayıcılara farklı MIME türü göndermek için kullanılabilir. :

```
<?php
if (stristr($_SERVER[HTTP_ACCEPT], "application/xhtml+xml") ||
stristr($_SERVER["HTTP_USER_AGENT"],"W3C_Validator")) {
    header("Content-Type: application/xhtml+xml; charset=iso-8859-
1");
    header("Vary: Accept");
    echo("<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\n");
    }
else {
    header("Content-Type: text/html; charset=iso-8859-1");
    header("Vary: Accept");
    }
?>
```

Betik, kullanıcı ajanının "application/xhtml+xml" değerini içeren Accept HTTP başlığı gönderip göndermediğini ya da kullanıcı ajanının `application/xhtml+xml` MIME türünü işleyebildiği halde uygun bir Accept HTTP başlığı göndermeyen W3C HTML geçerleyicisi olup olmadığını kontrol eder. Bu durumlardan biri doğruysa döküman `application/xhtml+xml` olarak sunulur. Bu tarayıcılar ayrıca bir de XML deklerasyonu göndermişlerdir. Internet Explorer'ın da içinde bulunduğu diğer tarayıcılara ise `text/html` olarak sunulur ve dökümana, IE/Win tarayıcılarını Quirks (acayip) moduna dönüştüren -ki bunun olmasını istemeyiz- XML deklerasyonu eklenmez.

Content-Type başlığından sonra proksi sunucuları gibi ara önbelleklere, dökümanıın içerik türünün dökümanı isteyen istemcinin yeteneklerine göre değişebildiğini söyleyen bir Vary başlığı gönderilir.

PHP kullanılarak yazılmış daha gelişmiş bir içerik üzerine anlaşma betiğini Serving up XHTML with the correct MIME type (XHTML'i doğru MIME türü ile sunmak) yazısında bulabilirsiniz. Bu betik, kullanıcı ajanının q-rating'ini (belirli bir MIME türünü ne kadar doğru işleyebildiğini belirtir) dikkate alır ve dökümanı `application/xhtml+xml` desteklemeyen kullanıcı ajanlarına `text/html` olarak göndermeden önce XHTML'i HTML 4'e dönüştürür.

Bu da ASP ve VBScript kullananlar için benzer bir betik:

```
<%
If InStr(Request.ServerVariables("HTTP_ACCEPT"),
"application/xhtml+xml") > 0
Or InStr(Request.ServerVariables("HTTP_USER_AGENT"), "W3C_Validator")
> 0 Then
    Response.ContentType = "application/xhtml+xml"
    Response.Write("<?xml version=""1.0"" encoding=""iso-8859-1""?>"
& VBCrLf);
Else
    Response.ContentType = "text/html"
End If
Response.Charset = "iso-8859-1"
%>
```

MIME türü `application/xhtml+xml` olduğunda Mozilla gibi bazı tarayıcıların hata içeren dökümanları görüntülemediğini göreceksiniz. Bu geliştirme sürecinde çok iyi bir davranış olabilir ama XHTML uzmanı olmayan kişilerce güncellenen çevrimiçi sitelerde tüm kodun hatasız olduğundan emin olamadığınız sürece sorunlara yol açabilir. Böyle bir durumdaysanız HTML 4.01 kullanmayı seçebilirsiniz.

Aşağıdaki liste, HTML yerine XHTML 1.0 Strict kullanmayı düşündüğünüzde en çok dikkat etmeniz gereken noktaları göstermektedir:

- **Hep küçük harf kullanın ve tüm özellikleri tırnak içinde yazın:** Tüm eleman ve özellik isimleri küçük harf olmalı ve tüm özellik değerleri tırnak içinde yazılmalı.

  Yanlış: `<A HREF="index.html" CLASS=internal>`

  Doğru: `<a href="index.html" class="internal">`

- **Tüm elemanları kapatın:** HTML'de bazı elemanları kapatmayabilirsiniz. Bu elemanlar bir sonraki elemanın başlangıcında otomatik olarak kapanır. Ama XHTML buna izin vermez. `<img>` gibi içeriği boş olan elemanlar dahil tümü kapatılmalıdır.

  Yanlış: `<li>Bileşen 1`

  Doğru: `<li>Bileşen 1</li>`

  Yanlış: `<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.`

  Doğru: `<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>`

  Yanlış: `<br>`

  Doğru: `<br />`

  Yanlış: `<img src="image.jpg" alt="">`

  Doğru: `<img src="image.jpg" alt="" />`
- **Özellikler kısaltılamaz:** HTML'de bazı özellikler kısaltılabilir ama XHTML buna izin vermez.

  Yanlış: `<input type="checkbox" id="checkbox1" name="checkbox1" checked>`

  Doğru: `<input type="checkbox" id="checkbox1" name="checkbox1" checked="checked" />`
- **Kullanımdan kalkmış elemanları kullanmayın:** HTML 4.01 Transitional ve XHTML 1.0 Transitional'de kullanılabilen bazı elemanlar XHTML 1.0 Strict'de (ve HTML 4.01 Strict'de) kullanımdan kaldırılmıştır. Bunlara örnek şunlar verilebilir:`<font>`, `<center>`, `alink`, `align`, `width`, `height` (bazı elemanlar için) ve `background`.

Daha fazla okuma:

- HTML Versus XHTML

  Web Standartları Projesi W3C'ye HTML XHTML çiftinden hangisini kullanmanız gerektiğini ve bunun nedeni soruyor
- Better Living Through XHTML

  HTML'den XHTML'e geçiş hakkında bir A List Apart makalesi.
- The New York Public Library Online Style Guide

XHTML ve CSS'in nasıl kullanılacağını anlatan güzel bir yazı.

