Autoria Web

Aula 04 - XHTML e CSS: definições e particularidades

Apresentação

Nesta aula, você aprenderá as particularidades do XHTML, linguagem de marcação similar ao HTML criada com base no XML e que serviu como base (além do HTML tradicional) para a criação do HTML5; entenderá as diferenças entre HTML e XHTML; aprenderá como validar páginas XHTML e associar regras CSS a essas páginas, bem como em que consistem essas regras.

Vídeo 01 - Apresentação

Objetivos

Ao final desta aula, você deverá ser capaz de:

- Diferenciar HTML de (X)HTML.
- Criar páginas XHTML e validá-las.
- Compreender o objetivo para o qual a regra CSS foi criada, conhecendo suas funções e vantagens.
- Definir as partes constituintes de uma regra CSS.
- Conceituar e especificar seletores.

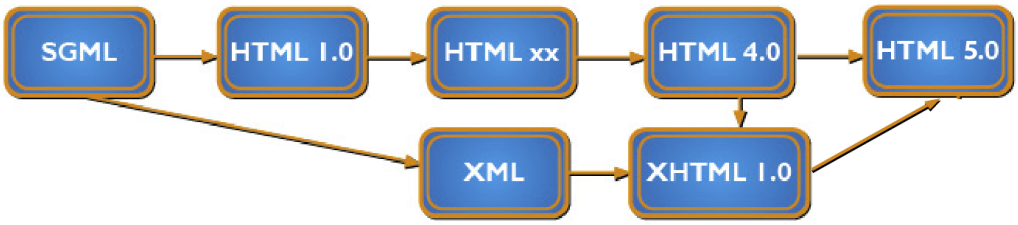
XHTML

XHTML é o acrônimo para eXtensible Hyper Text Markup Language, que pode ser traduzido para Linguagem Extensível de Marcação para Hipertexto. Ela é uma linguagem de marcação para confecção de páginas web muito parecida com HTML.

Origem

A maioria das linguagens de marcação é baseada em uma linguagem geral chamada SGML (SGML é uma metalinguagem complexa, desenvolvida para servir de base para a criação de outras linguagens.) (Standard Generalized Markup Language) como, por exemplo, o próprio HTML e a linguagem XML(eXtensible Markup Language). Como ilustrado na **Figura 1**, enquanto HTML foi criada com base em SGML diretamente, XHTML foi criada com base em XML. Isso confere à XHTML a característica de ser bem formada.

Figura 01 - Etapas do desenvolvimento e evolução das linguagens de marcação



Fonte: Alexandre Pereira – Web Standards Acesso em: 15 set. 2010.

A **Figura 1** mostra as etapas do desenvolvimento e evolução das linguagens citadas anteriormente. Primeiramente surgiu a linguagem SGML, criada no final da década de 1960 com o objetivo de construir um sistema portátil para intercâmbio e manipulação de documentos. Com base na especificação da SGML e no contexto da invenção da web, Tim Berners Lee criou o embrião do que seria a linguagem de marcação para hipertextos (HTML), no ano de 1990. Nos anos seguintes, HTML evoluiu de modo que surgiram inúmeras versões.

Com base em SGML, a XML foi criada em meados do ano de 1998 pelo W3C, com o objetivo de facilitar a comunicação entre programas e documentos. No mesmo ano, XML tornou-se um padrão internacional e várias linguagens de marcação foram

criadas com o objetivo de segui-lo. Várias adaptações feitas na linguagem HTML, para seguir o padrão XML, deram origem ao XHTML no final da década de 1990.

A transformação de um documento existente de HTML para XHTML é uma tarefa bem simples, já que as tags e os atributos da XHTML são os mesmos da HTML 4.01. XHTML nada mais é que um HTML refinado. Logo mais veremos as principais diferenças entre essas duas linguagens de marcação.

Vantagens de usar XHTML

Várias são as vantagens de se usar XHTML, dentre elas pode-se citar:

- Estabilidade – XHTML é uma linguagem que tem sofrido poucas atualizações desde que foi definida, estando na versão 2.0 atualmente.

- Compatibilidade – como o que se pretende é que haja uma padronização da web com o XHTML, então a tendência é que futuras versões de browsers e agentes de usuários (Aplicações que leem documentos HTML e/ou XHTML que não são necessariamente browsers.) passem a dar suporte integral a essa linguagem. Assim, sendo compatível com as futuras aplicações, garante-se que implementações em XHTML durarão por longos anos.

- Desempenho – carregamento de uma página XHTML é mais rápido, pois o código a ser interpretado é limpo e o browser não tem que decidir sobre renderização de erros de código. Um exemplo disso é não ter que interpretar a falta de tags ou tags fechando e/ou abrindo em lugares errados.

- Interoperabilidade e portabilidade – uma página XHTML é mais acessível aos browsers e aplicações de usuário padrão.

- Simplicidade – editar um código XHTML existente é uma tarefa bem simples por se tratar de uma escrita limpa e evidente, ou seja, que contempla apenas conteúdo, sem se preocupar com a aparência.

Diferenças entre HTML e XHTML

Um documento XHTML deve ser estruturado rigorosamente de acordo com as regras definidas para XML 1.0 e, assim, não são permitidos erros de marcação, ao contrário do que verificamos em HTML (nas aulas anteriores). Existem algumas diferenças básicas (as principais são citadas a seguir) entre um documento HTML e um XHTML. Padronizadamente em XHTML:

Vídeo 02 - html e Xhtml

1 – O documento (arquivo) deve ser bem-formado e possuir um único elemento raiz.

