MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE BENTO GONÇALVES

DESENVOLVENDO SITES ACESSÍVEIS

- Manual do Desenvolvedor -

SIEP

NÚCLEO CEFET-BG

Dezembro de 2007.

ÍNDICE

SOBRE OS AUTORES

ANDRÉA POLETTO SONZA

Graduada em Ciência da Computação, pela Universidade de Caxias do Sul – RS; Especialista em Psicopedagogia Institucional pela Universidade do Sul de Santa Catarina – SC; Mestre em Educação, Doutoranda em Informática na Educação; Educadora Especializada do Centro Federal de Educação Tecnológica de Bento Gonçalves RS, desenvolvendo atividades de coordenação do Núcleo de Atendimento às Pessoas com Necessidades Especiais (NAPNE), Núcleo da Ong RedEspecial Brasil, Infocentro Acessível e Núcleo do SIEP (Sistema de Informação da Educação Profissional e Tecnológica), além de atuar em cursos de Informática para deficientes visuais.

ANDRÉ GUSTAVO ESPEIORIN

Desenvolvedor Freelancer multiplataforma, Graduando em Ciência da Computação, atua na área de desenvolvimento web desde março de 2006, onde seus destaques estão na análise de sistemas, programação PHP OO e Java, além de desenvolvimento XHTML/CSS, autodidata em linguagens de programação, acessibilidade e padrões.

CARLOS ALBERTO TRISTACCI

Coordenador de Projetos Web da Infowebdesign é certificado como Adobe Certified Professional Flash Developer. Graduando em Administração com ênfase em Marketing, trabalha na área de internet desde 2003, na qual se destaca pelos conhecimentos nas tecnologias Flash e Flex e na linguagem ActionScript. Ministra aulas de webdesign desde 2004, é articulista do portal imasters.com.br e participa de projetos do MEC, no desenvolvimento de sites e sistemas acessíveis, seguindo as recomendações da W3C.

CAPÍTULO 1

Web para Todos

Devido a limitações sensoriais, cognitivas ou físicas, algumas pessoas são impossibilitadas de acessar os recursos de hardware ou software que o mundo digital oferece (Hogetop e Santarosa, 2002). Para compensá-las, existem próteses chamadas de Tecnologia Assistiva (TA) ou Ajudas Técnicas (AT), dependendo da influência norte-americana ou européia, respectivamente. Seu conceito refere-se ao conjunto de artefatos disponibilizados às pessoas com necessidades especiais (PNES), que contribuem para proporcionar-lhes uma vida mais independente, com mais qualidade e possibilidades de inclusão social (Bersch e Tonolli, 2006).

Mas apesar das inúmeras vantagens que tais ferramentas fazem emergir, novos obstáculos são impostos às pessoas que possuem alguma limitação, dificultando e, até mesmo, impossibilitando acesso aos ambientes virtuais. O que ocorre é que usuários que possuem limitações, ao interagirem em sites, portais e demais ambientes virtuais, muitas vezes têm dificuldades de acesso, navegação ou não compreendem as informações veiculadas. Nossa contribuição nesse artigo refere-se aos conceitos de qualidade de uso de sistemas, norteados pelas diretrizes do W3C (World Wide Web Consortiun) e sugestões para a construção de ambientes acessíveis, com uma boa usabilidade e comunicabilidade, especialmente para usuários deficientes visuais. O tributo desses últimos foi e tem sido fundamental para a modelagem de sistemas que realmente permitem o acesso, a navegação e comunicam de forma eficaz seu conteúdo.

Assim, o CEFET Bento Gonçalves, por ser o Núcleo de Acessibilidade do Sistema de Informações da Educação Profissional e Tecnológica vem trazendo esses conceitos para seu trabalho de testes e auxílio na acessibilização dos sites e portais do domínio MEC.

1) TECNOLOGIAS ASSSITIVAS

Como já mencionado, algumas pessoas precisam utilizar auxílios para ter acesso ao computador e, consequentemente, à web. Esses dispositivos/programas são também referenciados como Agentes de Usuário nas diretrizes do W3C. O Agente de usuário refere-se ao hardware ou software utilizado para acesso ao conteúdo web. Inclui navegadores gráficos, navegadores de texto, navegadores de voz, celulares, leitores de multimídia, suplementos para navegadores, além de leitores de tela e programas de reconhecimento de voz.

Dentre as TAs para usuários com limitações visuais destacamos o Dosvox , interface que se comunica com o usuário, em português, por meio de síntese de voz e os leitores de tela. Esses últimos são programas que interagem com o Sistema Operacional, reproduzindo, de forma sonora, os eventos ocorridos no computador. Virtual Vision, Jaws e Orca são três leitores de tela, com síntese em português, bem aceitos no Brasil. Já o Terminal ou Linha Braille é um equipamento eletrônico que possui uma linha régua de células Braille, cujos pinos se movem para cima e para baixo e representam uma linha de texto da tela do computador. Pode ser utilizado inclusive por usuários surdocegos.

Pessoas com limitações motoras também podem fazer uso de tecnologias assitivas, como os teclados adaptados, de acordo com suas especificidades. Alguns exemplos de teclados diferenciados são: ampliado, reduzido, de conceitos, para uma das mãos, ergonômico, dentre outros. Esses usuários podem também utilizar a colméia, que é uma placa de plástico ou acrílico com um orifício correspondente a cada tecla, que é fixada sobre o teclado (Damasceno e Filho, 2002). Outros exemplos são pulseiras de pesos, apontadores de cabeça e mouses e acionadores diversos. Dentre esses destacamos o mouse de ocular (Projeto Mouse Ocular, 2005), o mouse de sopro (Jouse, 2006), o mouse de nariz ou HeadDev (Ajudas.Com, 2006) e o acionador de pedal (Ausilionline.it, 2006).

Usuários com limitações motoras também podem fazer uso de simuladores de teclado, que são programas que simulam um teclado na tela do computador. Pessoas com tetraplegia ou limitações motoras severas podem utilizar o Motrix. O sistema permite que o usuário forneça comandos de voz para a maior parte das funções do computador. (Projeto Motrix, 2002).

Após apresentarmos alguns agentes de usuário utilizados por pessoas com limitações visuais ou motoras - informações importantes para justificarmos a necessidade de uma web verdadeiramente acessível - passamos a referenciar a semântica na web além de conceitos de qualidade de uso de sistemas.

2) PADRÕES DE DESENVOLVIMENTO WEB E WEB SEMÂNTICA

Quando tratamos de definição e arquitetura para implementação de interfaces web, sabemos que atualmente diferentes formatos de arquivos podem ser disponibilizados na rede; mas tudo começou com o HTML. Conforme Silva (2007), o embrião dessa linguagem de marcação surgiu para servir a uma comunidade bastante restrita, a comunidade de cientistas. Com a introdução gradativa de novas tags, atributos e aplicações específicas, essa linguagem tornou-se padrão mundial de apresentação de conteúdo na web. E "a velha linguagem de marcação passou a exercer uma dupla função: estruturar o conteúdo através da marcação e apresentá-lo, ou seja, dar a aparência final" (SILVA, 2007). Só que essa dupla função começou a causar problemas: os documentos publicados na Internet, cada vez mais sofisticados e extensos, estavam fugindo do controle de seus criadores (ibidem).

Essa problemática ocorre porque o HTML não foi concebido para usos tão amplos quanto aquele que as tecnologias atuais requerem, sendo limitado no que tange à aplicação de forma ao documento. Para solucionar esse problema os desenvolvedores web passaram a utilizar técnicas não comuns de uso dos comandos HTML, como: tabelas com bordas transparentes para dispor os elementos na página, uso de comandos que não eram padrão no HTML para efeitos de formatação, dentre outros. Acontece que "essas 'trapaças' causaram problemas nas páginas na hora de sua visualização em distintas plataformas" (CRIARWEB, 2008). Além disso, essa mistura entre conteúdo e apresentação tornou-se uma grande dor de cabeça aos desenvolvedores (SILVA, 2007). Só para dar um exemplo: se tivessem que alterar a cor de todos os títulos de um site com 180 páginas teria que fazê-lo em cada uma das linhas que apresentasse esses títulos. O tempo gasto para essa alteração, que parece tão simples, acabava sendo bastante grande. A solução encontrada foi dissociar linguagem de marcação da estilização. Surgiram assim as chamadas Folhas de Estilo.

As Folhas de Estilo em Cascata (Cascading Style Sheets) ou CSS referem-se ao conjunto de declarações que especificam a apresentação do documento. Trata-se de uma linguagem de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML. Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento. Trata-se de um arquivo, independente do arquivo HTML, no qual são declaradas todas as propriedades e valores de estilização para os elementos do HTML (SILVA, 2007). O efeito cascata das folhas de estilo refere-se ao estabelecimento de uma prioridade para aplicação de uma regra de estilo a determinado elemento ou grupo de elementos (SILVA, 2007).

Tangarife e Montalvão (2006) referem que a utilização do HTML juntamente com folhas de estilo para publicação de conteúdo na web, conforme recomendações do W3C podem ampliar o acesso à informação. Assim, "codificação correta e uso adequado das marcações HTML são condições necessárias ao desenvolvimento de tecnologias web-acessíveis, bem como a separação entre estrutura e apresentação" (TANGARIFE e MONT'ALVÃO, 2006). O exposto pelos autores refere-se aos web Standards ou padrões de desenvolvimento web. Um site projetado de acordo com esses padrões deve estar em conformidade com as normas HTML, XML, XHTML, CSS, etc e com o código de programação válido, acessível, semanticamente correto e amigável. Esses autores destacam alguns pontos primordiais, quando do desenvolvimento de sistemas web, quais sejam: a codificação correta e uso adequado das marcações XHTML (tags); a utilização de Tableless, ou seja, metodologia que não utiliza tabelas para a construção de layout; a separação entre layout e conteúdo, levando em consideração a semântica do código (X) HTML. Nesse contexto, separa-se a informação da formatação - a informação da interface é apresentada em (X) HTML e a sua formatação é apresentada por meio de CSS (folhas de estilo).

Segundo Pereira (2006), escrever algo semanticamente correto, nada mais é do que utilizar-se desses símbolos, ou tags, considerando o significado real pelo qual foram criados, ou seja, utilizar a tag certa no lugar certo. "E utilizar as tags no sentido correto é igual a 'código semântico', que por sua vez justifica o termo 'web Standards'. Seguir os 'web Standards' é respeitar a semântica" (PEREIRA, 2006).

3) ACESSIBILIDADE À WEB

De acordo com Cifuentes (2000), Caplan (2002) e Dias (2003), entende-se por acessibilidade à rede a possibilidade de qualquer indivíduo, utilizando qualquer tipo de tecnologia de navegação (navegadores gráficos, textuais, especiais para cegos ou para sistemas de computação móvel), poder visitar qualquer site e obter um total e completo entendimento da informação contida nele, além de ter total e completa habilidade de interação.

Se formos pensar nas vantagens relacionadas à acessibilidade, podemos destacar:

- Quantidade de usuários com alguma limitação: De acordo com a OMS (Organização Mundial de Saúde), 10% da população mundial possui alguma deficiência. Em países subdesenvolvidos como o Brasil, esse percentual pode chegar a 14,5%. Assim, o Brasil, que possui uma população aproximada de 180 milhões de brasileiros, teria cerca de 25,9 milhões de PNES.

- Referindo-nos ao mundo dos negócios, podemos dizer que consumidores deficientes (assim como qualquer outro) são inclinados a realizá-los onde são bem-vindos. Além disso, designs acessíveis são mais fáceis de serem utilizados por qualquer usuário, independente de possuir ou não alguma limitação.