- XHTML 1.0 Differences with HTML 4

  W3C, XHTML 1.0 ile HTML 4 arasındaki farkları anlatıyor.

- XHTML: Differences between Strict & Transitional

  XHTML 1.0 Strict ile Transitional arasındaki farkların bir özeti.

- Serving XHTML with the Right MIME Type

  Web Standartları Projesi W3C'ye XHTML ve HTML için hangi MIME türünü kullanmanız gerektiğini ve bunun nedeni soruyor.

- XHTML Media Types

  XHTML dökümanlarında hangi medya türlerinin kullanılması gerektiğinin bir özeti.

- Bad Tags

  HTML Dog'un XHTML'de kullanmamanız gereken eleman ve özellikleri anlattığı kılavuzu.

- Specifying a MIME Type

  MIME türleri ve farklı sunucu tarafı betik dillerinde içerik üzerine anlaşmanın nasıl yapılacağı hakkında bir döküman.

- Serving XHTML 1.0

  MIME türleri ve XHTML hakkında bir W3C dökümanı.

Doctype

Şu an, çok az HTML dökümanının doğru ve tam bir döküman tipi (doctype) ya da DTD'si (Document Type Declaration) vardır. Birkaç yıl öncesine kadar işlevsellikten çok dekoratif olarak

kullanılmaktaydı ama artık bir döküman tipinin oluşu tarayıcıda dökümanın nasıl çözümlendiğini büyük oranda etkilemektedir.

Geçerli olmak için tüm HTML ve XHTML dökümanlarının döküman tipi deklerasyonu olması gerekir. Doctype, HTML ya da XHTML'in hangi sürümününün kullanıldığını belirtir. Geçerleyiciler geçerleme işleminde, tarayıcılar ise dökümanı çözümlerken hangi modu kullanaclarını belirlerken bu ifadeden faydalanır. Dökümanda doğru ve tam bir doctype deklerasyonu varsa birçok tarayıcı CSS spesifikasyonunu yakından izleyeceği standart modda çalışır. Ayrıca dökümanın tarayıcıda çözümlenmesi de hızlanır çünkü bu durumda tarayıcı geçersiz HTML kodlarını algılamaya, çözümlemeye çalışmaz. Bu ayrıca farklı dökümanlarda farklı çözümleme yapılmasını engeller.

Aşağıdaki doctype dökümanı XHTML 1.0 Strict kullanılarak oluşturulmuştur ve "doctype switching" (döküman tipi geçişi) özelliği bulunan tarayıcıalrın standart moda geçmesini sağlar.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Daha fazla okuma:

- Fix Your Site With the Right DOCTYPE!

  Doctype deklerasyonunun neden kullanılması gerektiği ve nasıl kullanılacağı üzerine bir A List Apart makalesi.

- Activating the Right Layout Mode Using the Doctype Declaration

  Döküman tipi geçişi olan tarayıcıların farklı doctype deklerasyonlarından nasıl etkilendiğinin özeti.

- List of valid DTDs you can use in your document

  W3C'nin doğru doctype deklerasyonlarının listesi.

Karakter kodlaması (Character encoding)

HTML dökümanları karakter kodlamalarını belirtmek zorundadır.

Karakter kodlamasını belirtmenin en doğru yöntemi web sunucusunu karakter kodlaması ile birlikte `content-type` başlığını da gönderecek biçimde yapılandırmaktır. Bunu yapmanın detayları hakkında daha fazla bilgi için kullandığınız web sunucusu yazılımının belgelerine bakmalısınız.

Apache kullanıyorsanız, karakter kodlamasını belirtmek için `.htaccess` dosyanıza bir ya da birkaç kural eklemelisiniz. Örneğin, tüm dosyalarınız utf-8 kullanıyorsa şu kuralı ekleyin:

```
AddDefaultCharset utf-8
```

Belli bir dosya adı uazantısına sahip dosyalar için karakter kodlaması belirtmek için ise:

```
AddCharset utf-8 .html
```

Sunucunuz PHP betikleri kullanmanıza izin veriyorsa karakter kodlaması için şunu kullanabilirsiniz:

```
<?php
    header("Content-Type: application/xhtml+xml; charset=utf-8");
?>
```

Sayfalarınızı HTML olarak sunmak istiyorsanız `application/xhtml+xml` bildirimini `text/html` olarak değiştirin. Herhangi bir nedenle web sunucunuzu kullandığınız karakter kodlamasına göre yapılandıramıyorsanız dökümanlarınızın `<head>` bölümünde bir `<meta>` elemanı kullanın. Hatta web sunucunuz doğru yapılandırılmış da olsa dökümanlarınızda karakter kodlamasını belirtmek iyi bir fikirdir.

Örneğin, aşağıdaki `<meta>` elemanı tarayıcıya dökümanın `ISO-8859-1` karakter kodlaması kullandığını bildirir:

```
<meta http-equiv="content-type" content="text/html;
charset=ISO-8859-1" />
```

Daha fazla okuma:

- WaSP Asks the W3C: Specifying Character Encoding

  Web Standartları Projesi W3C'ye geliştiricilerin karakter kodlamasını nasıl belirtmeleri gerektiğini soruyor.

- The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)

  Farklı karakter kodlamaları hakkında bir makale.

- Using national and special characters in HTML

  HTML dökümanlarında yerel ve özel karakterlerin nasıl kullanılacağının açıklaması.

- Tutorial: Character sets & encodings in XHTML, HTML and CSS (DRAFT)

  Karakter kodlaması seçmek ve belirtmek hakkında bir ders.

6. CSS

Şimdiye kadar çoğunlukla font özelliklerini belirtmek için kullanılan CSS artık dökümanın tüm düzenini kontrol etmek için kulanılabilir. Ancak bunu etkili bir biçimde yapabilmek düzeni tablolar kullanarak belirlemekten çok daha farklı bir yaklaşım gerektirir.