Isso significa que esses documentos devem ter um elemento raiz (<html>) e todos os outros devem estar aninhados dentro dessa tag raiz, que é o elemento pai. Elementos filhos devem ser em pares (abrir e fechar) e corretamente aninhados dentro do seu elemento pai. A estrutura básica do documento deve ser conforme mostrado a seguir:

```
<html>
   <head><title></title>... </head>
   <body> ... </body>
</html>
```

Na verdade, existem alguns outros elementos obrigatórios para formação do documento XHTML, mas não se preocupe com eles agora, pois serão estudados em outro momento desta aula.

2 – Os elementos devem ser adequadamente aninhados.

Em XHTML, todos os elementos devem estar devidamente aninhados uns dentro dos outros, ou seja, a primeira tag a ser aberta é a última a ser fechada, a segunda é a penúltima e assim por diante. Por exemplo, a marcação mostrada abaixo:

 Esse texto está em negrito e itálico </ i></ b>

Enquanto que em HTML, alguns elementos podem ser indevidamente aninhados uns dentro dos outros, como mostrado abaixo:

 Esse texto está em negrito e itálico

Repare que, no exemplo anterior, a primeira tag a ser aberta foi a , seguida de . No fechamento das tags, o que ocorre é que é a primeira a ser fechada, ficando dentro de .... Embora esse código HTML possa ser lido pelos browsers, não necessariamente eles o interpretarão adequadamente.

3 – As tags devem ser escritas em caixa baixa (minúsculo).

A metalinguagem XML é sensível ao tamanho de caixa de fonte, ou seja, ela faz a diferenciação entre letras maiúsculas e minúsculas. Como XHTML é uma aplicação XML, ela também possui essa sensibilidade, sendo obrigatório escrever todas as tags XHTML em minúsculo. Assim, em XHTML, nunca escreva:

<P><I>Parágrafo</ I></ P>

O correto é:

Parágrafo</ i></ p>

4 – O uso de tags de fechamento é obrigatório.

XML não permite a omissão de qualquer tag de fechamento; como consequência, XHTML também não. Todo elemento deve ter uma tag de fechamento. Assim, em XHTML, nunca escreva:

 Este é um parágrafo

 Este é outro parágrafo

O correto é:

 Este é um parágrafo

 Este é outro parágrafo

Diferenças entre HTML e XHTML

5 – Elementos vazios devem ser fechados.

Os elementos vazios também devem ser fechados. Existem duas maneiras de se realizar isso:

a. Utilizando uma tag de abertura e uma de fechamento, como no exemplo abaixo:

Uma quebra de linha:
</br>

Uma régua horizontal: <hr></hr>

b. Terminando a tag de abertura com “/>”, como no exemplo abaixo:

Uma quebra de linha:

Uma régua horizontal: <hr/>

Em XHTML, nunca se deve escrever:

Uma quebra de linha:

Uma régua horizontal: <hr>

6 – Os atributos possuem uma sintaxe específica.

Em XHTML, existem algumas regras que devem ser seguidas para evitar transtornos, quando da definição dos atributos. São especificidades quanto aos nomes e aos valores desses atributos.

a. os nomes dos atributos devem ser escritos em minúsculo, tal como mostrado no exemplo abaixo:

<table border="0">

E nunca se deve escrever o atributo em caixa alta, como mostrado abaixo:

<table BORDER= "0">

b. Os valores dos atributos são tratados como strings (valores strings são tratados como caracteres, ou seja, quando preenchemos um valor do atributo com números, ele é tratado como texto.), e por isso devem ser escritos entre aspas, ainda que o valor seja numérico, tal como mostrado no exemplo abaixo:

<table border= "0">

E nunca se deve escrever o valor do atributo sem aspas duplas, como mostrado abaixo:

<table border=0>

c. A sintaxe dos atributos deve ser escrita por completo, ou seja, contendo os campos do nome e do valor, mesmo que o valor possua a mesma grafia do nome, como mostrado abaixo:

<input checked="checked" />

<input disabled="disabled" />

<option selected="selected" />

A definição do atributo apenas com o nome, sem atribuição de valor, está incompleta. Assim, estão incorretas as definições abaixo:

<input checked>

<input disabled>

<option selected>

A seguir estão listados alguns exemplos de atributos que se enquadram nessa recomendação e a sintaxe correspondente em XHTML.

HTML	XHTML
Checked	checked="checked"
Disabled	disabled="disabled"
Selected	selected="selected"
Multiple	multiple="multiple"
Noresize	oresize="noresize"

Você já está acostumado com a maioria das exigências de XHTML mostradas anteriormente, uma vez que elas foram trabalhadas quando você estudou HTML nas aulas anteriores. Para fixar melhor essas diferenças, vamos realizar uma atividade? Depois disso você aprenderá quais são as outras diferenças entre HTML e XHTML.

Atividade 01

1. O quadro a seguir mostra um exemplo de código HTML. Imagine que seja necessário transformá-lo em um código XHTML bem-formado. Que modificações deveriam ser feitas para essa transformação? Reescreva o código de maneira que se ajuste perfeitamente às particularidades de XHTML.

```
<HTML>
<head>
<title>Título
<HeaD>
<body>
<h1><b><a name=”A6”>Título H1</b></a></h1>
<hrsize=5/>
<p>Parágrafo1
<p><i><B>Parágrafo 2</i>
<HT>
<a HREF=#A6”>Voltar ao topo da página.</a><HT>
</BODY>
<HTML/>
```

Diferenças entre HTML e XHTML

Seguem mais algumas diferenças entre HTML e XHTML.

