Conhecimentos em HTML 5, CSS 3, Metodologia CSS (OOCss, BEM, Smacss)

HTML 5

Ao ser anunciada a versão 5 do padrão HTML, inicialmente parte do mercado não recebeu com grande entusiasmo, acostumado a receber poucos recursos novos de uma versão para outra. A verdade é que, desde que a versão 4.0 foi lançada em 1997, poucos avanços aconteceram nos dez anos seguintes. O padrão foi atualizado para 4.01 (praticamente uma errata) e o padrão XHTML foi criado e posteriormente atualizado para 1.1 – padrão este que se resume na HTML 4.01 com algumas variações em XML. A World Wide Web permaneceu praticamente estática neste período por várias razões.

Uma destas razões é que o W3C (órgão que regulamenta os padrões Web, entre eles a HTML) tem um ciclo de versão demorado, que exige versões de rascunho e períodos de contribuição da comunidade, sugerindo novos conceitos e posteriormente validando, concordando e discordando dos mesmos – processos estes que levam tempo.

Em segundo lugar, os fabricantes de navegadores web demoravam muito a adotar o novo padrão, por muitas vezes adotando-o parcialmente – isso quando não fugiam dele, criando elementos e conceitos particulares, tornando-o um website aderente apenas a este ou aquele navegador web. Alguns fabricantes atualizavam seus navegadores web e padrões suportados apenas na troca do sistema operacional, o que poderia levar até três anos para acontecer. Isso sem falar nos navegadores do mundo mobile.

E, finalmente, como se os dois primeiros obstáculos não fossem o bastante, os internautas não atualizavam suas versões de navegador web – seja por inexperiência ou descaso – permanecendo com versões antigas que não davam suporte ao novo padrão. O absurdo chega ao ponto que o Microsoft Internet Explorer em sua versão 6.0 era o navegador web mais utilizado na Internet Brasileira, mais de 10 anos após seu lançamento.

Como resposta à lentidão do W3C, uma comunidade paralela formada por profissionais da Mozilla Foundation (fabricante do Firefox), Opera Software (navegador Opera) e Apple (Safari) foi criada em 2004 com o nome de WHATWG (Web Hypertext Application Technology Working Group), com o intuito de discutir novos padrões e recursos para a Web.

Recebendo posteriormente contribuições de outras empresas como Google e Microsoft rapidamente o novo padrão foi ganhando forma de tal maneira que, três anos depois, foi submetido ao W3C que por sua vez decidiu adotá-lo batizando-o de HTML5. O novo padrão ajudou a enterrar o XHTML 2.0, padrão que o W3C estava trabalhando em 2007, considerado por muitos um equívoco – o padrão não era sequer compatível com a versão 1.1.

À primeira vista, a HTML5 parece se tratar meramente de um conjunto de novas tags para renderização de texto e formulários, impressão essa reforçada ao se folhear a maioria dos livros disponíveis hoje no mercado. Não se engane. Estas tags mencionadas são apenas a ponta do iceberg.

Embora seja chamada de HTML5 e a sigla signifique HyperText Markup Language, ou seja, linguagem de marcação de hipertexto, as novidades vão além disso. A HTML5 é um grande guarda-chuva tecnológico, pois pendurados a ele estão o novo padrão CSS na versão 3 e uma imensa gama de novas APIs na linguagem JavaScript estendendo as funcionalidades e possibilidades da WWW para patamar absolutamente impressionantes.

Diferenças da HTML4

Este artigo não estaria completo se não discorrêssemos acerca das diferenças entre a nova HTML5 e a sua antecessora, a HTML4, esta que ocupou lugar de destaque no mundo web por muitos anos. Ainda teremos um bom tempo de espera até que a HTML5 se torne de fato um padrão na web e que todos os browsers e tecnologias afins assumam a mesma como centro de implementação. Como programador web, é muito interessante que saiba quais as principais diferenças entre ambas as versões, justamente para que possa salvar tempo e aumentar a produtividade em situações como essa.