CSS'in düzeni etkili bir biçimde kontrol edebilmesi için yapısal ve semantik XHTML kullamak gerekir.

Tarayıcı desteği

Daha önce de değinildiği gibi CSS için tarayıcı desteği son yıllarda çok gelişmiştir. Ne yazık ki tüm tarayıcı üreticileri açık standartları kullanmakla ilgilenmemektedir, bu yüzden bahsedilen bu desteğin miktarı tarayıcıdan tarayıcıya değişmektedir. Bunun yanında taraycılarda beklendiği biçimde davranmamalarına yolaçan yazılım hataları da bulunmaktadır.

Şu anda (2004) CSS desteği en üst düzeyde olan tarayıcılar Mozilla (ve Gecko üzerine kurulu diğer tarayıcılar: Firefox, Camino, Netscape 6+), Opera ve Safari (WebCore üzerine kurulu diğer

tarayıcılar: OmniWeb 4.5 ve daha ileri sürümleri). Internet Explorer 6/Win aynı derecede CSS desteği sunmaz, ama yine de en temel işleri yapmanıza izin verir. Internet Explorer 5/Mac CSS1 için çok iyi desteğe sahiptir ancak CSS2'yi o kadar iyi desteklemez. Windows için IE 5.* bir miktar destekler ama dikkat etmeniz gereken bazı sorunları vardır. Internet Explorer'ın daha önceki sürümleri bahsetmeye değmeyecek derecede az destek sağlamaktadır. Aynı şey Netscape'in 6'dan önceki sürümleri için de geçerlidir.

Birçok insan şu anda Windows altında Internet Explorer kullanmakta olduğundan tarayıcıyı en küçük ortak bölen olarak düşünmelisiniz. Bu, daha iyi CSS desteği olan tarayıcılar için olan tasarımınızda onların yeteneklerini kullanmamalısınız ya da kullanmayacaksınız anlamına gelmez.

Kullanımda olan tarayıcıların tümü, grafiksel olarak çekici bir düzen yaratmak amacıyla tamamıyla CSS kullanan web sitelerini çözümlemek için gereken seviyede CSS desteğine sahip değildir. Ama ne mutlu ki, birçok web sitesine uğrayan ziyaretçilerin çok çok azı CSS-tabanlı sayfa düzenini doğu biçimde çözümleyemeyecek kadar eski tarayıcı kullanmaktadır.

Bu insanların tamamıyla dışarıda bırakılmayacaklarını belirtmek gerek. Doksanlarda, "yanlış" tarayıcı (aslında Windows için Internet Explorer dışındaki herhangi bir tarayıcı) kullananları, sayfaları doğru görebilmeleri için tarayıcılarını güncellemeleri gerektiğini belirten bir sayfaya yönlendirmek için kullanılan kontrol betikleri çok popülerdi.

Şimdilerde destelenmeyen tarayıcıları ele almanın daha iyi yöntemleri var. Mantıklı ve semantik XHTML kullanmanın bir büyük avantajı da CSS olmasa dahi dökümanları erişilebilir kılmasıdır. — Sayfanın nasıl göründüğü — desteklenen bir tarayıcıyla aynı olmayacaktır ama içerik yine ordadır. Birçok durumda, sitenin ziyaretçilerinin çoğu için, içerik sunumdan çok daha önemlidir. Bu yüzden desteklenmeyen tarayıcı kullananları tamamen dışarda bırakmaktansa stil uygulanmamış bir sayfa görüntülmek daha iyidir.

Bunu yapmanın değişik yöntemleri vardır. En sık kullanılan yöntemlerden biri ilgili CSS dosyasına bağlantıyı sağlamak için `@import` kullanmaktır. Netscape 4 ve daha eski tarayıcılar `@import` bildirimini tanımazlar ve CSS dosyasını alamazlar. CSS'i tarayıcılardan gizlemenin bir çok yöntemi vardır. CSS'i gizleme yöntemlerinin çoğunun kullandığı ortak nokta web tarayıcılarının CSS kodunu ele alışlarındaki yazılım hatalarıdır. Bunun anlamı, CSS'i gizlemek için kullanılan yazılım hatasını düzelten ama CSS'in bazı bölümlerini gizlemeyi gerektiren eksikliği gidermeyen bir güncellemenin

yapılabileceği riskinin varlığıdır. işte bu yüzden sırtınızı CSS atlatma yöntemlerine (CSS hacks) ne kadar az yaslarsanız o kadar iyidir.

Tabii ki tarayıcı kontrolü ve farklı tarayıcılara farklı CSS yönlendirmesi yapmak (ya da hiç CSS göstermemek) için sunucu tarafı teknolojilerini kullanabilirsiniz. Bunu yaparsanız kullandığınız betiği sürekli güncel tutmaya dikkat edin çünkü bir güncelleme ya da yeni bir tarayıcı çıkması durumunda hatalı CSS yönlendirmesi yapıyor duruma düşebilirsiniz.

Daha fazla okuma:

- Tricking Browsers and Hiding Styles

  Eric Meyer CSS'i bazı tarayıcılardan gizlemenin dört yöntemini anlatıyor.

- CSS Filters and Hacks

  CSS'i tarayıcılardan gizleme tekniklerinden çok fazla sayıda örneğin biraraya getirildiği bir yazı.

- Progressive enhancement using CSS

  Modern tarayıcı kullanan insanların deneyimlerini iyileştirmenin yolları hakkında döküman.

CSS kullanmanın değişik yöntemleri

HTML dökümanındaki elemanlara CSS uygulamanın birden çok yöntemi vardır.