7 – Pontos de âncora em XHTML.

Você aprendeu a fazer pontos de âncora em HTML na **Aula 2**. Viu que âncoras são pontos de um documento para onde certo link deve apontar e, para especificá-las, você atribui um nome através do atributo name. Em XHTML, para se obter o mesmo efeito, substitua o atributo name por id, como no exemplo abaixo:

 Voltar ao topo

Isso também deve acontecer para os elementos applet, form, frame, iframe, img e map. Mas, os elementos que compõem um form, como <input> e <select>, continuam permitindo (e exigindo, no caso de submissão de dados) o elemento 'name'.

Uma vez que os atributos id e name têm o mesmo objetivo, que é determinar um identificador para o elemento, em XHTML deve-se usar apenas o atributo id, e somente um por elemento. Além disso, cada identificador deve ser único no documento. Veja os exemplos a seguir:

Errado:

Certo:

8 – O atributo alt para imagens é obrigatório.

Na **Aula 3**, você aprendeu a inserir imagens em sua página de modo a torná-la mais atrativa e conheceu o atributo alt, que fornece uma descrição curta sobre a imagem quando o cursor se move sobre ela. Esse atributo é utilizado quando o

navegador não consegue exibir a figura. Em XHTML, o uso do atributo alt é obrigatório, ainda que seu preenchimento seja vazio (espaço entre aspas duplas). Veja os exemplos a seguir.

ou

Vídeo 03 - Convertendo html para xhtml

Elementos XHTML obrigatórios

Um documento XHTML possui, necessariamente, duas partes: a declaração da DTD (*Document Type Definition*) e o elemento raiz <html>. A tag <html> deve conter pelo menos a seção head (com a tag <title>) e a seção body. A seguir, é ilustrado um modelo de documento XHTML mínimo.

```
<!DOCTYPE Definição do tipo de documento>
<htmlxmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Título</title>
</head>
<body>
</body>
</html>
```

Veja ainda que, no exemplo anterior, o elemento <html> possui o atributo xmlns, pois este é exigido na linguagem XHTML. Entretanto, se você não o declarar, não haverá problema, pois o validador do W3C assume "xmlns=http://www.w3.org/1999/xhtml" como valor default e adicionará à tag mesmo que você não o inclua.

A DTD XHTML

Tal como a declaração DOCTYPE explicada para HTML na primeira aula desta disciplina, em XHTML, essa declaração define o tipo de documento e sempre deve aparecer na primeira linha do documento XHTML. O resto do documento é semelhante ao HTML. A seguir, listamos algumas funções e características da DTD:

- A DTD especifica a sintaxe de uma página web.
- A DTD é usada por aplicações SGML, tal como HTML, para especificar regras que se aplicam à marcação de documentos de um tipo particular, incluindo um conjunto de declarações de elementos e de entidades.
- Uma DTD XHTML descreve de forma precisa, em linguagem legível ao computador, a sintaxe e a gramática permitida da marcação XHTML.

O padrão XHTML possui três definições de tipos de documentos: strict, transitional e frameset. A mais comum é a XHTML Transitional. A seguir, nós temos um exemplo e a definição de cada tipo.

XHTML 1.0 Strict

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

A DTD strict deve ser usada quando se deseja uma marcação limpa, livre da confusão da apresentação, pois não admite qualquer item dessa natureza dentro dos elementos e tampouco elementos em desuso (DEPRECATED). É a mais rígida de todas, sendo indicada para uso com folhas de estilo em cascata (CSS).

XHTML 1.0 Transitional

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

A DTD transitional deve ser usada quando você precisar tirar vantagens das características de apresentação da HTML e quando você quer dar suporte aos navegadores que não entendem CSS. É mais flexível que a anterior e é indicada para documentos que empregam elementos em desuso.

XHTML 1.0 Frameset

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

A DTD frameset deve ser usada quando você quer usar frames para particionar a janela do navegador em duas ou mais molduras. É regida pelas mesmas diretrizes da declaração transitional.

Uma cópia da lista dos elementos de HTML 4.01 permitidos em cada um desses tipos de declaração de documento (DTD) é encontrada em <http://www.w3schools.com/tags/default.asp>.

Marcação semântica é essencial

Semântica é o estudo do significado das palavras e suas relações umas com as outras. Então, quando falamos em um código XHTML semântico, referimo-nos ao uso correto das tags, ou seja, utilizá-las na função real para a qual elas foram criadas.

Nesse sentido, se você vai escrever uma lista, você deve utilizar as tags criadas para esse fim: <ol>, <ul> e <dl>. Se precisar de um cabeçalho, você deve usar as tags <h1> a <h6>.