Uma das características mais marcantes dessa nova versão da linguagem é o fato de que a mesma não é uma versão final, isto é, estará sempre e continuamente mudando ao longo do tempo. Isso inclui dizer que os desenvolvedores da linguagem estarão sempre adicionando e removendo atributos, tags e o que considerarem interessantes à mesma. Ao mesmo tempo, constitui um risco se você estiver usando a mesma como algo definitivo no seu projeto. Isso significa que se optar pela HTML5 terá de seguir suas atualizações e estar constantemente evoluindo seu código também.

A HTML5 foi feita para ser simples, isso implica em uma sintaxe extremamente mais simples e limpa. A simples declaração do doctype foi apenas mais uma das facilidades incluídas na nova versão. Agora, você precisa inserir apenas um <!doctype html> no início do seu documento e tudo estará pronto. Além disso, a sintaxe da HTML5 é compatível também com a HTML4 e XHTML1.

A linguagem apresenta também um elemento novo, que veremos aqui no artigo, o <canvas>, responsável por substituir muitas das implementações antes feitas em Flash, o que faz muitos desenvolvedores considerar que este já se encontra obsoleto e futuramente morto.

A extensão de tags a um tool de novos e interessantes recursos fez uma grande diferença na linguagem. Tags como: <header> e <footer>, que estendem a funcionalidade de tabelas agora para a página como um todo, <section> e <article>, que permitem marcar áreas específicas dos layouts, <video> e <audio> para uma inclusão melhorada de conteúdos multimídia nas páginas, e <menu> e <figure> para bem arranjar textos, imagens e menus, trazem todo um conjunto de implementações e funcionalidades bem pertinentes para a web de hoje.

Além disso tudo, a remoção de alguns outros recursos como as tags , <big>, <font>, etc fazem com a responsabilidade do CSS aliado à nova linguagem só aumente, otimizando o desenvolvimento front-end.

Convertendo de versões antigas

Para converter códigos poliglotas e obsoletos para a nova versão da HTML5, basicamente temos de seguir três passos:

1. Remova os identificadores PUBLIC FPI e SYSTEM da declaração do DOCTYPE;

2. Substitua quaisquer tags depreciadas da HTML ou construa sua implementação baseada num código HTML que seja de acordo com os padrões da HTML5. Por exemplo, tenha certeza de assegurar que qualquer conjunto de tags <col> para colunas de tabelas HTML sempre tenham um elemento "colgroup" como seu parente de configuração;

3. Comece a tirar vantagem dos novos recursos da HTML5, tais como convertendo suas div's para as tags de seção da HTML5.

Mas e se necessitarmos realizar conversões de versões mais antigas ainda da HTML? Esse tipo de conversão requer um pouco mais de trabalho e atenção, uma vez que ocorreram notórias mudanças nas versões da HTML entre os anos 1997 e 2000 com o objetivo de suportar conversores baseados em XML, dispositivos móveis com regras de conversão bem restritas, templates client-side cacheáveis além de agregação com outros tipos de conteúdo como os velhos feeds RSS. Vejamos mais alguns passos para tal:

1. Confira sempre se o documento inicia com uma declaração XML e uma declaração de DOCTYPE;

2. Verifique se o elemento de top <html> inclui o atributo "xmlns="http://www.w3.org/1999/xhtml"". (A URI para XHTML e versões mais recentes da HTML incluem o ano "1999" porque foi definido aquele ano enquanto a W3C HTML Recommendation lançada em 2000 estava ainda em fase de desenvolvimento.);

3. Verifique se todas as tags estão marcadas com a tag de fim, ou se estão todas "self-closed" com o sinal de />. Não se esqueça de averiguar se os nomes dos elementos iniciam e terminam com a marcação em minúscula.

4. Verifique se todos os valores de atributos estão cercados pelas aspas. Verifique também se os atributos booleanos estão codificados na sua forma full, usando o atributo name entre aspas como o

valor (attribute="attribute") quando o mesmo valor for true e o omitindo completamente quando o valor for false.

5. A forma full será apropriadamente entendida pelos web browsers que convertem documentos com a ainda sintaxe HTML ou a sintaxe XML do HTML5. Evite usar formas minimizadas para selected="", a qual o XPath trata como false ao invés de true.