Farklı dosyada

Tüm CSS kurallarını bir yada daha fazla ayrı dosyada tutmanın faydaları vardır. HTML dökümanlarının boyutları küçülür, CSS dosyaları tarayıcının önbelleğinde tutulur ve bu yüzden sadece bir kez indirilmeleri yeter ve son olarak tüm web sitesinin görünümünü değiştirmek için yalnızca bir dosyayı değiştirmeniz yeterlidir. Farklı bir dosyada tutulan CSS şöyle görünebilir:

```
h1 {
    font-weight:bold;
```

```
}
```

Not: Farklı bir dosyada tutulan CSS'de `<style>` elemanı bulunmaz.

Bir HTML dökümanına bir CSS dosyasını `<link>` elemanı kullanarak bağlayabilirsiniz:

```
<link rel="stylesheet" type="text/css" href="styles.css" />
```

ya da bir `<style>` elemanı içinde `@import` kuralı kullanarak da CSS dosyasına bağlayabilirsiniz:

```
<style type="text/css">
@import url("styles.css");
</style>
```

Satıriçi

Bir HTML elemanında `style` özelliğini kullanarak doğrudan CSS uygulayabilirsiniz:

```
<h1 style="font-weight:bold;">Rubrik</h1>
```

Sunum ve yapıyı birleştirdiği için bu kullanımdan kaçınmalısınız.

Aynı dosyada

Aynı dosyada kullanılan CSS, dökümanın `<head>` elemanına ait bir `<style>` elemanı içinde bulunur:

```
<style type="text/css">
h1 {
    font-weight:bold;
}
</style>
```

HTML ve CSS kodlarını ayrı dosyalarda bulundurmak en iyi yol olduğundan bu kullanımdan da kaçınmalısınız.

Daha fazla okuma:

- At-Rules

  Diğer konuların yanısıra, CSS bağlama ve medya türleri üzerine bir yazı.

CSS Sözdizimi

Bir CSS kuralı bir seçiciden (selector) ve bir ya da daha fazla deklerasyondan oluşur. Seçici, kuralın hangi HTML elemanına ya da elemanlarına uygulanacağını belirler. Her deklerasyon bir özellik ve bir değerden oluşur. Deklerasyon bloğu {} arasında yazılır ve ; (notalı virgül) ile biter.

Basit bir CSS kuralı şöyle görünür:

```
p {
    color:#0f0;
    font-weight:bold;
}
```

Yukardaki örnekte, p seçicidir. Bunun anlamı bu kuralın dökümandaki tüm <p> elemanlarını etkileyeceğidir. Kuraldaki iki deklerasyon <p> elamanı içindeki tüm metnin yeşil ve kalın yazılmasına sebep olur.

CSS kuralları hakkında daha geniş bilgi için, örneğin, CSS Beginner's Guide veya CSS from the Ground Up sayfalarından ya da bir CSS kitabından faydalanabilirsiniz.

Daha fazla okuma:

- CSS Crib Sheet

  Okuyucularının da yardımıyla Dave Shea pratik CSS ipuçlarının bir listesini hazırlamış.

- Writing Efficient CSS

  John Gallant ve Holly Bergevin kısaltılmış CSS yazmanın yöntemlerini anlatıyorlar.

- Selectutorial - CSS selectors

Farklı CSS seçicilerinin ve nasıl kullandıklarının çok güzel anlatımı.

Gereksiz elemanlar ve sınıflar (class)

CSS kullanmaya başlarken en sık yapılan hata gereksiz XHTML elemanları, gereğinden fazla sınıf ve fazladan elemanı kullanmaktır. Bu kodunuzun geçersiz olacağı anlamına gelmez ama sunumu ve yapıyı birbirinden ayırmanızın en önemli sebeplerinden biri olan daha basit ve temiz kodlar amacını hiçe saymanız anlamına gelir.

Gereksiz XHTML elemanlarının kullanımına bir örnek:

```
<h3><em>Headline</em></h3>
```

Başlığı italik yapmak istiyorsanız <h3> elemanının görünümünü değiştirmek için şu kuralı kullanın:

```
h3 {
    font-style:italic;
}
```

Gereğinden fazla sınıf (class) kullanımı şöyle görünebilir:

```
<div id="main">
    <div class="maincontent">
        <p class="maincontenttext">
            Lorem ipsum dolor
        </p>
    </div>
</div>
```

Oysa şu işinizi görecektir:

```
<div id="main">
    <div>
        <p>
```

```
            Lorem ipsum dolor
        </p>
    </div>
</div>
```

`div#main` etiketi içindeki elemanları CSS aracılığıyla kontrol etmek için bağlamsal (contextual) seçicileri kullanabilirsiniz. Yukardaki örnek için şöyle görünecektir:

```
div#main p {
    /* kurallar */
}
```

Birçok durumda mantıksal XHTML sayfanızın görünümünü CSS kullanarak fazladan kod yazmadan belirleyebilirsiniz. Ancak bazen biraz ekstra kod eklemek buna değen sonuçlar verecektir.