- Um portal web acessível é indexado de forma mais rápida e precisa pelos mecanismos de busca. Isso faz com que os usuários o localizem com maior rapidez e facilidade. Triacca (2007) refere que quanto melhor a colocação do site, mais visitas ele terá. Segundo ele, o Google determina os sites que aparecerão melhor posicionados no resultado de nossas pesquisas, visitando semanalmente nosso site, e, quanto mais atualizados ele estiver, melhor classificação na busca ele terá. Só que o Google precisa conseguir ler o site. E para isso ele precisa de conteúdo, muito conteúdo e a melhor forma de conseguir isso é por meio do uso de pouco código na marcação, "e para isso existem os Web Standards [...] que separam estruturação de estilização" (TRIACCA, 2007). Assim, quanto mais acessível for o site, melhor cotado ele será pelo Google e, conseqüentemente, mais visitas terá.

- Adotar recomendações de acessibilidade faz com que o portal seja acessado tanto pelas tecnologias mais modernas, como a computação móvel, por exemplo, como pelas mais antigas, atingindo assim um maior contingente de visitantes.

- Razões pessoais também devem ser levadas em consideração quando do desenvolvimento dos projetos. Com conhecimentos adquiridos relativos à acessibilidade, o projetista passa a ter maior experiência com as linguagens hipertextuais, tornando-se, assim, um profissional mais ajustado às demandas da Sociedade da Informação.

- Cumprimento de medidas legais: A Lei 10.048/2000 dá prioridade de atendimento às pessoas que especifica (BRASIL, 2000[1]), no caso às pessoas com necessidades especiais. Já a Lei 10.098/2000, estabelece normas gerais e critérios básicos para a promoção da acessibilidade às pessoas com deficiência ou mobilidade reduzida (BRASIL, 2000[2]). Também, o Decreto 5.296/2004, que regulamenta as leis anteriores, versa, pela primeira vez no Brasil, especificamente sobre acessibilidade na Internet. Em seu capítulo VI, no artigo, 47º torna obrigatória a acessibilidade dos portais e sítios da administração eletrônica para usuários deficientes visuais, estipulando um prazo de doze meses. O mesmo artigo prorroga esse prazo por mais um ano, no caso de portais e sites muito complexos. Assim, o prazo, já prorrogado, expirou em dezembro de 2006.

3.1) Diretrizes para o desenvolvimento de páginas acessíveis

O W3C publicou, em maio de 1999, as Diretrizes para Acessibilidade do Conteúdo Web 1.0 (Web Content Accessibility Guidelines - WCAG 1.0), sendo, até hoje, a principal referência em termos de acessibilidade à web no mundo. De acordo com UTAD/GUIA (1999), o documento pretende explicar como tornar o conteúdo web acessível a pessoas com deficiências. As diretrizes são: Diretriz 1 - Fornecer alternativas equivalentes ao conteúdo sonoro e visual; Diretriz 2 - Não recorrer apenas à cor; Diretriz 3 - Utilizar corretamente anotações e folhas de estilo; Diretriz 4 - Indicar claramente qual o idioma utilizado; Diretriz 5 - Criar tabelas passíveis de transformação harmoniosa; Diretriz 6 - Assegurar que as páginas dotadas de novas tecnologias sejam transformadas harmoniosamente; Diretriz 7 - Assegurar o controle do

usuário sobre as alterações temporais do conteúdo; Diretriz 8 - Assegurar a acessibilidade direta de interfaces de usuário integradas; Diretriz 9 - Pautar a concepção pela independência em face de dispositivos; Diretriz 10 - Utilizar soluções de transição; Diretriz 11 - Utilizar as tecnologias e as diretrizes do W3C; Diretriz 12 - Fornecer contexto e orientações; Diretriz 13 - Fornecer mecanismos de navegação claros; Diretriz 14 - Assegurar a clareza e a simplicidade dos documentos.

Em maio de 2007, foi lançado, no site da W3C, um esboço da WCAG 2.0 - (W3C, 2007), segunda versão das Diretrizes de Acessibilidade. Essa versão está baseada em quatro princípios: 1)Princípio da percepção: o conteúdo deve ser perceptível ao usuário; 2) Princípio da operação: os elementos de interface do usuário devem ser operáveis; 3) Princípio da compreensão: o conteúdo e controles devem ser compreensíveis ao usuário; 4) Princípio da robustez – o conteúdo deve ser robusto suficiente para trabalhar com tecnologias atuais e futuras: maximizar a compatibilidade com agentes de usuários atuais e futuros, incluindo tecnologias assistivas.

Como podemos perceber, tais diretrizes/princípios são um tanto subjetivos, o que dificulta seu entendimento. Alguns autores como Soares (2007), Gomes (2007), dentre outros, questionam sua eficácia. Gomes (2007) refere que as diretrizes da WCAG 2.0 ainda estão em fase de revisão e que as regras e recomendações disponibilizadas não são fáceis de compreender porque estão escritas em uma forma demasiado genérica. Segundo o autor, a versão 2.0 das diretrizes buscou torná-las tecnicamente neutras para que fossem aplicadas a diversos tipos de elementos, inclusive àqueles que possam aparecer no futuro; só que isso dificulta bastante a própria percepção das recomendações.

Por essas razões muitos autores desistiram da WCAG 2.0 e formaram o grupo WCAG Samurai. A idéia do WCAG Samurai foi de criar uma Errata para o WCAG 1.0, de modo que seja possível utilizar essa versão do documento (1.0), mas adaptada à tecnologia atual (GOMES, 2007). Em junho de 2007 foi lançada a primeira versão da Errata, apesar de não ser a versão final (WCAG Samurai, 2007). De acordo com Gomes (2007), as principais alterações efetuadas no WCAG 1.0 foram: eliminação de termos como evite usar e substituição por uma linguagem mais incisiva como: não use ou é obrigatório ter; eliminação das regras de Prioridade 3, por serem praticamente inexequíveis; passa a ser obrigatório o respeito às recomendações das Prioridades 1 e 2. Isso significa que é obrigatório ter código válido em todos os casos; não foram adicionadas novas regras para deficiências cognitivas. Tanto o WCAG 1.0 como o WCAG 2.0 possuem falhas atinentes a esse ponto e o WCAG Samurai não certifica que, mesmo seguindo todas as regras, o website seja acessível para pessoas com este tipo de deficiência, como é o caso da dislexia; o uso de tabelas e frames para layout é completamente banido, no entanto podem ser utilizados ainda os iframes ; fim do noscript . Todos os scripts e applets mais conhecidos como AJAX e Flash , na maioria dos casos, devem ser diretamente acessíveis ao invés de utilizar a técnica do noscript; tudo o que estiver disponível em formato PDF deve também estar disponível em HTML; todos os vídeos com som devem ter legendas ou audio descrição (dependendo dos conteúdos).

Em nível de Brasil, na Cartilha Técnica do Manual de Acessibilidade do Governo Eletrônico (eMAG, 2005), constam oito diretrizes técnicas de acessibilidade, baseadas na WCAG 1.0, mas adaptadas à nossa realidade: Diretriz 1: fornecer alternativas equivalentes para conteúdo gráfico e sonoro; Diretriz 2: assegurar-se de que o site seja legível e compreensível mesmo sem o uso de

formatações; Diretriz 3: dar preferência às tecnologias de marcação e formatação; Diretriz 4: assegurar que toda a informação seja interpretada corretamente, com clareza e simplicidade; Diretriz 5: assegurar que as tecnologias utilizadas funcionem – de maneira acessível – independente de programas, versões e futuras mudanças; Diretriz 6: assegurar sempre o controle do usuário sobre a navegação do site; Diretriz 7: identificar claramente quais os mecanismos de navegação; Diretriz 8: em casos não contemplados pelas diretrizes anteriores, utilizar sempre recursos reconhecidos por instituições com propriedade no assunto, como tecnologias acessíveis.

3.2) Validações de Ambientes Virtuais

De acordo com eMAG (2005), as diretrizes de acessibilidade, por si só, não garantem a acessibilidade, tratam-se apenas de pontos orientadores para que os requisitos de acessibilidade sejam cumpridos. Assim, após atentar para os quesitos de acessibilidade, o desenvolvedor de páginas web deverá realizar a validação das mesmas. Ela é obtida por meio de testes, utilizando mecanismos automáticos e manuais e deve estar presente desde as fases iniciais de seu desenvolvimento.

Validação Automática: o desenvolvedor da página pode verificar se esta cumpre com as diretrizes de acessibilidade por meio de um validador on line, que é um serviço em linha, um software que detecta o código HTML de uma página web e analisa seu conteúdo, normalmente baseado na iniciativa de acessibilidade do W3C (SOARES, 2005[1]). O validador ajuda a comprovar se a interface foi desenvolvida utilizando os padrões web de acessibilidade. Em caso negativo, aponta onde está o problema. Os métodos automáticos são geralmente rápidos, mas não são capazes de identificar todos os aspectos da acessibilidade. Esses programas verificadores estão disponíveis na Internet. São alguns exemplos de verificadores automáticos: WebXACT (antigo BOBBY) - (inglês); Cyntia - (inglês); Lift - (inglês); W3C - (inglês); Valet - (inglês); Ocawa - (inglês); TAW - (espanhol); Da SILVA - (português); eXaminator - (português); Hera - (português).

Caso a página esteja acessível, o programa avaliador concederá um selo de acessibilidade denotando o nível de conformidade alcançado. De acordo com Soares (2005[1]) e (2005[2]), apesar de úteis, os validadores automáticos não são perfeitos e muito menos inteligentes. Uma validação automática pode avaliar apenas algumas das regras, e não todas. Os selos de acessibilidade fornecidos por esses programas não são garantia de acessibilidade; e da mesma forma, um site que não possui selo pode ser acessível. O autor continua referindo que, apesar da utilidade desses softwares, eles não podem substituir uma boa avaliação manual.

Validação Manual: outra etapa de avaliação de acessibilidade de um site, recomendada pelo W3C (W3C, 2005) é a avaliação manual. Esta é necessária, pois nem todos os problemas de acessibilidade de um site são detectados mecanicamente por meio dos verificadores automáticos. A existência de um bom contraste entre o fundo e o primeiro plano, por exemplo, só pode ser verificada por um ser humano (EVALDT, 2005). Além disso, conforme destaca Dias (2003), a avaliação humana pode ajudar a garantir a clareza da linguagem e a facilidade de navegação.

Além de permitirmos o acesso aos usuários com alguma limitação, torna-se importante também garantir uma boa navegabilidade e clareza das informações veiculadas; por isso trazemos dois novos conceitos: usabilidade e comunicabilidade aplicadas à acessibilidade.

3.3) Usabilidade aplicada na Acessibilidade

Um conceito que começa a ser utilizado na atualidade é o da Usabilidade aplicada à Acessibilidade. Tal prática amplia o entendimento de acessibilidade virtual ao mencionar a importância não apenas de se aplicar as recomendações do W3C, mas também de se tornar os ambientes fáceis de usar para todos, ou seja: "aplicar usabilidade nos sites para torná-los verdadeiramente acessíveis" (SPELTA in SOARES, 2005[2]).

Ao trazer o termo Usabilidade na Acessibilidade, Amstel (2006) refere:

o princípio básico da web é acesso por qualquer tipo de pessoa, em qualquer lugar, mas são poucos os websites que seguem esse princípio. Ora por incompetência técnica, ora por desinteresse comercial, a maioria dos criadores de websites ignora boas práticas que viabilizam o acesso à informação (acessibilidade) e seu uso (usabilidade) por pessoas com necessidades especiais (AMSTEL, 2006).