Observe que a especificação HTML5 explicita estado como:

Os valores "true" e "false" não são permitidos em atributos booleanos

Isso acontece porque os browsers olham para o código com o valor booleano para os atributos e tratarão a string "false" como um valor false enquanto browsers que somente olham para a presença ou ausência do atributo tratarão esse código como true, resultando em comportamentos inconsistentes e confusos.

Atributos booleanos que precisam ser mudados incluem alguns dos tipos ilustrados na Tabela 1.

Tabela 1. Lista de atributos e respectivas mudanças de tipo booleano

Atributo	Mudança
Async	Mudar para async=”async”
Checked	Mudar para checked=”checked”
Compact	Mudar para compact=”compact”
Declare	Mudar para declare=”declare”
Defer	Mudar para defer=”defer”
Disabled	Mudar para disabled=”disabled”
Ismap	Mudar para ismap=”ismap”
Multiple	Mudar para multiple=”multiple”
Noresize	Mudar para noresize=”noresize”
Noshade	Mudar para noshade=”noshade”
Nowrap	Mudar para nowrap=”nowrap”
Open	Mudar para open=”open”
Readonly	Mudar para readonly=”readonly”
Required	Mudar para required=”required”
Reversed	Mudar para reversed=”reversed”
Scoped	Mudar para scoped=”scoped”
Selected	Mudar para selected=”selected”

Observe que “true” e “false” são valores válidos para alguns atributos “não booleanos”, em particular atributos enumerados que o atributo draggable.

Outra forma de se trabalhar migrando esse tipo de documento é verificando qual o tipo de versão que foi usada para construir o documento HTML. Uma boa forma de se fazer isso é submeter a URL do website ao site do W3C: Markup Validation Service. Ao executar, os possíveis resultados são:

• HTML5: Indica que o site já foi convertido para o padrão HTML5. Por exemplo, o próprio site do Google é um bom exemplo de teste que usa o padrão.

• XHTML 1.0 Strict: Indica que o site está usando a versão padrão 2000 W3C da HTML. Por exemplo, o próprio site do W3C adere à essa versão.

• XHTML 1.0 Transitional or HTML 4.01 Transitional: Indica que o site está usando o formato transicional entre a versão padrão da HTML4 de 1997 e a versão padrão da HTML de 2000. Por exemplo, o site AltaVista.com usa o formato da HTML 4.01 Transitional, já o site da Microsoft usa XHTML 1.0 Transitional.

• HTML 4.01 Strict: Indica que o site está usando a versão antiga da HTML4 de 1997. Por exemplo, o site do Yahoo usa o antigo padrão.

As Oito Áreas da HTML5

O novo padrão é vasto. Por esta razão, O W3C dividiu nestas novidades em oito áreas tecnológicas. A divisão ajuda na homologação do padrão e suporte dos fabricantes, que estão sendo feitos em porções. Estas áreas são:

• Semantics (Semântica): A ponta do iceberg. Nesta área, estão as novas tags que auxiliam na análise semântica dos textos presentes nas páginas. Muito úteis especialmente para melhorar a eficiência dos mecanismos de busca, como o Google. Adicionalmente, novas tags para renderização de formulários, em destaque caixas de texto com validação nativa.

• Offline & Storage (Fora do Ar e Armazenamento): Aborda todas as funcionalidades referentes ao tratamento do website quanto o visitante estiver em modo offline, além das novidades em armazenamento de informações no navegador como o LocalStorage e o Banco de Dados IndexedDB, indo mais além dos famigerados cookies.

• Device Access (Acesso por Dispositivos): Compreende todas as APIs que estendem a experiência de visitantes de dispositivos móveis como tablets e smartphones, como a Geolocalização (usando GPS ou outros recursos para determinar onde está o usuário), o uso do acelerômetro (utilizado para determinar a orientação do dispositivo no espaço), o uso do microfone e câmera destes dispositivos, além de eventos específicos para telas sensíveis a toque.

• Connectivity (Conectividade): Avanços incríveis na conectividade de aplicações web, como a comunicação usando WebSockets e SSE (Server Side Events), essenciais especialmente em jogos eletrônicos.

• Multimedia: Novas tags para a publicação de áudio e vídeo na Internet, além de novos formatos multimedia suportados.