CSS ipuçları

CSS'i ciddi biçimde kullanmaya başladığınızda eninde sonunda bir takım sorunlarla karşılaşacağınız kesindir. Bazı sorunlar yanlış anlamalardan bazılarıysa açıkça ortaya konmamış belirtimler ve hatalı tarayıcılardan kaynaklanabilir. CSS Crib Sheet başlıklı yazıda güzel öenriler Dave Shea tarafından biraraya getirilmiştir. Aşağıda bu yazıdaki en önemli ipuçlarından birkaçını ve CSS Crib Sheet'de olmayan fazladan birkaç ipucunu bulabilirsiniz:

- **Önce geçerleyin**: Hata ayıklama sürecinde öncelikle hem HTML'i hem de CSS'i geçerleyin. Birçok sorun geçersiz kodlardan kaynaklanır.
- **İlk etapta en modern tarayıcıyla test edin, daha sonra kodun diğer tarayıcılarda çalışmasını sağlayın**: Test yaparken kötü ve/veya yanlış CSS gerçekleştirimi olan eski bir tarayıcı kullanıyorsanız yazdığınız CSS o tarayıcının CSS gerçekleştirimine uyarlanır. Daha gelişmiş bir tarayıcıda test yaparken bazı şeyler sizin yapmayı istediğiniz şekilde görünmeyebilir. Kodlarınızı, önce standartlara en uygun tarayıcıda çalışır duruma getirmek sonra yetenekleri daha yetersiz olan tarayıcılara uyarlamak en doğru yöntemdir.
- **CSS kutucuk modelini (box model) anlayın**: Bir elemanın gerçek genişliğini ya da yüksekliğini bulmak için `padding` ve `border` değerlerini `width` ya da `height` değeri ile

toplarsınız. Internet Explorer 5.*/Win'de ise, `padding` ve `border` değerleri verilen `width` ya da `height` içindedir.

Diyelim ki CSS'iniz aşağıdaki gibi:

```
div.box {
    width:300px;
    padding:20px;
    border:10px solid;
}
```

Bu div'in toplam genişliği 360px'dir.

```
10px + 20px + 300px + 20px + 10px = 360px
```

Internet Explorer 5.*/Win'de, toplam genişlik 300px'dir. İçeriğin genişliği ise 240px'dir.

```
300px - 10px - 20px - 20px - 10px = 240px
```

Bu sorunu aşmak için, ya bunu doğru ve yanlış algılayan tarayıcılara farklı değerler vermek amacıyla CSS atlatma yöntemleri kullanırsınız ya da bir eleman için `width` değeriyle **birlikte** `padding` ya da `border` kullanmaktan kaçınırsınız.

CSS kutucuk modeli hakkında detaylı bir anlatım için Box model dökümanından faydalanabilrisiniz.

- **Sıfırdan farklı sayısal değerler için birim belirtin:** CSS'de `width`, `height` ve `font-size` gibi özellikler için birim belirtmeniz gerekir. Bunun istisnası, değerin sıfır (0) olması durumudur. Bu durumda, sıfır her şekilde sıfır olduğundan birim kullanmak gereksizdir.
- **Float mantığını kavrayın:** float kavramını anlamak biraz zor olabilir ama CSS bazlı sayfa düzenlerinde sıklıkla kullanıldığından bu konu çok önemlidir. Float kavramı ile ilgili bazı güzel makaleler şunlardır: Containing Floats, Floatutorial, ve Float: The Theory.
- **"LoVe/HAte":** Bağlantılar için yarı (pseudo) sınıfları sırayla tanımlayın; Link, Visited, Hover, Active.
- **"TRouBLed":** Bir elemanın `margin`, `padding` ya da `border` özelliklerini kısa yöntemle belirtirken değerleri yukarıdan başlayarak saat yönünde verin: Top, Right, Bottom, Left.
- **Sınıfları ve IDleri görünümlerine göre değil işlevlerine göre isimlendirin:** Bir sınıfa `.smallblue` ismini verip daha sonra metni büyük ve kırmızı yapmak istediğinizde sınıf adı kafa karıştırıcı bir hal alacaktır. İsimleri, `.copyright` ya da `.important` gibi, yapıyı ya da işlevi yansıtacak biçimde vermek daha akıllıcadır.

- **CSS büyük/küçük harf duyarlıdır:** `class` ve `id` HTML özelliklerinin değerleri CSS ile kullanıldığında büyük/küçük harf duyarlıdır (bkz. CSS2 syntax and basic data types).
- **IDlerinizi kontrol edin:** Bir dökümanda bir `id` ismi yalnızca bir elemanda olabilirken `class` isimlerini birden fazla eleman alabilir.
- **`class` ve `id` isimlerinde yalnızca izin verilen karakterleri kullanın:** `Class` ve `id` isimleri yalnızca [A-Za-z0-9] ve tire (-) karakterlerinden oluşabilir, ve tire ya da rakam ile başlayamaz.(bkz. CSS2 syntax and basic data types).
- **Yorumlarınızı doğru biçimde ekleyin:** CSS yorumları `/*` ile başlar `*/` ile sonlanır:

```
/* Bu bir yorumdur. */
```

CSS sayfa düzenleri

CSS'in sayfa düzenini belirlemek amacıyla nasıl kullanılacağını anlatan bir çok örnek ve adım adım dersler vardır. Basit bir düzenle başlayıp nasıl çalıştığını öğrenmek ve sonra daha gelişmiş düzenlere geçmek iyi bir yöntem olabilir.

Daha fazla okuma:

- Simple 2 column CSS layout

  Bir başlık, iki sütun ve bir altlık içeren basit bir sayfa düzeni yaratma örneği.

- CSS Layouts

  Değişik CSS sayfa düzenlerine bağlantılar içeren bir sayfa.

7. Erişilebilirlik

Erişilebilirlik, bir web sayfasını erişilebilir kılmak için en önemli neden bu olsa da sadece engelli ziyaretçilerin kullanımını desteklemek değildir. Erişilebilir bir web sayfası engelli ya da değil herkes için en iyi şekilde çalışır ve farklı web tarayıcılar veya araçlar kullanan çok sayıda insan tarafından erişilebilir.

Bir web sayfasını erişilebilir yaptıkça o sayfanın daha az çekici olacağı ya da erişilebilir olmayan bir web sayfasından çok farklı görüneceği sık rastlanan bir yanlış algılamadır. Bu kesinlikle doğru değildir. Erişilebilirlik görünümü etkilemek zorunda değildir.