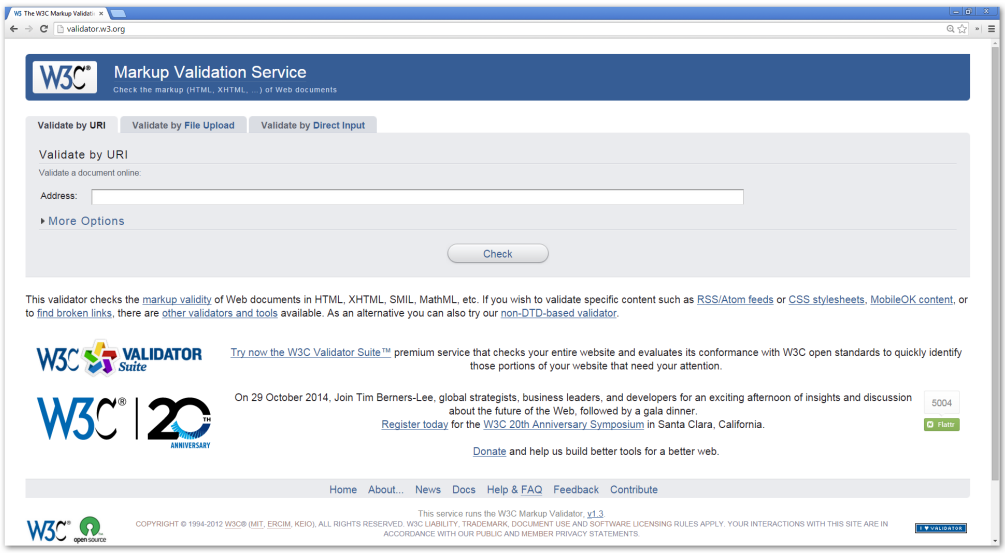
Utilizar as tags no sentido correto justifica o termo Web Standards (Normas para web.). Sendo assim, o desenvolvedor deve ter preocupação com a semântica da marcação.

Validação do XHTML

A função dos validadores XHTML é ajudar a verificar se seu código segue às regras da linguagem. Ele pode mostrar onde você esqueceu-se de fechar uma tag, onde esqueceu um atributo que não deveria ter esquecido ou o lugar onde colocou as aspas e não deveria ter feito.

Como apresentamos na primeira aula desta disciplina, o W3C disponibiliza em sua página um validador gratuito para documentos XHTML. Nele você pode digitar a url, o caminho para o arquivo no seu disco rígido ou mesmo o código digitado diretamente, e então esse documento ou código será analisado, disponibilizando-se um relatório completo e detalhado das não conformidades porventura existentes. Essa é uma ferramenta excelente para você usar durante a elaboração ou migração do seu documento de HTML para XHTML. Serve como um verdadeiro revisor do código que você cria. A figura a seguir mostra a interface do validador disponível no site do W3C.