O mesmo autor também destaca que "acessibilidade e usabilidade são condições básicas para a inclusão social digital" (AMSTEL, 2006).

Soares (2005[2]) endossa o exposto acima ao mencionar:

Não basta ter uma página web acessível, é importante que ela também seja fácil de usar e entender. A diferença entre teoria e prática é grande quando o assunto é desenvolvimento de sites acessíveis. De um lado do rio encontra-se uma página web com todas as regras de acessibilidade aplicadas exatamente como nas cartilhas, guias e recomendações do W3C, e do outro lado, uma página verdadeiramente acessível (ibidem).

Queiroz (2006[1]) complementa referindo que não basta incluirmos na codificação de uma página etiquetas ou atributos do modo a torná-la acessível; é preciso imergir na lógica da navegação dessa página via teclado, para que sua utilização fique fácil e confortável. Dessa forma, segundo ele, o conceito de acessibilidade une-se ao de usabilidade. O autor destaca que ao confeccionarmos páginas amigáveis, via teclado, e permitirmos o uso de teclas de atalho, obteremos uma boa usabilidade e atingiremos um ótimo percentual de acessibilidade, não apenas para pessoas cegas, como para aquelas com alguns tipos de limitações físicas, além de propiciar uma navegação mais rápida, fácil e eficiente a todos. Segundo esse autor é preciso ter sempre em mente que existem usuários que navegam apenas por meio do teclado, como é o caso de pessoas com limitação motora ou visual. Quando isso ocorre, o deslocamento do foco nos links e objetos da página, por padrão, se realizam de cima para baixo e da esquerda para a direita, e os comandos são lidos sequencialmente pelo navegador e softwares de leitura.

3.4) Comunicabilidade aplicada na Acessibilidade

Uma funcionalidade imprescindível para que um ambiente respeite os padrões de acessibilidade refere-se à utilização de equivalentes textuais para todo o conteúdo não textual. Assim, imagens de figuras, fotografias, botões, animações, linhas horizontais separadoras, mapas, filmes, sons... devem ser acompanhados de uma descrição textual; só que essa descrição deve ser equivalente, ou seja, deve transmitir "as mesmas informações que os elementos disponibilizados" (QUEIROZ, 2006[2]), pois será por meio dela que o usuário que não enxerga terá o entendimento de seu conteúdo. O equivalente textual tem a função de traduzir em texto, em linguagem clara e simples, a imagem ou som, especialmente se os mesmos possuírem uma funcionalidade. Quando procedemos dessa forma, estamos realmente comunicando ao usuário, com limitação visual, o conteúdo daquela imagem ou ao usuário com limitação auditiva, o conteúdo daquele som. A intenção, quando se refere que o conteúdo não textual seja disponibilizado também em forma textual, no caso de usuários com limitações visuais, "se deve à necessidade que um leitor de telas tem para transmitir as informações, uma vez que não consegue ler nada além de textos" (QUEIROZ, 2006[2]). Em caso de imagens decorativas, a equivalência textual deve existir nula. Isso evita que uma pessoa cega tenha que ouvir informações desnecessárias, causando o problema conhecido como verborragia (QUEIROZ, 2007).

Quando uma pequena descrição não é suficiente para a compreensão de todo o conteúdo constante na imagem, é preciso utilizar outro recurso. Queiroz (2006[2]) traz um exemplo de uma imagem que apresenta a população de cada capital brasileira – um mapa de imagem. Nesse caso, a imagem deverá ter um equivalente textual (descrição), com um pequeno texto do tipo: População das capitais brasileiras. Como complemento, é preciso agregar uma página em HTML com todas as capitais e suas respectivas populações, que poderá ser acessada por meio da própria imagem ou por técnicas não perceptíveis aos usuários que estejam navegando com o mouse, como um link com uma imagem transparente, por exemplo. Dessa forma, o mapa de imagem pode ser visualizado normalmente por usuários que enxergam, sem agregar informações desnecessárias aos mesmos e também estará acessível aos usuários que utilizam leitores de tela.

Assim, quando tratamos do processo de comunicação desenvolvedor X usuário final, para que haja clareza no conteúdo veiculado, precisamos ter bem presentes o conteúdo que desejamos comunicar e, no caso de usuários cegos, o que será sonorizado pelos leitores de tela. Queiroz (2006[2]) destaca também que se o logotipo de uma empresa tiver apenas a função de anunciá-la, sua descrição deve ser apenas algo como Logotipo da <nome da empresa>, sem a necessidade da descrição visual do logotipo. E ainda, se esse logotipo for também um link que remete, por exemplo, para a página principal, nas páginas internas em que o mesmo aparece, ele deve estar descrito como: Voltar para a Página Principal ou outra descrição que traduza sua real função.

Ainda relativo à utilização de linguagem clara e simples para as descrições dos links, Queiroz (2006[2]) refere que pessoas cegas, normalmente, utilizam duas formas de navegação (leitura no interior dos sites): a leitura corrida de todo o texto que se encontra na página ou a leitura sintética, que é a que percorre apenas os links e campos de formulário. Essa última é utilizada quando os usuários desejam obter um resumo do conteúdo total do site. Esse procedimento é realizado, a partir do início da página, utilizando a tecla Tab. A página é percorrida link a link ou por campos de formulário, pulando-se os textos, imagens e tudo o que não for link ou campo de formulário. Assim o deficiente visual vai escutando,

por meio do leitor de telas, ou tateando, por meio do monitor Braille, os textos contidos nos links. O que ocorre é que são muito utilizadas para nomear links expressões do tipo: Saiba Mais, Clique Aqui, Leia Mais... Quando um deficiente visual encontra uma expressão desse tipo no link, não pode continuar sua navegação por links, "pois tal texto não é completo e suficiente para ele ter conhecimento sobre o que ele deve saber mais, ou mesmo por que ele deve clicar naquele link" (QUEIROZ, 2006[2]). A pessoa com limitação visual deve interromper a leitura rápida (por links), posicionar seu leitor de telas algumas linhas antes e proceder a uma nova leitura, só que detalhada. Assim, uma linguagem clara significa, nesses casos, "o texto do link ter uma continuidade", que explicita o texto anterior (ibidem), como, por exemplo: Leia Mais Notícias.

Funcionalidades que agregam objetos programáveis, como scripts e applets, são outros tipos de elementos não textuais. São escritos em linguagens diferentes ao HTML, objetivando criar na interface um comportamento dinâmico ou interativo, como Java ou Flash. Esses elementos possuem uma dificuldade para serem disponibilizados em um formato acessível (QUEIROZ, 2006[2]). Diante disso, se não for possível evitá-los, é preciso que haja uma descrição equivalente também nesses casos.

Além da clareza na descrição equivalente de elementos não textuais e links, é preciso assegurar que a interface, como um todo, apresente uma linguagem simples e clara a todos os perfis de usuário, permitindo assim o rápido entendimento do conteúdo da página. Para que isso ocorra, Queiroz (2006[2]) sugere: que seja realizada uma criteriosa revisão do texto; que sejam utilizados títulos pertinentes, que se divida o texto em parágrafos afins, utilizando cabeçalhos que definam o conteúdo a seguir; que se forem utilizadas palavras desconhecidas, específicas de determinada matéria, seja criado um glossário de fácil acesso, para que a linguagem do texto seja compreendida pelo maior número de pessoas possível; que abreviaturas sejam evitadas ou que sejam utilizadas marcações que façam o leitor de telas ler por extenso tais abreviaturas; que seja utilizado um corretor ortográfico e que seja verificada a pontuação, pois os leitores de tela reproduzem exatamente o conteúdo do texto escrito. O autor também refere que a importância da pontuação toma dimensões ainda maiores quando são utilizados sintetizadores de voz, pois os mesmos identificam a pontuação por meio de pausas, silêncios na voz, por vezes quase imperceptíveis. Assim, um ponto tem um tempo de silêncio, a vírgula tem um tempo menor que o ponto e tempos mais fracionados ainda são usados para o ponto e vírgula e a vírgula. E "a exclamação e a interrogação têm sonoridades semelhantes ao que representam, tanto quanto a reticências" (QUEIROZ, 2006[2]).

4) PONTOS IMPRESCINDÍVEIS PARA AMBIENTES COM QUALIDADE DE USO

Tomando como base o referencial teórico atinente à acessibilidade à web, as interações até hoje realizadas com usuários deficientes visuais (SONZA, 2007), (SONZA, 2008) e o trabalho do Núcleo do SIEP no CEFET BG, passamos a mencionar os itens que consideramos imprescindíveis para que uma interface atenda à acessibilidade, usabilidade, comunicabilidade.

Após a interface ser implementada de acordo com os padrões de desenvolvimento web, utilizando cada comando com seu real propósito e separando layout de conteúdo, é fundamental atentar para:

Acessibilidade:

Etiquetagem: para que a página possa ser lida pelos leitores de tela, é preciso fornecer alternativas ao conteúdo visual. Diante da multiplicidade e constante expansão de recursos e possibilidades que o mundo web hoje nos oferece, explicitaríamos e complementaríamos essa necessidade da seguinte forma: utilizar uma descrição clara e significativa, condizente com o conteúdo que agrega, para imagens, mapas de imagens, links, botões, caixas de listagem, frames e qualquer elemento não textual da interface Quando falamos de etiquetagem não podemos esquecer das animações em Flash - recurso amplamente utilizado atualmente, seja em sites, portais ou ambientes de aprendizagem. Quando da existência desses eventos é preciso inserir uma descrição inclusive nos botões e controles internos, objetivando sua devida leitura com os agentes de usuário Caso haja a necessidade de disponibilização de arquivos, como aqueles em PDF, é preciso inserir outros formatos, como TXT e/ou DOC com todo o conteúdo não textual devidamente descrito/adaptado. Isso permite o acesso com navegadores textuais, além do entendimento completo de todos os elementos que compõem o arquivo.

Uso adequado das folhas de estilo: por uso adequado de folhas de estilo referenciamos: separar completamente apresentação (estilo visual) e conteúdo de uma interface evitando assim a chamada Poluição Sonora (leitura de itens desnecessários ao usuário de leitor de telas), tornando-a mais leve e permitindo sua interação também com agentes de usuário cuja leitura possível é apenas aquela propiciada por interfaces programadas em (X) HTML. Como destaca Silva (2007) além de a interface não apresentar erros tanto no arquivo HTML como no(s) CSS, é preciso que todos os elementos de estilização sejam programados nos arquivos de folhas de estilo, deixando para o arquivo HTML a tarefa exclusiva de marcar e estruturar o conteúdo do documento.

Navegação por teclado: a interface deve prever a navegação independente de dispositivos. No caso dos deficientes visuais, o uso do teclado é imprescindível, por isso é necessário permitir a navegação via teclado em todos os elementos da página, inclusive nas caixas combinadas, caixas de contexto, caixas de listagem e aqueles programados em JavaScript e Flash.

Usabilidade:

Cores, Redimensionamento e Contraste: além de não recorrer apenas à cor para veicular informações e utilizar um bom contraste entre fundo e primeiro plano, é preciso oferecer na interface opções de alteração de contraste e de redimensionamento dos elementos que a compõem, visto que existem usuários com baixa visão e outros com cromodeficiências que poderão necessitar de outras combinações de cores e/ou sentirão maior conforto com os elementos da interface ampliados.