Erişilebilirliğin herkese nasıl faydası olabileceğine bir örnek: Bir web sitesi bir seminere kayıt olmak amacıyla hazırlanmış bir form içermektedir. Formu kullanarak, seminere üç şehirden hangisinde katılınacağı seçilebilmektedir. Her şehrin adı bir seçim düğmesinin (radio button) yanında yer almaktadır. Eğer formu yaratan kişinin aklında erişilebilirlik yoksa grafiksel bir tarayıcı kullanan kişilerin bir şehir seçmek için imleçlerini minik seçim düğmesinin üzerine getirip tıklamaları gerekecektir. Geliştirici erişilebilirlik konusunu biliyorsa ve her seçim düğmesinin yanındaki etiketleri `<label>` elemanı ile kodlamışsa şehir seçmek için isimlerine de tıklanabilecektir. Sizce formu kullanan herhangi biri için hangisi daha basittir?

Semantik ve iyi yapılandırılmış XHTML kullanmak sizi erişilebilir web sayfalarına oldukça yaklaştıracaktır. Bir dökümanın ne kadar erişilebilir olduğunu anlamak için o dökümanı Lynx gibi metin tabanlı bir taratyıcıda görüntülemeye çalışıp içeriğinin ne kadar anlam ifade edebildiğine bakın. Bu yapmanız gereken erişilebilirlik testlerinden en küçüğü de olsa iyi bir başlangıç olacaktır.

Çerçeveler

Birçok web tasarımcısı tarayıcı penceresini, herbiri kendi HTML dökümanını içeren bağımsız parçalara bölmek için çerçeveleri kullanmayı sever. Bu intranet uygulamaları için faydalı olabilir. Ama herkese açık bir web sitesinde çerçeveleri kullanmanın bir çok sakıncası vardır:

- **Ziyaretçilerin kafasını karıştırısınız.** Web'in en temel presniplerinden biri her sayfanın tekil bir URL tarafından temsil edilmesidir. Bu prensibi kırarak ziyaretçilerin web sitenizin yapısını anlamalarını zorlaştırırsınız.
- **Çerçeveler arama motorları için sorun yaratır.** Bir arama motorunun çerçeveli bir web sitesini indeksleyebilmesi için tüm içerik sayfalarına bağlantı vermeniz gerekir. Yönlendirme bağlantıları gibi sitenin önemli parçalarının eksik olduğu bir dökümanı görüntülemek istiyor olacaklarından arama motorları aracılığıyla gelen ziyaretçiler de sorunlarla karşılaşacaktır. Bazı çerçeve temelli web siteleri, alt sayfaların indekslenmesini engellemek için `robots.txt` dosyası kullanarak bunun üstesinden gelmeye çalışırlar.

Diğer siteler ise arama motorları aracılığıyla siteye gelen herkesi bir JavaScript yardımıyla ana sayfaya yönlendirirler. Hedef daha az ziyaretçi sayısı ise her iki yöntem de iş görür.

- **Çerçeveler favorilere ekleme işlemini engelleyebilir.** Birçok tarayıcı çerçeve temelli bir sitedeki sayfaları favorilere ekleyemez. Böyle bir favoriyi açtığınızda çerçeve grubunun varsayılan durumuna yönlendirilirsiniz, ki bu genellikle web sitesinin ana sayfasıdır.
- **Yazdırmak çok zorlaşabilir.** Ziyaretçiler dökümanları yazdırmak istediğinde zorluklar yaşayacaktır. Birçok tarayıcı bir çerçeveyi yazdırmadan önce o çerçeveyi aktive etmenizi bekler.
- **Bağlantıları e-posta aracılığıyla yollamak zorlaşır.** Çerçeveler, sitedeki bir sayfaya bağlantıları e-posta ile gönderme olasılığını ortadan kaldırır. Bunu aşmanın yöntemleri vardır ama bu yöntemler siteyi karmaşıklaştırır.
- **Sitenin erişilebilirliğini artırmak zorlaşır.** Çerçeveleri destekleyen grafiksel bir tarayıcı kullanmayan ziyaretçiler sorun yaşarlar. Bu yüzden erişilebilirlik kılavuzları çerçeve kullanılmamasını önerirler.

Tüm bunların yanısıra, kendinizi de zora sokarsınız. Çerçeveler bir web sitesinin teknik olarak karmaşıklığını artırır.

Tablolar

Genelde insanlar “sayfa düzeni için tablo kullanmayın” önerisni “tablo kullanmayın” olarak yorumlamaktadır. Ancak bu yanlış bir algılamadır. Kodlamaya çalıştığınız tablo verisiyse bir tablo kullanmalsınız. Ancak, veri tabloları oluştururken tabloları daha mantıklı ve erişilebilir yapmanın birçok yolu olduğunu unutmayın.

Daha fazla okuma:

- A table, s'il vous plaît

  Nasıl erişilebilir tablolar oluşturulabileceği hakkındaki makalelere bağlantılar.

- Tables for Tabular Data

  Tablo verisi için nasıl tablo kullanılacağı üzerine bir makale.

Formlar

Genellikle formları kullanmak gereksiz derecede zordur. Bunun nedeni biraz mantıksız yöntemlerle hazırlanmış olmaları biraz da arka plandaki HTML kodunun formları daha erişilebilir ve kullanımı kolay kılmak amacıyla varolan elemanları kullanmıyor olmasıdır. `<label>`, `<fieldset>` ve `<legend>` elemanları kullanılmak için vardır.

"Formların düzeni için ne kullanılmalı" sorusu sık sorulan bir sorudur. Kimileri formların tablo verileri olduğunu ve tablo biçiminde kodlanmaları gerektiğini söylerken kimileri de CSS kullanılması gerektiğini savunur. Her ikisini de kullanabilirsiniz ama tablo kullanırsanız, formun doğrusallaştırıldığında dahi anlamlı ve kullanılabilir olmasına dikkat edin.