Figura 02 - Interface do validador XHTML da W3C

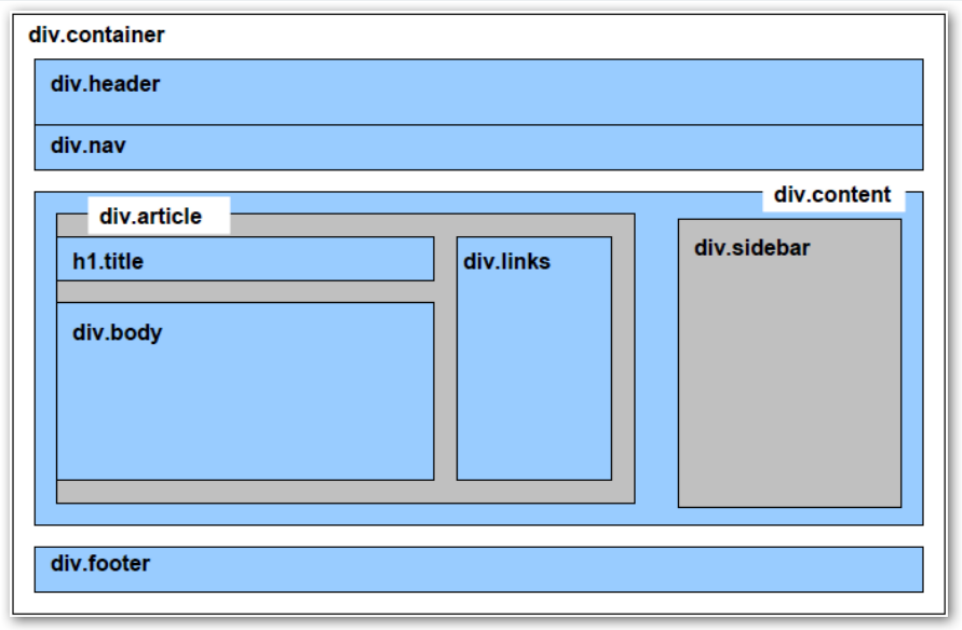


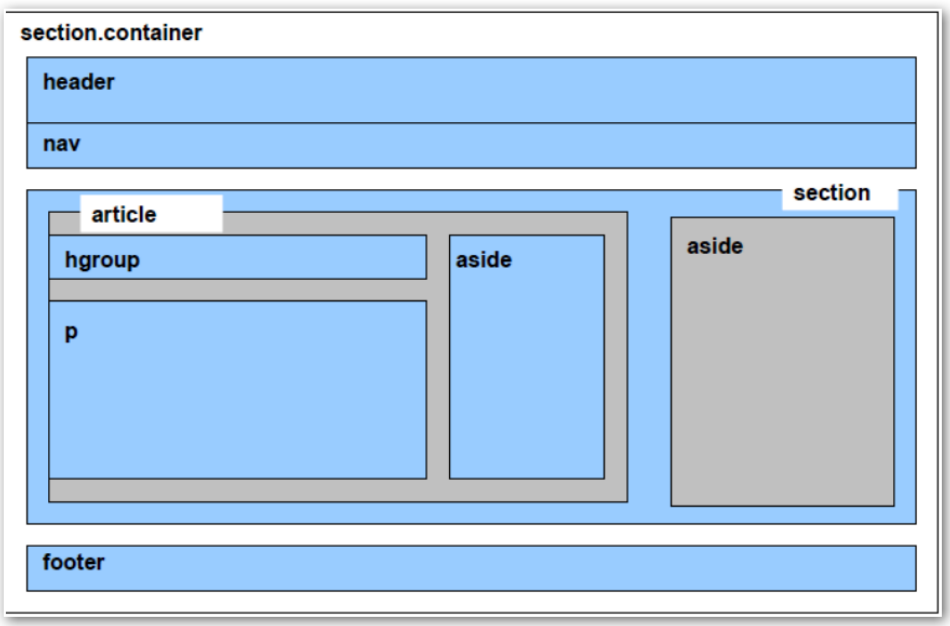
Vídeo 04 - Validador html

Com a validação dos documentos (X)HTML, você garante que está construindo páginas de acordo com um padrão o qual (futuramente) garantirá compatibilidade com todos os browsers. Vamos ver se os documentos HTML que você desenvolveu até agora são validados por essa ferramenta?

Além de novas tags e APIs um dos principais avanços do HTML5 foi a adição de tags semânticas. As novas tags (<header>, <footer>, <article>, <aside>, <section>, <nav> e <figure>) não alteram o comportamento visual do site pois é somente uma alteração semântica, ou seja, de significado.

Nas versões anteriores do HTML não havia uma forma apropriada oferecida pela linguagem para definir as divisões e partes da estrutura de uma página web. Neste caso, era comum os desenvolvedores usarem as tags e os atributos id e class com nomenclaturas para cada região. Vamos ver um exemplo!





Nas imagens acima podemos observar o uso das novas tags semâticas do HTML5. Não ficou mais intuitivo?

Atividade 03

---

1. Acesse o validador XHTML da W3C e verifique quais são os erros apontados por essa ferramenta para o documento HTML que você desenvolveu sobre si na aula passada.
2. De acordo com as regras de XHTML, reescreva os links e imagens desse documento.

Então, você conseguiu entender e corrigir os erros relacionados pelo validador?

A partir de agora você deve adotar o uso dessa ferramenta como rotina para validar todos os documentos que você desenvolver. Com o tempo, você perceberá que esse hábito trará muitas vantagens para o desenvolvimento e manutenção dos documentos web sob sua responsabilidade.

Com isso, concluímos o básico do nosso estudo em HTML. Muitos outros aspectos podem ser aprendidos no próprio site da W3C e nas mais diversas fontes que estão disponíveis na web. Agora, começaremos a cuidar da parte visual do nosso site. Vamos, então, começar a estudar CSS?

CSS

CSS (Cascading Style Sheet) significa, em português, Folhas de Estilo em Cascata. Mas o que significa o termo cascata? Significa que a aplicação de CSS a um documento ocorre como um efeito cascata. Vamos ser mais claros:

Uma folha de estilo pode ser originária de 3 fontes distintas (SILVA, 2008): autor (você, projetista web), usuário e browser. Usualmente, o autor cria o CSS e vincula-o em seus documentos web. Entretanto, o usuário também pode criar estilos de acordo com suas preferências e necessidades e inseri-las nos browsers (geralmente os browsers fornecem funcionalidades para criação de folhas de estilo do usuário.). E os browsers, por sua vez, possuem folhas de estilo próprias.

Cada um desses CSS possui um peso, assim, há uma ordem de precedência na aplicação deles: primeiro é aplicado o CSS do navegador, depois do autor e, por fim, do usuário. O último estilo aplicado é aquele visto no browser, porque ele sobrepõe os demais. Esse mesmo efeito cascata ocorre quando o autor define em um mesmo CSS duas ou mais regras para estilizar um mesmo elemento. Por exemplo, suponha que seja definida a cor vermelha para os títulos <h1> e no mesmo documento seja definida a cor azul para esses mesmos títulos, então, a cor que aparecerá no browser será a última declarada.

Origem

A linguagem HTML foi criada com a finalidade de marcar e estruturar páginas web. Ela, na sua essência, não se preocupava com o estilo de apresentação dos elementos (cores de fonte, tamanhos de texto etc.). Com a evolução do HTML, novas tags e atributos foram criados com o objetivo de incumbir tal função à linguagem. Com isso, a velha linguagem de marcação passou a exercer uma dupla função: estruturar o conteúdo através da marcação e dar a aparência final, ou seja, estilizá-lo.

Essa dupla jornada do HTML ocasionou vários problemas e tornou os códigos cada vez mais extensos e complexos. A mistura, que de início agradou, acabou por trazer uma grande dor de cabeça aos projetistas web. A solução era dissociar a estilização da linguagem de marcação.

Com o objetivo de solucionar esse problema, no ano de 1994, surgiu a proposta de implementação das Folhas de Estilo em Cascata (CSS – Cascading Style Sheet), que seriam responsáveis pela formatação das páginas. Assim, as tags (X)HTML seriam usadas apenas para marcação do conteúdo do documento. Toda formatação de cor, fonte, alinhamento de parágrafo, entre outros, ficaria a cargo das CSS. Com isso, na prática, o que aconteceu foi uma centralização da formatação, através da qual se tornou possível formatar milhares de páginas simultaneamente, com um mínimo de esforço. No ano de 1996, o W3C recomendou oficialmente o uso das CSS.