Atalhos: fornecer atalhos por teclado do tipo: Ir para Menu, Ir para Conteúdo, Ir para a Página Principal, Voltar para a Página Anterior, além de âncoras para locais específicos da interface.

Contexto, orientação e auxílio para a navegação: fornecer contexto e orientações, inclusive um feedback, ou seja, localização do usuário na interface. Além de dividir a interface por blocos mais fáceis de gerir, é preciso também propiciar a orientação na interface por esses blocos ou partes onde cada um esteja devidamente identificado, além da indicação de início e fim de cada bloco. Para o usuário

de leitor de telas, a leitura é realizada de forma seqüencial, sob a forma de links, textos, caixas, botões. Assim, muitas vezes, eles não diferenciam as informações/ferramentas contidas nos menus daquelas que são apenas links. Para o usuário normo-visual, o menu fica claramente identificável devido ao destaque que é dado ao mesmo e ao seu posicionamento, geralmente no lado esquerdo e/ou na parte superior da tela. A inserção dessa informação agiliza e facilita a navegação, sendo um quesito importante para a usabilidade da interface. Também é fundamental, além de fornecer informações sobre a organização geral de um ambiente, como aquelas encontradas nos Mapas de Site, inserir Dicas de Navegação na interface, com os principais comandos para navegação na mesma, inclusive em conjunto com Tecnologias Assistivas.

Comunicabilidade:

Qualidade da etiquetagem de todos os elementos não textuais: para que o ambiente realmente comunique o que deseja, é preciso que haja, não só a etiquetagem dos elementos não textuais, pura e simplesmente; mas uma etiquetagem de qualidade, que realmente transmita a informação aos usuários. Assim é necessário que seja significativa - que realmente descreva, de forma clara, precisa, objetiva e sem erros ortográficos o conteúdo que agrega.

Qualidade e clareza de todo o conteúdo: Assegurar a clareza e simplicidade em toda a interface garantirá uma comunicação eficaz entre usuário e desenvolvedor. Como sinônimo de clareza e simplicidade destacamos: uso de uma linguagem simples e objetiva em toda a interface, inclusive no conteúdo textual, tomando o cuidado de prover uma escrita sem erros ortográficos e com pontuação correta. Sendo assim, para um correto entendimento do conteúdo veiculado pontuação e ortografia corretas são fatores relevantes. É preciso também especificar, por extenso, cada abreviatura quando de sua primeira ocorrência visto que os usuários que acessam a interface poderão não saber o significado de tais abreviaturas.

Destino dos links: identificar claramente o destino de cada link, ou seja, que ele realmente descreva o item ao qual remete, pois é por meio dessa descrição que o usuário de leitor de telas decidirá pelo seu acesso ou não.

CONSIDERAÇÕES FINAIS

Atualmente alguns auxílios podem ser utilizados para validar a acessibilidade de uma interface. Um exemplo disso são os validadores automáticos. Esses robôs fornecem o selo de acessibilidade para os ambientes que respeitam as diretrizes, seja do W3C ou do e-Gov. Apesar de terem seu mérito, esses programas normalmente validam apenas a primeira página da interface, sendo que, se desejarmos validar as demais, teremos que realizar a validação página por página. Outra fragilidade do validador refere-se à descrição dos elementos não textuais. Os validadores aceitam qualquer descrição, até mesmo caracteres em branco, verificando apenas se há uma descrição e não sua qualidade. E essa fragilidade não se resume a etiquetagem dos elementos não textuais, mas a toda a interface. Por serem automáticos, os validadores não realizam uma validação semântica. Por mais modernos que sejam, nunca irão substituir uma validação manual.

Quando tratamos de web semântica, do uso do comando certo no lugar certo, de separação completa entre layout e conteúdo, de utilização do conceito de tableless, de descrição clara e objetiva de links e de elementos não textuais, de seqüência lógica de disposição dos elementos em uma interface – todos esses princípios se encontram na WCAG e nos padrões de desenvolvimento web e são essas diretrizes que buscam ser verificadas pelos validadores automáticos, que comparam o código com cada uma das 14 diretrizes (WCAG 1.0 – UTAD/GUIA, 1999) e seus respectivos subitens. O que acontece é que os mesmos não verificam a semântica do código, não verificam a lógica de programação embutida nas interfaces, não verificam a qualidade de descrição de links e elementos não textuais e, por isso, um rótulo – selo de acessibilidade – ou mesmo selo da validação do código HTML ou CSS, apesar de importante, não garante uma web semântica e acessível. Nossos estudos reafirmaram a convicção de que diversos aspectos da acessibilidade, usabilidade e comunicabilidade só poderão ser validados por usuários reais, ratificando a importância da validação manual – ação fortemente executada no núcleo do SIEP do CEFET BG.

Utilização de códigos HTML e CSS válidos, com cada comando sendo utilizado para seu real propósito e separação completa entre layout e conteúdo são a base para interfaces com qualidade de uso. Sobre esses pilares sólidos, é preciso atentar para todos os quesitos de acessibilidade, usabilidade e comunicabilidade já mencionados no aporte quatro desse artigo.

Cabe destacar, entretanto, que, além de envidar esforços no sentido de apresentar um ambiente que vá de encontro aos preceitos de qualidade de uso de sistemas, sem cercear o acesso, navegação e comunicação a nenhum perfil de usuário, é preciso garantir a qualidade de sua interface. Cientes de que a intervenção e sensibilidade humanas são imprescindíveis em todas as etapas da implementação e manutenção do mesmo, torna-se necessário que a pessoa responsável pela manutenção/atualização da interface tenha bem presentes essas considerações, para não incorrermos no erro de concebermos uma interface com essas qualidades e, na ocorrência das primeiras atualizações, já deixe de lado alguns aspectos.

Apesar desse movimento de info-inclusão, temos a convicção de que estamos apenas iniciando uma longa caminhada; caminhada esta, felizmente, sem volta. Esperamos que, para um futuro bastante próximo, informatas, projetistas web, educadores e os próprios alunos com e sem necessidades especiais, imbuídos em um espírito mais solidário, mais justo e ético trabalhem juntos, em prol de um acesso igualitário e autônomo a todos. Estamos certos de que se tivermos a oportunidade de utilizar ambientes digitais que realmente sejam acessíveis à pluralidade de usuários, daremos passos decisivos na senda da tão sonhada inclusão virtual. E esse trabalho, que se constituiu um grande e necessário desafio, não pára por aqui...

CAPÍTULO 2

WEB SEMÂNTICA

"Semântica é a parte da gramática que estuda o sentido e a aplicação das palavras em um contexto."

... Explicação descomplicada sobre web semântica

... Colocar origem

Muitas vezes, desenvolvedores que estão acostumados a desenvolver sites utilizando tabelas estarão inclinados a criar marcação com várias tags , claramente espelhando-se em layouts estruturados com tabelas. Esta prática é chamada de *divite*, mas deve ser evitada.

Similarmente, devemos ser cautelosos sobre a divisão de elementos únicos com tags . Isto é frequentemente realizado para propósitos de estilização, mas usualmente isto destrói com a marcação semântica, além de ser totalmente desnecessário.

A tag é usada para criar divisões na página, mas ao mesmo tempo é utilizada para criar grupos de conteúdo. Por exemplo, por que fazer isto:

```
<div id="menu">
    <ul>
        <li>First item</li>
        <li>Second item</li>
        <li>Third item</li>
    </ul>
</div>
```

Quando podemos estruturar a página usando menos código:

```
<ul id="menu">
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
</ul>
```

A este uso desnecessário da tag damos o nome de “divite”. O mesmo problema pode ocorrer comumente com desenvolvedores web ao utilizar o atributo class, ao que chamaremos de classite.

O atributo class existe para que possamos utilizar a mesma esitilização para vários elementos em comum, mas como a divite, muitos desenvolvedores utilizam o atributo class no lugar da tag apropriada, como no exemplo:

```
<p class="address">
        John Smith<br />
        1234 Rolling Rock Rd. <br />
        Albany, NY, 12345<br />
</p>
```

O correto é usar o elemento (X)HTML address, veja abaixo:

```
<address>
        John Smith<br />
        1234 Rolling Rock Rd. <br />
        Albany, NY, 12345<br />
</address>
```

Desenvolvedore tem também a tendência de usar o atributo class em diversos elementos que se repetem ao invés de simplificar aplicando a class para o elemento pai, veja o seguinte código:

```
<ul>
        <li class="cheese-type">Cheddar</li>
        <li class="cheese-type">Mozzarella</li>
        <li class="cheese-type">Parmesan</li>
        <li class="cheese-type">Swiss</li>
</ul>
```

Aqui uma maneira muito mais simplificada:

```
<ul class="cheese-types">
    <li>Cheddar</li>
    <li>Mozzarella</li>
    <li>Parmesan</li>
    <li>Swiss</li>
</ul>
```

Mas, quando temos o cuidado no uso das tag , <p>, e o atributo class, iremos utilizar elementos (X) HTML mais apropriados, eliminando a confusão do documento e possibilitando um código mais semântico, fácil de dar manutenção, leve e rápido de ser carregado pelo browser e enfim acessível.

O grande paradigma do desenvolvimento de sites e sistemas acessíveis está no uso correto de cada tag para o seu respectivo conteúdo, como por exemplo, ao pegarmos uma citação e seu autor, como segue abaixo, de que forma devemos implementar a demarcação?

> “Somente aquele que administra suas idéias de forma livre é dono de suas idéias, e somente aquele que é dono de suas idéias não é escravizado por elas.”
>
> - Lin Yutang

Que tal desta forma:

```
<p>“Somente aquele que administra suas idéias de forma livre é dono de suas idéias, e somente aquele que é dono de suas idéias não é escravizado por elas.”</p>
<p>- Lin Yutang</p>
```

Para muitos, isto será muito familiar, mais está forma não é correta, sendo que temos tags apropriadas como elementos semânticos:

```
<blockquote>“Somente aquele que administra suas idéias de forma livre é dono de suas idéias, e somente aquele que é dono de suas idéias não é escravizado por elas.”</blockquote>
<cite>- Lin Yutang</cite>
```

AS TRÊS CAMADAS DE UM DOCUMENTO WEB MODERNO

Modernos e bem-estruturadios documentos web tem três camadas distintas de dados (Figura 1).