Daha fazla okuma:

- Creating Accessible Forms

  Erişilebilir formlar üzerine bir WebAIM makalesi.

- Better Accessible Forms

  Daha iyi ve erişilebilir formlar hazırlamanın temelleri üzerine bir makale.

JavaScript ve cookieler

JavaScript'e bağımlı olmaktan kaçının. Düşündüğünüzden çok daha fazla insan güvenlik nedeniyle ya da açılan pencereleri engellemek amacıyla tarayıcısında JavaScript kullanımı devre dışı bırakmıştır. Hatta JavaScript desteklemeyen tarayıcı bile kullanıyor olabilirler. TheCounter.com'a göre web kullanıcılarının 6%'sı JavaScript'i devre dışı bırakmış durumdadır. Bu rakam W3Schools.com'da 8%dir.

JavaScript'in kullanıldığı birçok durumda aslında Javascript ziyaretçiye hizmet etmemektedir. Tabii ki JavaScript kullanmanın ziyaretçilere çok daha iyi deneyimler yaşattığı durumlar da vardır. Bunlara form girişlerinin geçerlenmesi örneği verilebilir.

Tüm bunların JavaScript kullanmayın demek olmadığını unutmayın. Söylemek istediğim bir web sitesinin çalışmasını JavaScript'e bağımlı hale getirmeyin.

Aynı şey cookieler için de geçerlidir. Ziyaretçi cookieleri engellediğinde siteniz çalışmayacak duruma gelecek denli cookielere bağımlı olmasın.

8. URLler

Bu bölüm aslında web standartları ya da erişilebilirlikle ilgili değil. Ama burada, çünkü, URLlerin oluşturulma şekli bir web sitesinin arama motorları tarafından ne kadar iyi indeksleneceğini ve ziyaretçileri tarafından ne kadar kullanılabilir olduğunu belirler.

Bazı arama motorları sonları sorgu ifadeleri ile (query string) biten URLleri takip etmez. İçeriğini dinamik olarak veritabanlarından oluşturulan web sitelerinde bu tür URLler çok yaygındır ve yaklaşık şöyle görünürler:

```
http://yourdomain.com/products.asp?item=34627393474632&id=4344
```

Hem arama motorları hem de insanlar için daha kullanışlı bir URL oluşturmanın en kolay yolu URL sanki bir klasöre bakıyormuş gibi görünecek biçimde değiştirmektir. Böyle yapıldığında yukarıdaki örnek şu şekilde görünür:

```
http://yourdomain.com/products/item/34627393474632/id/4344/
```

Web sunucusu bu yeni URLyi yorumlar ve kendi içinde ilk örnekteki haline dönüştürür. Bunun nasıl yapılacağı ile ilgili detaylı dökümanlara erişebilmeniz için bağlantıları bu bölümün sonunda bulabilirisiniz.

Daha kolay ama daha karmaşık bir yöntem ise görünür URLleri yeniden yazmak (rewrite) ve insanlar tarafından okunur kılmaktır:

```
http://yourdomain.com/products/flowers/tulips/
```

Bu tür URL kullanmanın avantajları, arama motorları tarafından sitenin daha etkili indekslenmesi, URLlerin insanlar tarafından kolay okunabilmesi ve sunucu tarafında hangi teknolojiyi kullandığınızı gizleyebilmenizdir. URL, .asp, .cf, .cgi or .jsp gibi, sunucuya özgü dosya uzantılarını

içermediğinden, bunu yapmanız gerektiğinde sunucu tarafında kullandığınız teknolojiyi değiştirmek de kolay olacaktır.

URLlerinizde sorgu ifadeleri kullanma yöntemini seçerseniz ampersandları, &, HTML karşılıklarına, `&`, kodlamalısınız. Böyle yapmazsanız bazı tarayıcılar, yapmaları gerektiği gibi ampersandı bir HTML bileşeninin başlangıcı olarak algılar ve eğer hemen ardından gelen metin bir HTML bileşenine denk geliyorsa tarayıcı URLyi dönüştürür ve birçok durumda da sorgu ifadesini bozar.

Bahsedilmeye değer bir başka konu da birçok web sitesi için `www` alt etki alanı kullanmanın gereksiz oluşu ve `http://www.yourdomain.com/` yerine `http://yourdomain.com/` şeklinde kullanılması gerektiğidir. Bu konuda daha fazla bilgiye no-www.org adresinden ulaşabilirsiniz. `www` alt etki alanı kullanın ya da kullanmayın, web sunucunuzda `http://www.yourdomain.com/` ve `http://yourdomain.com/` adreslerine gelen tüm trafiği aynı URI'a yönlendirmek iyi bir fikirdir.

Daha fazla okuma:

- Generating Simple URLs for Search Engines

  Belirli URL türleri için varolan problemleri ve URLlerinizi nasıl daha iyi hale getirebileceğinizi anlatan bir makale.

- Slash Forward (Some URLs are Better Than Others)

  URLleri "/" ile bitirmenin neden iyi bir fikir olduğunu anlatan yazı.

- Friendly Lasting URLs

  URLler hakkında makale ve derslere bağlantılar içeren sayfa.

- Ampersands and validation

  Sorgu ifadelerindeki kodlanmamış ampersandlarla ilgili sorunların daha detaylı anlatımı ve bir test durumu.

9. Referanslar

Önerilen kitapların, web sitelerinin ve e-posta gruplarının bir listesi.

Kitaplar

- Eric Meyer on CSS: Eric Meyer, ISBN 0-73571-245-X

  CSS kullanımını anlatan proje bazlı bir kitap.

- More Eric Meyer on CSS: Eric Meyer, ISBN 0735714258

  İlk Eric Meyer on CSS'in devamı.