• 3D, Graphics & Effects (3D, Gráficos e Efeitos): Engloba aqui o Canvas que permite desenhar elementos gráficos em uma página, como uma tela de desenho. Adicionalmente, o suporte a WebGL para renderizações em 3D, o suporte ao formato SVG e efeitos 3D obtidos através de CSS3.

• Performance & Integration (Performance & Integração): Melhorias significativas na manipulação e submissão de formulários pela Internet. Possibilidades de processamento paralelo utilizando os novíssimos Web Workers, além de melhorias integração com o visitante (interface) com o excelente recurso de Drag 'n Drop (arrastar-e-soltar).

• CSS3: Novíssimos estilos e efeitos sem sacrificar performance ou indexação dos mecanismos de busca.

CSS3

CSS é chamado de linguagem Cascading Style Sheet e é usado para estilizar elementos escritos em uma linguagem de marcação como HTML. O CSS separa o conteúdo da representação visual do site. Pense na decoração da sua página. Utilizando o CSS é possível alterar a cor do texto e do fundo, fonte e espaçamento entre parágrafos. Também pode criar tabelas, usar variações de layouts, ajustar imagens para suas respectivas telas e assim por diante.

CSS foi desenvolvido pelo W3C (World Wide Web Consortium) em 1996, por uma razão bem simples. O HTML não foi projetado para ter tags que ajudariam a formatar a página. Você deveria apenas escrever a marcação para o site.

Tags como <font> foram introduzidas na versão 3.2 do HTML e causaram muitos problemas para os desenvolvedores. Como os sites tinham diferentes fontes, cores e estilos, era um processo longo, doloroso e caro para reescrever o código. Assim, o CSS foi criado pelo W3C para resolver este problema.

A relação entre HTML e CSS é bem forte. Como o HTML é uma linguagem de marcação (o alicerce de um site) e o CSS é focado no estilo (toda a estética de um site), eles andam juntos.

CSS não é tecnicamente uma necessidade, mas provavelmente você não gostaria de olhar para um site que usa apenas HTML, pois isso pareceria completamente abandonado.

Vantagens do CSS

A diferença entre um site que implementa CSS e outro que não o usa é gigantesca e notável.

Você já deve ter visto um site que não carrega completamente ou tem um plano de fundo branco com texto azul e preto. Isso significa que a parte CSS do site não foi carregada corretamente ou não existe.

E é assim que um site somente com HTML se parece. Acredito que você vai concordar comigo de que isso não é muito bonito, certo?

Antes de usar CSS, toda a estilização tinha que ser incluída na marcação HTML. Isso significa que você deveria descrever separadamente todo o plano de fundo, as cores das fontes, os alinhamentos, etc.

Mas o CSS permite que você estilize tudo em um arquivo diferente, criando assim o estilo separadamente. E, mais tarde, faça integração do arquivo CSS na parte superior da marcação HTML. Isso mantém a marcação HTML limpa e fácil de manter.

Resumindo, com o CSS você não precisa mais escrever repetidamente como os elementos individuais se parecem. Isso economiza tempo, encurta o código e diminui a chance de erros.

O CSS permite que você tenha vários estilos em uma página HTML, tornando as possibilidades de personalização quase infinitas. Hoje em dia, isso está se tornando mais uma necessidade do que um simples recurso.

Como CSS Funciona

O CSS é uma ferramenta muito potente que possibilita criar diversas funcionalidades ao invés de usar JavaScript ou outra linguagem mais pesada. Se usado com moderação, CSS pode viabilizar uma ótima experiência ao desenvolvedor e usuários das páginas web.

Com o Cascading Style Sheets é possível criar animações complexas, criar efeitos com uso de parallax, que faz parecer que a imagem de fundo tem uma profundidade diferente um dos outros, criar sites interativos e também jogos com HTML5 e CSS3.

O CSS usa uma sintaxe simples baseada em inglês com um conjunto de regras que o governam. Como mencionamos anteriormente, o HTML nunca teve a intenção de usar elementos de estilo, apenas a marcação da página. Foi criado para descrever apenas o conteúdo. Por exemplo: Este é um parágrafo. .

Mas como você estiliza o parágrafo? A estrutura da sintaxe CSS é bem simples. Tem um seletor e um bloco de declaração. Você seleciona um elemento e depois declara o que deseja fazer com ele. Bastante simples, certo?