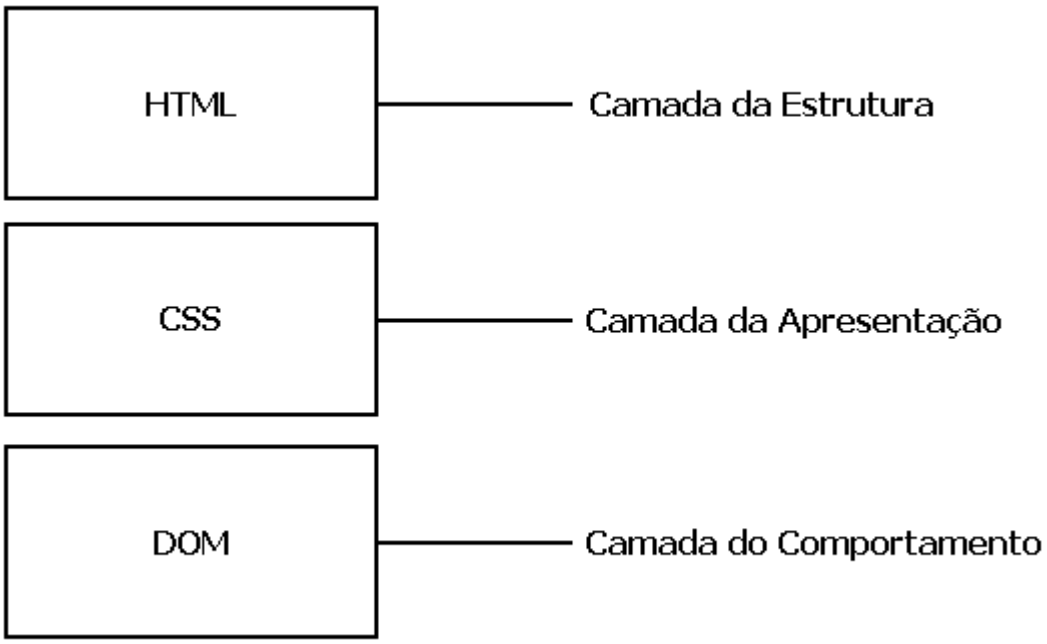

Figura 1: Camadas de um documento Web Moderno

A primeira camada da estrutura, que é o documento de texto marcado em HTML ou XHTML. Este contém o conteúdo de seu documento, juntamente com a informação semântica indicando o que cada bit texto é (títulos [h1, h2, h3, h4, h5, h6], parágrafos [p], listas [ul, ol, dl] e etc)

A segunda camada é a camada de apresentação, o qual irá determinar como seu documento será apresentado para o usuário, determinando os elementos de layout e tipografia, cores, decoração de imagens, e apresentaçao para a família de leitores de tela mais usados. Geralmente, a camada de apresentação de um documento web é escrito usando CSS.

Além destas duas camanda, temos também como referência para a layer de comportamento do documento.

Não iremos discutir esta camada em profundidade, mas deveríamos entender ao que se refere usando scripts (usualmente JavaScript para manipulação do Modelo de Objeto de Documento, ou DOM) para atualizar, adicionar, ou remover itens de um documento baseado em comportamentos do usuários

Para simplificar o uso destas três camadas em conjunto, considete uma página de contato básica do sitie de uma empreas. O formulário é produzido em (X)HTML. Este texto é então estilizado numa apresentação estética para tela usando CSS. Depois de preencher o formulário e clicar no

botão de enviar, o JavaScript irá validar os campos e verificar os campos obrigatórios e se estiver preenchido corretamente os dados do formulário são enviados, caso contrário surgirá uma mensagem de aviso.

W3C

O W3C é um consórcio internacional totalmente dedicado à produção de padrões para desenvolvimento na web. Todos estes padrões são construiídos com participação das instituições membros, mais uma equipe de tempo integral e especialistas convidados.

Desde 1994, já publicou mais de 90 padrões, chamados “W3C Recommendations” (http://www.w3.org/TR).

Tem por objetivo desenvolver e disseminar a cultura de adoção de padrões como vetor de desenvolvimento pleno da web a longo prazo, garantindo competitividade, interoperalidade e acessibilidade, a expansão e a durabilidade das aplicações em longo prazo, pois as ferramentas também evoluem com base nesses padrões, browsers e leitores de tela.

Para cumprir o seu objetivo, a W3C trabalha constantemente com empresas e desenvolvedores para difundir principalmente a cultura e as vantagens de adotar os padrões web. Jeffrey Zeldman (http://www.zeldman.com), um dos mais famosos desenvolvdores web do mundo, afirma que desenvolver sem considerar padrões é não pensar na evolução futura da tecnologia, que acontece toda sobre padrões internacionais.

A acessibilidade é uma das principais diretrizes da W3C, onde no Brasil é de certa forma apoiada pelo decreto-lei 5296/2004, do Governo Federal, que estabelece normas gerais e critérios básicos para a promoção da acessibilidade, utilizando-se, assim, de padrões da W3C.

Porém, o Goveno não participa ainda da elaboração desses padrões, mas o fato de recoomendar padrões W3C é relevante como reconhecimento da importância das atividades do consórcio mundialmente.

Por outro lado, as recomendações podem ser seguidas por quaiquer pessoas ou instituições, independentemente da orientação governamental. Essa é uma das grandes vantagens das nossas ações: a evolução dos padrões não esta condicionada a adesão do Governo. Mas o peso institucional deles faz uma diferença significativa na consolidação do uso de padrões.

Escrevendo código semântico

Você pode encontrar uma documentação bastante abrangente das tags do HTML, em especial aquelas usadas pelo XHTML, no endereço:

http://www.w3schools.com/xhtml/xhtml_reference.asp

Abaixo as principais tags utilizadas no deseenvolvimento web.

Blocos de texto

div

"Divisões", seções em seu conteúdo.

p

Parágrafos

span

Seções em seu conteúdo. A diferença para o div é que o span é usado para marcar trechos de seu texto. Por exemplo: aqui vai uma palavra especial.

Listas

ul, ol e li

Listas, numeradas ou não.

dl, dd, dt

Listas de definição (para escrever um glossário, por exemplo)

Significado especial

h1, h2, h3, h4, h5 e h6

Títulos. Quanto menor o número, maior a importância.

abbr

Abreviatura

acronym

Acrônimo

address

Informações sobre o autor ou dono da página, como email de contato e endereço físico.

blockquote

Uma citação longa (um bloco)

q

Uma citação curta (para usar na mesma linha)

cite

Autor da citação

em

Texto enfatizado. A renderização padrão é itálico.

strong

Texto "forte". A renderização padrão é negrito.

del

Texto excluído. (Não entendeu? Pense no sistema de controle de reviões do Word, se você já usou...)

ins

Texto inserido.

Código de computador e etc.

pre

Texto pré-formatado

code

Código de computador

kbd

"Entrada de teclado"

CAPÍTULO 3

XHTML

O W3C criou o XHTML. XHTML é uma forma de escrever HTML de modo que ele também seja um documento XML válido. Com o objetivo de adotar maior organização e legibilidade do código HTML, utilizando-se das regras de escrita do XML no HTML: Toda tag que for aberta deve ser fechada; Todas as tags devem ser escritos em minúsculo; Todos os atributos devem conter um valor, entre outros. Veja:

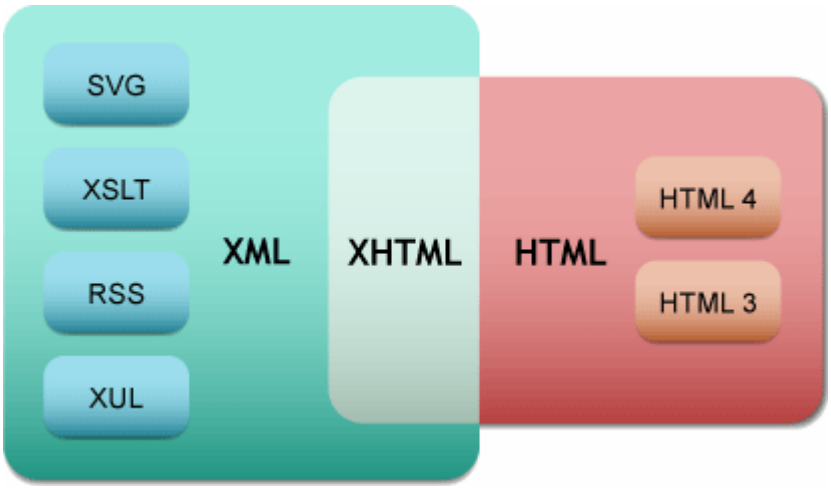

Figura 2: XHTML e sua relação com XML e HTML

Você vai notar no mapa que XML é uma linguagem usada para definir outras linguagens. Simplificando podemos dizer que XML é uma linguagem que serve para criar outras linguagens. Linguagens como XSLT, SVG, XUL e RSS são todas XML. XHTML também é XML e é ao mesmo tempo HTML. Assim, um browser lê XHTML como um arquivo HTML normal e um interpretador de XML pode lê-lo como XML.

XHTML é um subconjunto de XML usado para escrever HTML. Na prática trata-se de escrever o HTML com uma sintaxe ligeiramente diferente, de modo que ele também seja um arquivo XML válido.

Escrevendo XHTML válido

DOCTYPE

O Doctype (Document Type Definition, vulgo DTD) é a primeira coisa que se deve escrever em um arquivo XHTML. Ele vai na PRIMEIRA LINHA do seu documento, e serve para informar ao browser que tipo de documento será visualizado. Existem quatro tipos:

Strict: Este tipo é usado quando você fez um código 100% XHTML, sem erros. Deve ser escrito assim:

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

XHTML 1.1: Assim como o Strict utilizamos o XHTML 1.1, no desenvolvimento e implantação de todos os padrões web, que discutimos até este momento. Deve ser escrito desta forma

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.1t//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

Transitional: Este é o modo mais usado, durante a transição entre o HTML e o XHTML Sua sintaxe é:

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

Frameset: É usado quando você está utilizando FRAMES em seu site (Atualmente os Frames são depreciados, pois prejudicam a acessibilidade). Escreve-se assim:

<!DOCTYPE html

PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

Feche TODAS as tags

Para se fazer um XHTML válido, devemos fechar TODAS as tags:

Não devemos esquecer de fechar as tags que estamos carecas de conhecer: , , etc...

E também não podemos esquecer de fechar as tags "solitárias". Assim, ao invés de
, escrevemos:

Use letras minúsculas

Quem nunca viu um código fonte de um documento HTML escrito assim:

<A HREF="http://tags.com.letras.minúsculas/" TARGET="_BLANK"> </A>Um documento XHTML deve ter TODAS as tags e seus respectivos atributos escritos com letra minúscula:

Não esqueça das "ASPAS"

Esta regra é bem simples. Nas tags XHTML, todos os valores dos atributos devem estar entre "ASPAS".

Atributo NAME

O antigo atributo NAME foi substituído pelo atributo ID. Se seus usuários, clientes, etc., utilizam ainda antigos browsers, você pode sem problema nenhum utilizar as duas formas juntas durante o período de migração:

Atributos sem valor

Não devemos esquecer também os atributos que escrevemos sem valor. Por exemplo:

ERRADO:

<*option selected*>

<*input checked*>

<*input readonly*>

CERTO:

<*option selected="selected"*>

<*input checked="checked"*>

<*input readonly="readonly"*>

E assim por diante...

Siga as regras

Existem algumas outras regras para se escrever XHTML. As principais tratam sobre onde os elementos podem ou não aparecer. Por exemplo, você não pode ter um elemento <ul> dentro de um p. Não vamos estudar agora todas essas regras, não vale a pena. Você pode aprender as que lhe são

necessárias ao tentar validar o seu código. Para ter certeza que escreveu dentro dos padrões web seu XHTML, sempre submeta seu arquivo ao validador do W3C.

O conteúdo apresentado a seguir foi traduzido e adaptado de W3Schools (2007).

O quadro abaixo apresenta as TAGs HTML, sua função e os navegadores nos quais cada uma delas é suportada.