- Cascading Style Sheets – The Definitive Guide: Eric Meyer, ISBN 0-596-00525-3

  CSS önerilerini açıklayan bir referans kitap.

- Designing With Web Standards: Jeffrey Zeldman, ISBN 0735712018

  Standart temelli web tasarımı ve geliştiriminin gerçek dünya örnekleri.

- Building Accessible Websites: Joe Clark, ISBN 073571150X

  Erişilebilirlik ve nasıl erişilebilir web sayfaları oluşturulabileceği hakkında mükemmel bir kitap.

CSS

- CSS (Cascading Style Sheets) Level 1

  W3C'nin resmi belirtimi.

- CSS Level 2 revision 1

  W3C'nin resmi belirtimi.

- CSS Level 3 (in development)
- css-discuss

CSS'in pratik kullanımları ve uygulamaları hakkında konuşmalara adanmış bir e-posta grubu.

- HTML Dog

  HTML ve CSS dersleri, referanslar ve makaleler içeren çok geniş bir web sitesi.

- css Zen Garden

  Aynı HTML dökümanının görünümünün CSS kullanarak tamamen farklı yöntemlerle nasıl değiştirileceğine çok sayıda örnek.

- Max Design Presentations and articles

  CSS hakkında yazılmış çok iyi birkaç makale.

- Position Is Everything

  Makaleler, uygulamalar, tarayıcı hataları ve daha fazlası.

Genel web tasarımı

- A List Apart

  Özellikle web standartları kullanarak tasarımın getirileri ve teknikleri üzerine odaklanmış, web içeriğinin tasarlanması, geliştirilmesi ve anlamı konusunda makaleler içeren haftalık çevrimiçi dergi.

- webdesign-L

  Web'i yaratmakla ilgilenenlerin e-posta listesi. Listede en çok konuşulan konular, web tasarımı ve web geliştirme.

HTML

- HTML 4.01 Specification

W3C'nin resmi belirtimi.

- HTML Dog

  HTML ve CSS dersleri, referanslar ve makaleler içeren çok geniş bir web sitesi.

Erişilebilirlik

- Building Accessible Websites Serialization

  Joe Clark'ın erişilebilirlik kitabının çevrimiçi biçimi.

- Dive Into Accessibility

  Mark Pilgrim'in erişilebilirlik üzerine kitabı.

- Web Content Accessibility Guidelines 1.0

  W3C'nin erişilebilir web siteleri oluşturmak konusunda resmi kılavuzları.

- Evaluation, Repair, and Transformation Tools for Web Content Accessibility

  W3C'nin biraraya topladığı, web sitelerinin erişilebilirliğini sınamak ve geliştirmek için kullanılabilecek araçların listesi.

Web standartları

- The Web Standards Project

  "Web Standartları Projesi, tüm web teknolojilerine basit ve karşılanabilir erişimi garantileyen standartlar için savaşan en köklü koalisyondur."

- MACCAWS: Making A Commercial Case for Adopting Web Standards

  MACCAWS'ın misyonu, web standartlarını müşteriler için ticari anlamda çekici bir seçenek haline getirmek için web yazarlarına gereken kaynakları sağlamaktır. MACCAWS'taki

dökümanlardan en önemli ikisi şunlardır: What Every Web Site Owner Should Know About Standards: A Web Standards Primer ve The Way Forward with Web Standards.

- A Roadmap to Standards

  Web standartlarını kullanmaya başlamak için Dave Shea'nın hazırladığı bir kılavuz.

XHTML

- XHTML™ 1.0 The Extensible HyperText Markup Language

  W3C'nin resmi belirtimi.

- HTML Dog

  HTML ve CSS dersleri, referanslar ve makaleler içeren çok geniş bir web sitesi.

10. Terimler sözlüğü

Erişilebilirlik

Erişilebilir bir web sitesi, donanım ve yazılım olarak ne kullanırlarsa kullansınlar ve sitede ne ile geziniyor olurlarsa olsunlar herkes için erişilebilir sitedir.

CSS (Cascading Style Sheets)

Bir dökümanın nasıl sunulacağını tanımlayan kurallar.

HTML (HyperText Markup Language)

Bir dökümanının yapısını kodlamada kullanılan dil.

Sunum

Bir web sitesinin görünümü (ya da duyuluşu).

Yapı

Bir dökümanın olmazsa olmaz parçaları ve içeriğinin mantıksal kodları.

Kodlama/İşaretleme (Markup)

Kodlayarak/işaretleyerek bir dökümana ve içeriğine yapısını ve anlamını verirsiniz. Web'de kodlama için HTML ve XHTML kullanılır.

Geçerleme

Geçerleme, bir dökümanın yazıldığı dilin kurallarına uyup uymadığını kontrol etmektir. Bir metnin yazım ve dilbilgisi hatalarını kontrol etmeye benzetilebilir.

W3C (World Wide Web Consortium)

Web için belirtimler, kılavuzlar ve araçlar üreten organizasyon.

Web standartları

Web standartları, W3C ve diğer standart oluşumlar tarafından yayınlanan ve web tabanlı içeriğin yaratılması ve çözümlenmesi için kullanılan teknolojilerdir. Bu teknolojiler webde yayınlanan dökümanların gelecekte de kullanılabilir ve olabildiğince erişilebilir olmasını sağlamak için tasarlanmışlardır.

XHTML (Extensible HyperText Markup Language)

XML'in kurallarına uyacak biçimde yeniden formüle edilmiş HTML.

XML (Extensible Markup Language)

HTML gibi görünen ama yazarın uygun elemanlar belirterek veriyi tanımlayabildiği kodlama dili.

Yorumlar, sorular ve öneriler? Lütfen bana ulaşın..