Mas tem algumas regras que você precisa saber. Isso também é simples, não se preocupe.

O seletor aponta para o elemento HTML que você deseja estilizar. O bloco de declaração contém uma ou mais declarações separadas por ponto e vírgula.

Cada declaração inclui um nome de propriedade CSS e um valor, separados por dois pontos. Uma declaração CSS sempre termina com um ponto-e-vírgula e os blocos de declaração são cercados por chaves.

Vamos ver um exemplo:

Todos os elementos <p> serão estilizados e serão coloridos de azul e negrito.

```
1.  <style>
2.
3.  p {
4.  color: blue;
5.  text-weight: bold;
6.  }
7.
8.  <style>
```

Em outro exemplo, todos os elementos <p> serão centralizados, com tamanho 16x e de cor pink.

```
1.  <style>
2.  p {
3.
4.  text-align: center;
5.  font-size: 16px;
6.  color: pink;
7.
8.  }
9.  </style>
```

Anatomia de um comando CSS

O CSS estipula regras para o arquivo em html. Com cada regra é possível estilizar o conteúdo todo ou somente determinados elementos. Por isso entenda, um comando básico é composto por seletor e declarações, que contém propriedade e valor.

Seletor {Propriedade: Valor}

A sintaxe do CSS é muito simples de aprender. O seletor seleciona quais elementos em html receberão a propriedade. Pode ser p (parágrafo) ou o body (corpo da sua página). Já a propriedade pode ser a cor ou algo mais específico como cor do fundo (background). E por último o valor, que determina o valor da propriedade.

Vamos simular um exemplo. Digamos que o objetivo é mudar a fonte de uma tag h1. Para isso podemos usar h1 {font-size: 20px;}

• h1 – é o seletor. Neste caso selecionamos o h1.

• font-size – é a declaração que contém a propriedade (font-size) e o valor é (20px).

Metodologia CSS

SMACSS

O sistema estabelece e é bastante baseado em cinco categorias de regras de CSS: base, layout, module, state e a pouco importante theme. As regras de base são as do tipo que não utilizam seletores com classes ou ids, as encontramos em um CSS Reset ou normalize.css.

O sistema alerta sobre a agressividade dos CSS Resets mas não alerta sobre as regras deste tipo definidas no próprio projeto, ainda mais quando aplicadas a divs, spans ou headings. Regras cujos seletores não utilizam classes são globais e qualquer decisão tomada neste nível irá perpetuar por todo o projeto, cuidado.

As categorias de layout e module são bastante semelhantes. Pense no layout como elementos agregadores e geralmente únicos como header, footer e sidebar. O sistema propõe que regras de layout tenham ids ou classes com o prefixo. l- como seletores.

As regras da categoria module englobam os demais componentes da página. O sistema não encoraja o uso de elementos nos seletores, preferindo. box. title ao invés de .box h2. Ainda, o seletores como. box-title são defendidos para facilitar a leitura do HTML.

Assim como o OOCSS, o sistema repudia regras do tipo #sidebar. media onde a localização do elemento passa a ser relevante para sua apresentação. O SMACSS reforça que seja adicionada uma classe para abrigar as variações. O elemento da sidebar passa a ter a classe do módulo e também a do sub-módulo: .

A categoria de state engloba regras responsáveis por gerenciar estado de componentes enquanto o usuário estiver navegando. Regras desta categorias são as únicas que podem e talvez precisem utilizar !important. O padrão indica que as classes possuam o prefixo. is-. Com certeza, algum dos seus projetos já precisou de uma classe como. is-active, .is-collapsed ou is-current.

O SMACSS é mais uma série de tutoriais de como escrever um bom código que propriamente um sistema de CSS. Contra os padrões, não concordo com qualquer aparição de #id em folhas de estilos por ir contra os preceitos de reuso.

O atributo id na realidade serve mais como um destino de navegação, por isto a necessidade de ser único. Considero um pouco estranho também o prefixo l- para as classes de layout, o que me atreve a desconsiderar totalmente a categoria e a gerir suas regras como se pertencessem a categoria module. Não existe necessidade desta distinção nos seletores, apenas separar as regras em um arquivo layout.css já é suficiente.