Tag	*Descrição*	*Suportado*
<!--...-->	Define um comentário	
<!DOCTYPE>	Define o tipo do documento	
<a>	Define uma âncora	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<abbr>	Define uma abreviação	Netscape 6.0 +, Mozilla 1.0 +, Opera 6.0 +, Safari 1.0 +
<acronym>	Define um acrônimo	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Safari 1.0 +
<address>	Define um elemento de endereço	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<applet>	Depreciado. Define um applet	Internet Explorer 2.0 +, Netscape 2.0 +, Mozilla 1.0 +, Safari 1.0 +, Opera 7.0 +
<area>	Define uma área dentro de um mapa de imagem	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<b>	Depreciado. Define texto em negrito	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<base>	Define uma URL base para todos os links numa página	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<basefont>	Depreciado. Define uma fonte base	Internet Explorer 2.0 +, Netscape 1.0 +, Safari 1.0 +
<bdo>	Define a direção da exibição do texto	Internet Explorer 5.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 7.0 +, Safari 1.0 +
<big>	Define tamanho do texto como grande	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<blockquote>	Define uma citação longa	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<body>	Define o corpo do documento	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
 	Insere uma quebra de	Internet Explorer 2.0 +, Netscape 1.0 +,

	linha forçada	Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<button>	Define um botão	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 5.0 +, Safari 1.0 +
<caption>	Define o título de uma tabela	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
	Depreciado. Define texto centralizado	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<cite>	Define uma citação	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<code>	Define texto de código de computador	– Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<col>	Define atributos para colunas de tabela	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 7.0 +, Safari 1.0 +
<colgroup>	Define grupos de colunas de tabela	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 7.0 +, Safari 1.0 +
<dd>	Define uma descrição de definições	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<del>	Define texto deletado (usado junto com ins)	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 7.0 +, Safari 1.0 +
<dir>	Depreciado. Define uma lista de diretório	Internet Explorer 2.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
	Define uma seção em um documento	Internet Explorer 2.0 +, Netscape 2.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<dfn>	Define termo de lista de definição	Internet Explorer 2.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<dl>	Define uma lista de definição	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<dt>	Define um termo de lista de definição	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
	Define texto enfatizado	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<fieldset>	Define um campo fieldset	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<font>	Depreciado. Estilização do texto	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<form>	Define um formulário	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<frame>	Define uma subjanela (um quadro)	Internet Explorer 2.0 +, Netscape 2.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<frameset>	Define um conjunto de	Internet Explorer 2.0 +, Netscape 2.0 +,

	quadros	Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<h1> to <h6>	Define cabeçalho 1 a 6	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<head>	Cabeçalho do documento	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<hr>	Define uma barra horizontal	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<html>	Define um documento html	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<i>	Define texto itálico	Internet Explorer 4.0 +, Netscape 4.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<iframe>	Define uma subjanela inline	Internet Explorer 3.0 +, Netscape 3.0 +, Mozilla 1.0 +, Opera 5.0 +, Safari 1.0 +
<img>	Define uma imagem	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<input>	Define um campo de entrada	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<ins>	Define texto inserido (usado junto com del)	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<isindex>	Define um campo receptor de uma linha	Internet Explorer 4.0 +, Netscape 4.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<kbd>	Define texto de teclado	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<label>	Define um rótulo de um campo de entrada	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<legend>	Define um título em um fieldset	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
	Define um item de lista	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<link>	Define uma fonte de referência	Internet Explorer 3.0 +, Netscape 4.0 +, Mozilla 1.0 +, Safari 1.0 +
<map>	Define um mapa de imagem	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<menu>	Depreciado. Uma lista de menu	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<meta>	Define meta-informação	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<noframes>	Define uma seção sem quadros (frames)	Internet Explorer 2.0 +, Netscape 2.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<noscript>	Define uma seção sem	Internet Explorer 3.0 +, Netscape 2.0 +,

	scripts	Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<object>	Define um objeto embutido	Internet Explorer 3.0 +, Netscape 4.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
	Define uma lista ordenada	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<optgroup>	Define um grupo de opções	Internet Explorer 6.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<option>	Define uma opção numa lista drop-down	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<p>	Define um parágrafo	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<param>	Define um parâmetro para um objeto	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<pre>	Define texto preformatado	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<q>	Define uma citação curta	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<s>	Depreciado. Define texto riscado	Internet Explorer 2.0 +, Netscape 3.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<samp>	Define exemplo de código de computador	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<script>	Define um script	Internet Explorer 3.0 +, Netscape 2.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<select>	Define um menu tipo “drop-down”	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<small>	Define o tamanho do texto como pequeno	Internet Explorer 2.0 +, Netscape 2.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
	Define uma seção em um documento	Internet Explorer 3.0 +, Netscape 3.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<strike>	Depreciado. Define texto riscado	Internet Explorer 2.0 +, Netscape 3.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
	Define texto enfatizado	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<style>	Define uma determinação de estilo	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<sub>	Define texto subscrito	Internet Explorer 2.0 +, Netscape 2.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<sup>	Define texto superescrito	Internet Explorer 2.0 +, Netscape 2.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +

<table>	Define uma tabela	Internet Explorer 2.0 +,Netscape 1.1 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<tbody>	Define o corpo de uma tabela	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<td>	Define uma célula de uma tabela	Internet Explorer 2.0 +, Netscape 1.1 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<textarea>	Define uma área de texto	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<tfoot>	Define o rodapé de uma tabela	Internet Explorer 4.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 5.0 +, Safari 1.0 +
<th>	Define o cabeçalho de uma tabela	Internet Explorer 2.0 +, Netscape 1.1 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<thead>	Define o cabeçalho de uma tabela	Internet Explorer 3.0 +, Netscape 6.0 +, Mozilla 1.0 +, Opera 5.0 +, Safari 1.0 +
<title>	Define o título do documento	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 5.0 +, Safari 1.0 +
<tr>	Define uma linha de uma tabela	Internet Explorer 2.0 +, Netscape 1.1 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<tt>	Define texto “teletype”	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<u>	Depreciado. Define texto sublinhado	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
	Define uma lista desordenada	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<var>	Define uma variável	Internet Explorer 2.0 +, Netscape 1.0 +, Mozilla 1.0 +, Opera 4.0 +, Safari 1.0 +
<xmp>	Depreciado. Define texto preformatado	Internet Explorer 2.0 - 6.0, Netscape 1.0-7.0, Opera 4-7

CAPÍTULO 4

Introdução ao CSS

CSS (Cascading Style Sheet) é uma linguagem simples proposta pelo W3C para formatar as tags do HTML. As tags foram feitas para mostrar ao browser que tal parte do documento era, por exemplo, um parágrafo, um título, etc. CSS serve para dar formatação às tags deste arquivo HTML, separando assim a formatação da informação.

Então podemos concluir que CSS serve para criarmos o "visual" do site, ou seja, o layout: fontes, boxes, imagens, backgrounds, etc.

Há três formas de se inserir CSS em um arquivo HTML:

CSS inline

Você pode inserir código CSS diretamente em uma tag do HTML, através do atributo style, desta forma: ... conteúdo ... . (Isto tornaria o texto desse parágrafo vermelho.) Usar CSS assim prejudica muito a separação entre formatação e conteúdo, e não é muito diferente de usar formatação HTML.

Inseridas no HTML

Você pode colocar seu código CSS na seção head de seu arquivo, dentro de uma tag style, assim:

```
<style type="text/css">
    <!--
    p{
            color: red;
    }
    -->
</style>
```

(Este código deixaria todos os parágrafos do documento vermelhos.)

CSS externo

O método mais usado. Você armazena seu código CSS em um arquivo separado do HTML. Proporciona a melhor separação entre formatação e conteúdo, além da flexibilidade de se ter um único arquivo CSS servindo o site inteiro.

Existem diversas técnicas para desenvolver arquivos CSS em conjunto com arquivos HTML, muitas das quais podemos encontrar em livros e revistas, escritos por especialistas da área como Jeffrey Zeldman, que mencionamos anteriormente.

Zeldman, menciona em seu livro "Desenvolvendo Sites Compatíveis", que devemos utilizar 3 tipos de arquivos CSS principais, possibilitando a compatibilidade entre os diversos browsers existentes, antigos ou modernos ou mesmos para outros dispositivos como PDA`s e celulares.

A idéia baseia-se na utilização de um arquivo basico.css, o qual conterá apenas o código CSS para formatação básica do site, a fim de, deixá-lo legível em qualquer agente de usuário, com um código semelhante a este:

```
body {

    font-family : Arial, Helvetica, sans-serif;

    font-size : 14px;

    color : #000;

    background-color : #FFF;

}
```

Este código tem como objetivo dar contraste ao conteúdo em relação ao fundo e ter uma família e tamanho maior que o padrão de fonte para facilitar a leitura.

Este arquivo deve ser linkado com o arquivo (X)HTML, utilizando-se a tag <link> dentro da seção head, pois tanto os navegadores modernos como os antigos interpretam esta tag.

```
<link rel="stylesheet" type="text/css" href="basico.css" media="screen">
```

O atributo rel – relativo – identifica que o link é relativo a uma folha de estilo CSS.

O atributo type – tipo – identifica o tipo de conteúdo do arquivo: texto CSS.

O atributo href – hiperlink de referência – identifica a localização relativa ao documento (X)HTML do arquivo CSS.

O atributo media – mídia – identifica o objetivo deste arquivo, neste caso, o objetivo é a formatação do arquivo para apresentação na tela. Outra media bastante utilizada é a print, quando o arquivo tem como objetivo formatar / estilizar um arquivo para impressão.

O arquivo mais importante que será desenvolvido é o principal.css, o qual conterá toda a formatação padrão do site e sua estruturação, com foco apenas nos navegadores modernos, por isso utilizamos uma forma de ligação entre arquivo CSS e (X)HTML, interpretada apenas por estes navegadores:

```
<style type=”text/css” media=”screen”>
    <!--
            @import url(‘principal.css’);
    -->
</style>
```

Pode observar que utilizamos a tag <style> com os mesmos atributos encontrados na tag <link> e é importante ressaltar que essa tag como foi implementada a pouco tempo é interpreatada apenas pelos navegadores modernos.

Todo o conteúdo escrito dentro da tag <style> é interpretada como CSS e devido a isso o comentário HTML utilizado dentro da mesma não surtirá efeito algum, sendo que o comentário CSS é:

```
/* Comentário CSS */
```

A utilização por tanto do comentário HTML é para que o código CSS escrito dentro da tag <style> seja comentado caso o usuário acessar a página por um navegador antigo e evitar que o texto desconhecido seja apresentado na tela.

Baseado em experiências profissionais coordenando equipes de desenvolvimento, onde vários profissionais realizam a manutenção do código CSS, recomendo que a subdivisão do arquivo principal.css. Trabalhando com o objetivo de realizar e estruturar os elementos em comum a todas as páginas do site e/ou ao sistema web e desenvolvendo arquivos menores com código especifico, formatando e estruturando as perticularidades de cada documento individualmente, linkando desta forma:

```
<style type=”text/css” media=”screen”>
    <!--
            @import url(‘principal.css’);
            @import url(‘empresa.css’);
    -->
```

```
</style>
```

Desta forma a administração do código é facilitada e diminui a possibilidade de código redundante ou mesmo desnecessário que aumenta o tamanho do arquivo e o tempo necessário para o carregamento e renderização da página.

O terceiro tipo de arquivo que devemos utilizar é quando ocorrem incompatibilidades entre o Internet Explorer e os demais navegadores:

```
<!--[if IE 6]>
    <link rel="stylesheet" media="screen" href=" hacks_ie6.css" />
<![endif]-->
```

Provavelmente ao ler este código achará estranho perceber que está comentado, e é neste momento que irá perceber que mesmo na imperfeição existem possibilidades. Hack é nome dado ao código CSS ou (X) HTML que aproveita-se de algum erro / despadronização / particulariedade de um navegador para corrigir sua incompatibilidade do código em relação com outros navegadores.

A Microsoft ao desenvolver a Internet Explorer criou esta forma condicional para seu código, que utilizamos para aplicar código CSS apenas para ele ou para alguma versão específica. Pois, existem diferenças de interpretação entre a Internet Explorer e o Firefox, como entre o Firefox e o Opera, entre a Internet Explorer 6.0 e 7.0 e assim por diante.