Vantagens de usar CSS

Para que você entenda melhor a vantagem do uso das CSS, vamos imaginar uma situação hipotética, na qual você construiu um site de 100 páginas (100 arquivos) e por algum motivo necessita mudar a cor de todos os títulos marcados com o elemento <h1>.

Se você estivesse usando tags HTML para formatar a cor desses títulos, o atributo de estilização estaria dentro da tag <h1> e, para mudar a cor do título, você teria que mudar o valor desse atributo. Nesse caso, isso teria que ser feito em cada tag. Se seu documento tivesse uma média de 5 elementos <h1> por página, você teria que modificar 500 tags (5*100)! Imagine o tempo que você perderia nessa tarefa, abrindo e fechando arquivos, procurando as tags <h1> e modificando a cor! Imagine também a quantidade de erros que essa árdua tarefa poderia ocasionar.

Enquanto que usando CSS bastaria você associar a tag <h1> à cor desejada, através de regras que serão estudadas ainda nesta aula. E no caso de se pretender realizar a mudança, seu único esforço seria modificar a cor na declaração da regra CSS uma única vez.

Sendo assim, podemos elencar algumas vantagens do uso de CSS, quais sejam: economia de tempo, diminuição do tamanho do código, facilidade de manter e alterar o conteúdo de sua página e a possibilidade de um maior controle e padronização na sua apresentação.

Sintaxe CSS

Um estilo é definido através de uma regra, que possui sintaxe própria. Um conjunto de regras CSS forma uma folha de estilos.

Uma regra CSS, compõe-se de três partes: um seletor, uma propriedade e um valor. Veja a declaração a seguir e as definições de cada uma das partes:

```
seletor{propriedade:valor;}
```

1) Seletor – genericamente, é o elemento de marcação (X)HTML para o qual a regra será válida. São alvos da regra CSS, por exemplo, os parágrafos e os links.

2) Propriedade – define qual característica do elemento (X)HTML será estilizada pela regra, por exemplo, a cor, o tamanho ou o tipo de fonte.

3) Valor – é a característica quantitativa ou qualitativa a ser assumida pela propriedade. São os valores que essas propriedades podem assumir.

Na sintaxe de uma regra CSS, escreve-se o seletor e, em seguida, a propriedade e o valor. Esses dois últimos devem ser separados por dois pontos e entre chaves, como mostra o exemplo anterior. Chamamos de declaração o par formado por uma propriedade e seu valor.

Uma regra pode conter mais de uma declaração, ou seja, podemos atribuir várias características de estilo a um mesmo elemento (X)HTML usando apenas uma regra, através de um único seletor. Por exemplo, eu posso desejar que um parágrafo, marcado pelo elemento p, tenha cor vermelha e seja escrito em itálico. A regra que estiliza o parágrafo citado acima está descrita a seguir.

```
p {color:red; font-style:italic;}
```

No exemplo anterior, você vê uma regra formada por duas declarações separadas por ponto e vírgula, na qual se define que o texto dos parágrafos é estilizado com a cor vermelha e a fonte itálica. O seletor é o elemento (X)HTML p (que marca um parágrafo através da tag <p>). As duas declarações são color:red e font-style:italic, nas quais as propriedades são color e font-style e os seus respectivos valores são red e italic.

Algumas regras sintáticas merecem destaque para que você escreva suas folhas de estilo da melhor maneira. Veja a seguir quais são elas.

---

1) Separação das declarações de uma regra

Quando mais de uma declaração for definida na regra, deve-se usar ponto e vírgula para separá-las. O ponto e vírgula é facultativo no caso de propriedade única e também após a declaração da última propriedade, no caso de haver mais de uma. Entretanto, recomenda-se que sempre se use o ponto e vírgula após uma declaração, isso porque, ao construir nossa folha de estilo, sempre estaremos acrescentando declarações, e essa prática nos poupará tempo de revisões.

Você também pode escrever todas as declarações na mesma linha ou em linhas separadas, o efeito será o mesmo. Sendo assim, tanto faz você escrever de uma ou de outra maneira mostrada a seguir.

```
p {color:red;font-style:italic;}
ou
p{
  color:red;
  font-style:italic;
}
```

É notório que a declaração feita em uma única linha prejudica a legibilidade. Assim, dê preferência à segunda forma de escrever. Dessa forma, será mais fácil ler e manter o código CSS.

Vídeo 05 - Exemplo CSS

2 – Sintaxe de valores que são palavras compostas

Quando uma propriedade tiver um valor que for uma palavra composta, ou seja, a palavra composta é separada por espaços, esse valor deverá ser escrito entre aspas duplas ou simples. Essa regra não se aplica aos valores de palavras compostas separadas por hifens, os quais devem ser definidos normalmente. As duas declarações abaixo estão corretas:

```
h1{font-family:“times new roman”;}

ou

h1{font-family:‘times new roman’;}
```

3 – Sensibilidade ao tamanho da caixa da fonte

Ao contrário do XHTML, a sintaxe da regra CSS não é sensível ao tamanho da caixa de fonte. Sendo assim, você pode escrever suas regras com letras maiúsculas ou minúsculas. Além disso, você pode inserir espaços entre os componentes das regras, que não fará diferença alguma, ficando a seu critério. Abaixo, temos alguns exemplos de regras válidas:

```
p{color:red;}
P { color: red;}
p {COLOR: RED;}
```

Já que existem inúmeras formas de você escrever o seu código CSS, adote a que melhor se adapta a você. Mas não misture as formas, por exemplo, se você começou a escrever o código em letras minúsculas, escreva-o dessa forma até o fim. Isso faz com que seu código seja consistente e facilita sua manutenção.