A categoria de state é a mais interessante, o padrão fica muito conveniente para ser utilizado em código JavaScript. Quando o estado de um componente demanda regras muito específicas, o SMACSS sugere o seletor .is-tab-active. Desta maneira, o JavaScript perde um tanto da sua modularidade, seletores aninhados como. tab.is-active podem ser uma melhor jogada.

Por fim, mesmo sendo bastante válido, o SMACSS não soluciona alguns desafios típicos do design de componentes de médio porte pois em nenhum momento endereça como nomear adequadamente elementos descendentes.

Bem

O BEM – sigla para block, element, modifier – é uma metodologia com várias versões cujo o preceito de esclarecer o desenvolvedor mais sobre o markup através de suas classes. Este sistema permite escrever sites de maneira rápida, auto-explicativa e com manutenção descomplicada. Esqueça seus preconceitos com os caracteres duplos de hifen, eu já deixei o meu de lado, e reflita a seguir o quanto mais de informação a classe. report-graph__bar_size_bigoferece em relação as tradicionais .bar, .report-graph-bar ou .graph-bar.

O block é uma entidade independente da aplicação, podendo ser o mais alto nível de abstração (header, footer) ou componente (graph, tabs). O element é um descendente dependente de um block que possui uma certa função. Para permitir nomes compostos e evitar ambiguidades, o padrão estabelece o controverso __ como separador. Desmembrando a classe report-graph__bar, identificamos bar como element e sabemos da existência do elemento pai report-graph, que é o block.

O estilo define o modifier como uma propriedade de um block ou element que altera sua aparência. Desta maneira, o block . menu poderia ser acrescido da classe .menu_size_big, note o uso de _.

A fim de deixar o padrão um pouco mais claro, veja um exemplo:

```
<div class="report-graph">

  <div class="report-graph__bar">...</div>

  <div class="report-graph__bar report-graph__bar_size_big">...</div>

</div>
```

As classes .report-graph, report-graph__bar e report-graph__bar_size_big são, respectivamente, block, element e modifier. Neste ponto você já deve ter refletido e concluído que sim, há muito mais informação numa classes nomeada neste padrão.

Uma das falhas do estilo é não possuir categorias como as do SMACSS. Segundo o padrão, o estado de um componente deve ser endereçado como um modifier, não há uma categoria de state. Conforme citei anteriormente, classes modifier como. menu__item_state_current tiram a modularidade do JavaScript fazendo com que o código dependa do componente. Alguns desenvolvedores também podem sentir falta do estilo não versar nada sobre regras aplicadas diretamente a elementos (categoria base do SMACSS) e boas práticas de CSS.

O grande trunfo do padrão é mesmo sugerir uma nomenclatura adequada para elementos descendentes. Outro aspecto interessante é o padrão de organização de arquivos.

DRY CSS

O princípio consiste em não repetir propriedades com mesmos valores em seu código. De maneira simples, a todo momento que isto for necessário, estas propriedades devem ser agrupadas e endereçadas por vários seletores. O sistema define que o agrupamento seja nomeado através de um primeiro seletor, algo como #MEDIUM-WHITE-BACKGROUND para um agrupamento de background e border brancas. As custas deste primeiro seletor, muitos desenvolvedores confundem que o pattern sugere que este seletor seja usado no HTML, o que não é verdade.

A técnica, assim como outros sistemas, sugere que seu código seja pensando em termos de padrões de aparência. O problema está no fato da técnica não sugerir melhor uso das classes de CSS. Segundo o sistema, sempre que a aparência de um elemento mudar, sua classe precisará ser movida para outros agrupamentos nas folhas de estilo. Sistemas como SMACSS e BEM endereçam mudanças de aparência com a criação de um submódulo ou modificador que será adicionado ao HTML evidenciando a alteração e permitindo manter a antiga aparência.

Não entenda mal, os conceitos por trás do DRY CSS são válidos. Inclusive, já sugeri uma variação da técnica em conjunto com placeholders do SASS para nomear propriedades que definem um elemento da aparência do seu projeto. Os placeholders carregam os conceitos de DRY CSS e ainda permitem basear sua arquitetura em outros sistemas mais poderosos.