Recomendo que não utilize esta opção, caso realmente perceba a necesssidade use-a, mas antes revise todo o seu código CSS, pois na maioria dos casos, o HACK I.E. é utilizado por mau formulação do arquivo principal.css.

Sintaxe

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

O seletor é o elemento do HTML que receberá a formatação. A propriedade é o atributo deste elemento que você deseja manipular. Cada atributo tem uma lista de valores possíveis.

```
body {
    background-color: black;
}
```

Acima definimos que o atributo BACKGROUND-COLOR do seletor BODY receberá o valor BLACK. Ou seja, o corpo do documento terá cor de fundo preta.

Agrupando seletores

Para definir atributos iguais para vários elementos, é possível agrupar vários seletores, separando-os por vírgula:

```
p, h1, div { background-color: blue; }
```

Neste caso, todos os parágrafos, títulos de nível 1 (h1) e divs terão cor de fundo azul.

Definindo valores

Se o valor tiver uma palavra composta, você deverá colocá-la entre aspas simples:

```
p {
    font-family: 'Lucida Sans';
 }
```

Existem algumas propriedades que podem agrupar valores de outras em uma só declaração. Veja o código abaixo:

```
div {
   margin-top: 10px;
   margin-right: 20px;
   margin-bottom: 5px;
   margin-left: 35px;
}
```

A propriedade "margin" pode agrupar os valores das propriedades "margin-top", "margin-right", "margin-bottom" e "margin-left", obedecendo à seguinte sintaxe:

```
div {
   margin: top right bottom left;
}
```

Assim, o código se torna menor e mais limpo, fácil de ser entendido. O exemplo acima ficaria:

```
div {
   margin: 10px 20px 5px 35px;
}
```

Criando estilos usando "Class" e "Id"

Os atributos Class e Id são usados para "entiquetar" elementos específicos do documento HTML. Assim, esses elementos podem receber uma formatação individual, personalizada.

Usando o atributo Class

O atributo Class pode ser usado para definir diferentes estilos para um mesmo tipo de elemento (tag) HTML. Vamos supor que você tenha em sua página, dois tipos de parágrafos: um com o texto alinhado à direita e o outro com o texto alinhado ao centro. Você pode criar as seguintes classes no CSS:

```
p.direita {
      text-align:right;
}
p.centro {
      text-align:center;
}
```

No HTML, você atribui estas classes da seguinte forma:

```
<p class="direita">
      Acessibilidade
</p>
<p class="centro">
      Acessibilidade
</p>
```

Se no seletor CSS, você tirasse a tag P da frente do atributo Class, deixando apenas:

```
.centro {
    text-align:center;
}
```

Você poderia atribuir esta classe para qualquer elemento HTML (não só parágrafos), usando o atributo: class="centro".

Podemos atribuir várias classes a um mesmo elemento. A última classe no css será a que prevalecerá. Exemplo:

```
<p class="direira centro">Acessibilidade</p>
```

Logo acima, atribuímos duas classes para o parágrafo. Se as duas classes tiverem definindo a cor da letra, prevalecerá a cor da última classe escrita no CSS.

Usando o atributo Id

O atributo Id é um pouco diferente do atributo Class. Você pode atribuir a mesma classe a vários elementos, mas o Id é um identificador único na estrutura do documento. Ou seja: você não pode ter dois elementos com o mesmo Id. Exemplo:

```
p#direita {
    text-align:center;
}
```

O documento deverá ter somente UM elemento P com o Id DIREITA. E para atribuir o Id no HTML:

```
<p id="direita">Acessibilidade</p>
```

Antigamente, não tão antigamente assim, talvez, quase que com certeza, se quisesse formatar o texto de um parágrafo alinhado ao centro com a fonte Verdana e um tamanho médio, acredito que desenvolveria algo parecido com isso:

```
<center>
  <p>
    <font face=”Verdana” size="2">
        E aqui dentro ia o texto.
    </font>
```

</p>

Isso tudo dentro do arquivo HTML, deixando o código complicado, sujo e, conseqüentemente, maior.

Sem falar naqueles problemas típicos do chefe mudando de idéia sobre o tamanho ou a cor da fonte, fazendo você perder um tempo precioso procurando as tags para mudar um simples valor.

Com CSS temos total controle sobre a formatação do texto: alinhamento, cor, família de fonte, altura de linha, espaçamento das letras, etc...

Você pode definir todas estas propriedades numa mesma declaração. Por exemplo:

p {font: italic bold 10px Verdana;}

PROPRIEDADES CSS

Acompanhe a tabela a seguir para saber como aplicar cadas propriedade CSS, no seu desenvolvimento.

FONTES:

Propriedade	Descrição	Valores Aceitos	Exemplos
font-family	Família de Fontes a ser utilizada no texto, em ordem de propriedade.	Nomes das famílias de fontes ou seus nomes genéricos, separados por vírgula. Os nomes genéricos reconhecidos são: serif, sans-serif, cursive, fantasy e monospace.	body{ font-family: Geórgia, “Times New Roman”, Times, serif;}
font-style	Estilo de fonte normal ou itálico (inclinado)	normal, italic, oblique	p.italico { font-style: italic; }
font-variant	Texto normal ou todas em maiúsculas	* normal: tal como digitado * small-caps: Converte todo o texto	h1, h2 { font-variant: small-caps}

		para maiúsculas. Obs: No navegador Netscape, as letras maiúsculas no texto original são destacadas em relação as demais	
font-weight	Corresponde ao peso ou “espessura” da fonte	Bold, bolder, lighter, normal, 100, 200, 300, 400, 500, 600, 700, 800, 900	p em { font-weight: bolder }
font-size	Tamanho da fonte	* Valor em unidades de medida (pt, px, pc, %) * Valores constantes (xx-small, x-small, small, médium, large, x-large, xx-large	p small { font-size: 10pt } p big { font-size: x-large } ul { font-size: 200% }
Font	Sintaxe simplificada (atalho) para a especificação da fonte, em que são omitidos os nomes das propriedades e usados somente seu valores, separados por espaços.	Os mesmos valores aceitos pelas propriedades: font-style, font-variant, font-weight, font-size, line-height, font-family	Li { font: italic small-caps bold 16pt Verdana; color:#FF6633; }

TEXTO

Propriedade	Descrição	Valores Aceitos	Exemplos
text-align	Alinhamento horizontal de texto	left (esquerda) right (direita)	p cite { text-align: center }

	em bloco	center (centralizado) justify (justificado)	

text-decoration	Efeitos especiais (adornos) no texto	none (nenhum), underline (sublinhado), overline (linha acima do texto), line-throught (texto riscado), blink (piscante). Use mais de um valor separando-os com espaço.	p.sublinhado {
			
			text-decoration: underline;
			
			}
			
			.cancelado { text-decoration: underline line-throught; }

text-indent	Endentação da primeira linha de um texto em bloco.	Valores definidos em unidades de medida (px, pt, mm, % etc.). Podem ser usados valores negativos.	div.recuado{ text-indent: 2cm; }
letter-spacing	Distância entre as letras do texto	* normal. * valor definido em unidades de medida (px, pt, mm, % etc.). São aceitos valores negativos.	h2 small { letter-spacing: 1.2ex }
word-spacing	Distância entra as palavras do texto	* normal * valor definido em unidades de medida (px, pt, mm, %, etc.).	code { word-spacing: 150%; }
text-transform	Tratamento das letras maiúsculas e minúsculas	* none: padrão * capitalize: primeira letra de cada palavra em maiúscula. * uppercase: todas maiúsculas * lowercase: todas	div.titulo { text-transform: capitalize }

		minúsculas	
white-space	Tratamento dos espaços em branco no texto	* normal: mais de um espaço é tratado com um só. * pre: mais de um espaço produz um espaço maior. * nowrap: múltiplos espaços são considerados como parte da palavra e não são quebrados.	code { white-space: pre }
line-height	Altura da linha do texto	Valores definidos em unidades de medidas (px, pt, mm, % etc).	p { line-height:220%; }
vertical-align	Alinhamento vertical da linha de texto.	baseline, sub, super, top, text-top, middle, bottom, text-bottom, %;	li { vertical-align: bottom }

EFEITOS VISUAIS E POSICIONAMENTO

Propriedade	Descrição	Valores Aceitos	Exemplos
Width Height	Referem-se, respectivamente, à largura e à altura de um elemento em bloco.	Valores definidos em unidades de medida (px, pt, mm, % etc.).	div.coluna1 { width:75%;} p { height: 200px; }

Position	Determina se um elemente terá sua posição controlada pelas propriedades left (esquerda), right (direita), top (superior), bottom (inferior)	* static: não é controlado, seguindo o fluxo normal do documento.
		
		* relative: a posição é controlada pelas propriedades left, right, top e bottom, ainda dentro do fluxo do documento.
		
		* absolute: posição fixa, em uma nova camada que não segue o fluxo do documento.
		
		* fixed: posição fixa na janela, em uma nova camada que não obedece, inclusive, a rolagem. Não funciona no IE.
			div.fixo {
			position: absolute;
			top: 50px;
			left: 80px;
			}

Left
 	
right
 	
top
 	
bottom
	Define os limites esquerdo, direito, superior e inferior dos elementos posicionados (propriedades “position” com o valor “relative”, “absolute”	* auto: determinado pelo navegador.
		
		* um valor definido em uma das unidades de medida (px, pt, mm, % etc.).
			.logotipo {
			position:relative;
			top: 1cm;
			left: 20px;
			}

	ou “fixed”).		

Overflow	Trata o texto que excede a área delimitada para um elemento posicionado com dimensões fixas.	* auto: deixa o controle a cargo do navegador, que normalmente insere barras de rolagem no elemento.
		
		* hidden: instrui o navegador a preserver o tamanho especificado para o elemento. O tratamento ao texto excedente dependerá da propriedade “clip”, analisada a seguir.
			div.fixoRolavel {
			position:absolute;
			left:50px;
			right:120px;
			top:300px;
			bottom:380px;
			overflow:scroll;
			}

Clip	Define a apresentação do texto excedente em um elemento com a propriedade “overflow” setada para “auto”, “hidden” ou “scroll”.	* auto: o navegador controla o texto excedente.
		
		* um retângulo definido por “topo”, “direita”, “base” e “esquerda”
			div.bloxoFixo {
			position: absolute;
			clip: rect(50px, 15px, 70px, 35px);
			}

Display	Controla a forma de exibição de um elemento	* block: cria um elemento em bloco. O navegador incluirá uma quebra de linha antes e depois.
		
		* inline: Valor padrão, Cria um elemento em linha no fluxo normal do documento.
		
		* list-item: refere-se a um item da lista. Consulte as propriedades de
			span.novoBloco {
			display: block
			}
			
			img.oculto {
			display: none;
			}

		listas. * none: o elemento não é exibido na tela, e nenhum espaço é reservado para ele.	

Visibility	Visibilidade de um elemento na página	* visible: elemento visível (padrão)
		
		* hidden: o elemento não aparece na página, porém seu espaço no conteúdo é reservado.
			img.invisivel{
			
			visibility: hidden;
			
			}

Float	Determina como o texto fluirá ou "flutuará" em volta de um elemento em bloco.	* none: sem efeito
		
		* left: o elemento ficará à esquerda do fluxo de texto, ou seja, o texto fluirá à sua direita.
		