4) Agrupamento de seletores

Se vários elementos serão estilizados da mesma forma, pode-se agrupá-los em uma mesma regra. Por exemplo, ao construir a sua página, você pode desejar escrever todos os títulos de uma mesma cor, independente do seu nível (h1, h2 ou h3). Ao invés de você declarar uma regra para cada seletor, você pode agrupá-los. Os exemplos a seguir mostram as duas formas de se fazer essa estilização, com agrupamento de seletores e sem agrupamento.

Com agrupamento:

```
h1,h2,h3{color:red;}

Sem agrupamento:

h1{color:red;}
h2{color:red;}
h3{color:red;}
```

Vinculação das folhas de estilo ao documento

Você deve especificar no documento (X)HTML onde o browser deve buscar as folhas de estilo, ou seja, você precisa vincular a declaração CSS ao documento (X)HTML. Existem três formas de vinculação, abaixo descritas.

1) Escritas no próprio elemento (X)HTML (estilo inline)

A declaração é feita dentro das próprias tags do (X)HTML. É um método direto e simples de aplicar estilos a um elemento de marcação através do atributo style. Seu efeito estará restrito a essa única tag. Nesse caso, a principal vantagem das CSS fica prejudicada, uma vez que a centralização da estilização não se aplica. A seguir, temos um exemplo da aplicação dessa vinculação.

```
<p style="color:red;font-style:italic;">...</p>
```

2) Arquivos externos

Nesse caso, um arquivo contendo todas as regras da folha de estilo é criado e ele pode ser vinculado a qualquer página de um site. Você pode criar esse arquivo em um editor de texto simples e salvá-lo com a extensão .css, da mesma maneira que você cria arquivos HTML.

Essa é a forma de vinculação mais utilizada, pois centraliza todos os estilos de um site em um mesmo arquivo. Com isso, é mais fácil controlar e padronizar as diversas páginas de um mesmo site.

Essa forma de vinculação pode ser feita de duas maneiras: a) utilizando o elemento link, ou b) usando a diretiva (Construção de algumas linguagens de programação que especificam como o compilador ou montador deve processar o código fonte: http://pt.wikipedia.org/wiki/Diretiva) @import dentro do elemento style. Em ambos os casos, esses elementos pertencem ao cabeçalho do documento (X)HTML, descrito pela tag <head>. Veja a seguir as duas formas de realizar esse tipo de vinculação.

```
a)  <head>
...
<link rel="stylesheet" href="estilo.css" type="text/css" />
</head>
```

```
b)  <head>
...
<style type="text/css">@import url("estilo.css") screen, projection; </style>
</head>
```

Mas fique atento, pois essa segunda maneira de ligar o documento (X)HTML ao CSS, através do @import, não é reconhecido por browsers mais antigos. Assim, browsers antigos podem não renderizar suas páginas da maneira que você especificou no CSS. Portanto, dê preferência ao uso da tag <link> para associar um CSS às páginas (X)HTML.

3) Incorporadas ao documento

Os estilos incorporados são declarados no corpo do documento HTML e seu efeito abrange apenas a página na qual foi declarado. Sua declaração também é feita dentro da seção <head>, através da tag <style>.

Esse tipo de declaração deve ser utilizado quando você desejar que a estilização seja feita apenas na página em que ela está escrita, por exemplo, quando o site é constituído de um único arquivo.

```
<head>
...
<style type="text/css">
h1,h2,h3{color:red;}
</style>
</head>
```

Visto que há diferentes maneiras de associar um CSS a um documento (X)HTML, é possível utilizar mais de uma maneira ao mesmo tempo. Dessa forma, você deve estar se perguntando: Como fica o efeito cascata? Qual CSS prevalecerá se eu tentar estilizar o mesmo elemento de maneiras diferentes?

Respondendo, existe a seguinte prioridade entre as definições, da mais baixa para a mais alta:

1. os padrões adotados pelo browser;
2. CSS externos;
3. CSS interno, definido na seção style do cabeçalho;
4. CSS inline, definido no próprio elemento HTML.

Assim, o estilo inline é o que tem maior prioridade, ele sobrescreverá qualquer outro. Atente também para o seguinte: se o link para o CSS externo for colocado depois da tag style do cabeçalho, então o CSS externo sobrescreverá o CSS interno.

Vídeo 06 - Tipos CSS

Identificadores e Classes

Nas seções anteriores, vimos alguns exemplos de tags (X)HTML usados como seletores. Mas não pense que apenas elementos podem ocupar tal papel. Além desses, atributos, valores, classes e identificadores também podem ser seletores.

Na **Aula 2** desta disciplina, você aprendeu a criar classes através do atributo class. Você também aprendeu a utilizar o atributo id. Veremos, a seguir, que as definições criadas através desses atributos no seu código (X)HTML podem ser usadas para definir a estilização de uma página.

Usando classes como seletores

Esse seletor é bastante utilizado nas CSS. Quando você aplica uma classe, você associa um nome a um elemento. Para facilitar seu entendimento, veja como é feita a aplicação de classes a elementos (X)HTML.

```
<h1 class="portugues">Cabeçalho 1</h1>
<h1 class="ingles">Cabeçalho 2</h1>
```

A definição do estilo desses cabeçalhos pode ser feita escrevendo o elemento (h1) seguido de um ponto e o nome da classe, da seguinte maneira:

```
h1.portugues{color:blue;}
h1.ingles{color:red;}
```

Ou ainda podemos omitir o elemento e definir apenas a classe, da seguinte maneira:

```
.portugues{color:blue;}
.ingles{color:red;}
```

No exemplo anterior, temos dois cabeçalhos de nível h1. Cada um definido em uma classe diferente. Se sua página tivesse vários cabeçalhos nível 1 (h1) e você desejasse que metade deles (os que são em portugues) fosse de cor azul e a outra metade (os que são em ingles) de cor vermelha, você poderia definir classes para facilitar essa estilização e poderia estilizá-los utilizando apenas duas regras CSS, como mostrado anteriormente.

Dessa maneira, o seletor classe possibilita aplicar estilos diferentes a um mesmo elemento (X)HTML, ou o mesmo estilo a diferentes elementos.

Vídeo 07 - Classes e seletores

Usando identificadores como seletores

O seletor id funciona de maneira idêntica ao seletor classe, quando nos referimos à estilização. No código (X)HTML, ele é bem mais específico que o atributo class e deve ser único no documento. Se a classe é cantor, o identificador (id) pode ser o cantor Luan Santana, por exemplo. Sendo assim, em termos de estilização, o identificador apenas aplicará a formatação a um único elemento (X)HTML. Abaixo, temos um exemplo de definição de um identificador.

<h1 id="TituloPrincipal" href="url">Principal</h1>

Para definir a estilização do exemplo anterior, utiliza-se o sinal # para separar o elemento do identificador.

```
h1#TituloPrincipal{background-color:black;}
Ou ainda omitir o elemento:
#TituloPrincipal{background-color:black;}
```

Atividade 03

1. Em que consiste uma regra CSS? Cite um exemplo.
2. O que é um seletor? Cite 3 exemplos, pelo menos um usando identificador e um usando classe.
3. Explique com suas palavras o que significa efeito cascata.
4. Vincule das três maneiras citadas anteriormente (inline, externa, incorporada ao documento) a seguinte regra CSS ao documento (X)HTML que foi desenvolvido por e sobre você. Cada forma de vinculação deve ser usada separadamente, ou seja, crie uma versão diferente do documento XHTML para cada uma delas, e posteriormente analise a vantagem e desvantagem de cada uma, se você decidisse modificar a estilização para outra cor de fonte.

```
h1, h2, h3 {
    color: white;
    word-spacing: 10px;
    background-color:black;
    width: 350px;
}
```

Leitura Complementar

No endereço a seguir você pode conhecer e baixar os guias de referência de XHTML e CSS <http://www.w3c.br/Materiais/PublicacoesW3C>.

Saiba de que se trata HTML 5 em: <http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf>.

Resumo

Nesta aula, você conheceu um pouco mais sobre XHTML e CSS. Com relação à XHTML, você entendeu suas particularidades, conheceu as características que a diferenciam de HTML, aprendeu também os elementos obrigatórios do XHTML, a DTD XHTML, a marcação semântica e a validação do XHTML. Com relação à CSS, você estudou sua história e sintaxe, aprendeu a fazer a vinculação das folhas de estilo a um documento e a usar identificadores e classes.

Autoavaliação

1. Relacione as diferenças entre HTML e XHTML.
2. Na aula passada, você desenvolveu pelo menos dois documentos HTML, um sobre você e outro sobre esportes ou hobbies. Agora você deve transformar a especificação HTML daqueles documentos em XHTML. Atente para as principais diferenças citadas nesta aula, incluindo a utilização da declaração doctype apropriada. Ao final, utilize o validador de XHTML (disponibilizado pela W3C) para verificar suas especificações e garanta que não haja nenhum erro. Não se preocupe se alguns elementos não puderem ter a mesma aparência que você definiu em HTML através de elementos de formatação proibidos em XHTML, pois na próxima aula você começará a aprender CSS para formatá-los apropriadamente.
3. Qual a ordem de precedência entre as declarações CSS do usuário, do autor e do browser?

4. O que é um seletor? Dê 5 exemplos.

5. Explique duas maneiras pelas quais você pode associar um código CSS a um documento (X)HTML e quais as vantagens e desvantagens de cada uma.

Referências

SILVA, Maurício Samy.**Construindo sites com CSS e (X)HTML:** sites controlados por folhas de estilo em cascata. São Paulo: Novatec Editora, 2008.

TUTORIAL XHTML. Disponível em: <http://maujor.com/tutorial/xhtml.php>. Acesso em: 2 set. 2015.

W3c. Disponível em: <http://www.w3c.org/>. Acesso em: 2 set. 2015.

W3SCHOOL. Disponível em: <http://www.w3schools.com/>. Acesso em: 2 set. 2015.

W3C Br. **O W3C, Futuro da Web, HTML5, Web Semântica**. SENAC Ribeirão Preto - Road Show TI, 2011. Disponível em http://www.w3c.br/pub/Agenda/PalestraSenacRibeiraoHTML5WebSemantica/W3C_HTML5_WebSem.pdf. Acesso em: 2 set. 2015.