		* right: o elemento ficará a direita do fluxo do texto.
			div.foto{
			
			float:left;
			
			}
			
			img.btVoltar {
			
			float:right;
			
			}

Clear	Define a forma como um elemento se comportará em relação a outro elemento adjacente com a propriedade "float" setada para left ou right	* none: sem efeito * left: não aceita que outro elemento "flutue" ao seu lado esquerdo. *right: não aceita que outro elemento "flutue" ao seu lado direito. * both: bloqueia ambos os lados do elemento contra elementos flutuantes.	img { clear:right; } div.logo { clear: both }

z-index	Especifica o eixo “Z” de um elemento posicionado, exibindo-o em uma camada independente, acima ou abaixo dos demais elementos da página.	* auto: determinado pela ordem do elemento no código fonte.
		
		* um valor inteiro, que indica a posição do elemento nas “camadas” da página.
			img.popup{
			position:absolute;
			left: 50px;
			top: 200px;
			z-index:2;
			}

MARGENS E BORDAS

Propriedade	Descrição	Valores Aceitos	Exemplos

padding-top
 	
padding-rigth
 	
padding-bottom
 	
paddin-left
	Especificam valores independentes de “padding” superior, direito, inferior e esquerdo de um elemento. Padding é o espaço entro o conteúdo de um elemento e os limites de sua caixa delimitadora.	Definidos em unidades de medida (px, pt, mm, % etc.).	.colunaDireita{
			padding-right: 5px;
			}

Padding	Atalho para as quatro propriedades de padding:
	
	* Um valor: é aplicado para as quatro propriedades.
	
	* Dois valore: o primeiro é aplicado para top e bottom, e o segundo é aplicado para right e left.
	
	* Três valores: o primeiro valor é
		Definidos em unidades de medida (px, pt, mm, % etc.).	blockquote{
			padding: 1cm 20px 0.5cm 10px;
			}

	aplicado para top, o segundo para right e left, o terceiro para bottom. * Quatro valores: aplicados respectivamente para top, right, bottom e left.		

margin-top
 	
margin-right
 	
margin-bottom
 	
margin-left
	Especificam valores independentes para as quatro margens de um elemento. A margem é o espaço entro o limite da caixa delimitadora e sua borda.	Definidos em unidades de medida (px, pt, mm, % etc.).	.marginDireita {
			
			margin-right: 5px;
			
			}

Margin	Atalho para as quatro propriedades de margin:
	
	* Um valor: é aplicado para as quatro propriedades.
	
	* Dois valore: o primeiro é aplicado para top e bottom, e o segundo é aplicado para right e left.
	
	* Três valores: o primeiro valor é aplicado para top, o segundo para right e left, o terceiro para bottom.
	
	* Quatro valores: aplicados respectivamente para
		Definidos em unidades de medida (px, pt, mm, % etc.).	blockquote {
			
			margin: 1cm 10px 0.5cm 10px;
			
			}

	top, right, bottom e left.		

border-top-style
 	
border-ritght-style
 	
border-bottom-style
 	
border-left-style
	Define o estilo de apresentação das bordas superior, direita, inferior e esquerda de um elemento	* none: sem bordas
		
		* hidden: idem a none
		
		* dotted: pontilhada
		
		* dashed: linhas curtas
		
		* solid: linha sólida
		
		* double: duas linhas sólidas
		
		* groove: entalhe
		
		* ridge: outro tipo de entalhe, simulando 3D
		
		* inset: moldura interna
		
		* outset: moldura externa
			div.quadro {
			
			border-top-style: solid;
			
			border-right-style: double;
			
			border-bottom-style: groove;
			
			border-left-style: outset;
			
			}

boder-style	Atalho para as quatro propriedades de estilo das bordas:
	
	* Um valor: é aplicado para as quatro propriedades.
	
	* Dois valore: o primeiro é aplicado para top e bottom, e o segundo é aplicado para right e left.
	
	* Três valores: o primeiro valor é
		Os mesmos valores aceitos pelas propriedades de bordas supracitadas	div.quadro{
			
			border-style: inset;
			
			}

	aplicado para top, o segundo para right e left, o terceiro para bottom. * Quatro valores: aplicados respectivamente para top, right, bottom e left.		

border-top-width
 	
border-right-width
 	
border-bottom-width
 	
border-left-width
	Define a espessura das bordas de um elemento. Para que as bordas sejam exibidas, um dos estilos acima deve ser setado.	* thin: fina
		
		* médium: média
		
		* tick: grossa
		
		* Também são aceitos valores em unidades de medidas (px, pt, mm, % etc.)
			.destaque{
			
			border-style: solid;
			
			border-top-width:5px;
			
			border-right-width: thin;
			
			border-bottom-width:5px;
			
			border-left-width:10px;
			
			}

border-width	Atalho para as quatro propriedades de bordas:
	
	* Um valor: é aplicado para as quatro propriedades.
	
	* Dois valore: o primeiro é aplicado para top e bottom, e o segundo é aplicado para right e left.
	
	* Três valores: o primeiro valor é
		Os mesmos valores aceitos pelas propriedades de espessura das bordas.	.destaque{
			
			border-style:ridge;
			
			border-width: 5px 1px;
			
			}

	aplicado para top, o segundo para right e left, o terceiro para bottom. * Quatro valores: aplicados respectivamente para top, right, bottom e left.		

border-top-color
 	
border-right-color
 	
border-bottom-color
 	
border-left-color
	Define as cores das bordas superior, direita, inferior e esquerda de um elemento.	Especificação de cor (constante, hexadecimal ou rgb)	.quadroFantasia {
			
			border-style: dotted;
			
			border-top-color: #000099;
			
			border-right-color: #0066CC;
			
			border-bottom-color: #00CCCC;
			
			border-left-color: #000099;
			
			}

border-color	Atalho para as quatro propriedades de cor de borda:
	
	* Um valor: é aplicado para as quatro propriedades.
	
	* Dois valore: o primeiro é aplicado para top e bottom, e o segundo é aplicado para right e left.
	
	* Três valores: o
		Especificação de cor (constante, hexadecimal ou rgb)	.quadroAzul {
			
			border-color: marron;
			
			border-style: solid;
			
			border-width: 5px;
			
			}

	primeiro valor é aplicado para top, o segundo para right e left, o terceiro para bottom. * Quatro valores: aplicados respectivamente para top, right, bottom e left.		

border-top
 	
border-right
 	
border-bottom
 	
border-left
	Atalho para especificar, de uma só vez, as propriedades de estilo, largura e cor, para cada uma das quatro bordas de um elemento.	Os mesmos valores aceitos pelas respectivas propriedades de estilo, largura e cor.	.quadroFantasia {
			
			border-top: dotted 3px rgb(255, 160, 0);
			
			border-bottom: double 5px;
			
			}

Border	Atalho para especificar, de uma só vez as propriedades de estilo, largura e cor para todas as bordas de um elemento.	Os mesmos valores aceitos pelas respectivas propriedades de estilo, largura e cor.	.quadroFantasia {
			
			border: dotted 3px #FF3300;
			
			}

COR E BACKGROUND

Propriedade	Descrição	Valores Aceitos	Exemplos
Color	Cor do texto do elemento	* Nome da cor * Notação Hexadecimal * Valores RGB	body {color: navy}
background-color	Cor do fundo de um elemento	* Nome da cor * Notação	body {background-color: red}

		Hexadecimal * Valores RGB	

background-image	Imagem de fundo de um elemento. Por padrão, a imagem é repetida em “ladrilho”, de modo a preencher toda a área.	URL do arquivo de imagem, informada em um descritor url	body {
			
			background-imagem: url(“logo.gif”);
			
			}

background-repeat	Define se a imagem de fundo de um elemento deve ser repetida como “ladrilho”	* no-repeat: a imagem não é repetida
		
		* repeat (padrão): a imagem é repetida em “ladrilho”, ocupando toda a área do elemento.
		
		* repeat-x: repete a imagem em uma linha.
		
		* repeat-y: repete a imagem em uma coluna
			body {
			
			background-imagem: url(“logo.gif”);
			
			backround-repeat: no-repeat;
			
			}

Background-attachment	Define se a imagem de fundo deve rolar ou não com a página	* scroll (padrão): a imagem rola com a página.
		
		* fixed: a imagem permanece fixa quando a página é rolada.
			body {
			
			background-imagem: url(“esferas.gif”);
			
			background-attachment: fixed;
			
			}

background-position	Posição da imagem de fundo do elemento	Valor definido em unidades de medida (px, pt, mm, % etc.), ou pelas constantes “top” (topo), “center”	body { background-imagem: url(“esferas.gif”); background-position:

		(centro), “bottom” (base), “left” (esquerda), “right” (direita). Use dois valores para especificar a posição horizontal e vertical.	top center; }

Background	Atalho para especificar, de uma só vez, todas as propriedades de imagem de fundo de um elemento.	Os mesmos valores aceitos pelas propriedades “background-attachment”,
		
		“background-color”, “background-image”, “background-position” , “background-repeat”, separadas por um espaço
			body {
			
			background: url(“esferas2.gif”) repeat-y fixed right;
			
			}

LISTAS

Propriedade	Descrição	Valores Aceitos	Exemplos

list-style-type	Define o formato do marcador a ser exibido nos itens da lista	Para listas não ordenadas (ul): “square” (retângulo), “circle” (círculo vazado) ou o valor padrão “disc” (círculo preenchido).
		
		Para listas ordenadas (ol): “decimal” (1, 2, 3...), “decimal-leading-zero” (01, 02, ...)(Somente Netscape), “lower-roman” (i, ii, iii, iv...), “upper-roman” (I, II, III, IV....), “lower-
			ul{
			
			list-style-type: square;
			
			}

		alpha” (a, b, c, d...), “upper-alpha” (A, B, C, D...)	

list-style-image	Define uma imagem a ser usada como marcador nos itens de uma lista	* none: nenhum
		* url do arquivo de imagem, informado em um descritor url ()
			ol{
			list-style-image: url (“seta.gif”);
			}

list-style-position	Define se os marcadores de uma lista ficam dentro ou fora dos limites da caixa delimitadora do elemento.	* inside: dentro
		* outside: fora
			ol{
			list-style-position: inside;
			}

list-style	Atalho que permite especificar, de uma só vez, as propriedades de formato do marcador, posição e imagem de itens de uma lista.	Os mesmos valores aceitos pelas propriedades: “list-style-type”, “list-style-position” e “list-style-image”	ol{
			list-style: upper-roman inside;
			}

CAPÍTULO 5

Aplicando CSS

Seletores

Seletores Complexos

Podemos criar alguns tipos de seletores úteis para o desenvolvimento.

Seletores Agrupados

Seleteroes agrupados são usados quando queremos definir um mesmo estilo a vários elementos diferentes do documento HTML. Os seletores são separados por vírgulas. Veja abaixo:

div, p, span, h1 {width:200px;}

No código acima, definimos que os elementos div, p, span e h1 terão a largura de 200 pixels.

Seletores Encadeados

Usamos seletores encadeados quando nós queremos que algum elemento que esteja dentro de uma tag específica tenha um estilo personalizado. Neste caso, os seletores são separados apenas por espaços. Vamos entender mais com o exemplo abaixo:

div p a {color:blue;}

Assim, definimos que todos os links ("a") que estão dentro de algum parágrafo ("p"), que estão dentro de algum "div" terão a cor "blue" (azul).

Especificidade

No decorrer do nosso desenvolvemos entedemos o segnificado do nome da linguagem CSS (Cascading Style Sheet – Folha de Estilo em Cascata), que especifica que o CSS é interpretado da esquerda para direita e de cima para baixo, sendo que o último seletor declarado vai sobrescrever este mesmo seletor em suas propriedades semelhantes, que possam ter sido declarados anteriormente. Como por exemplo:

```
p{
    color : blue;
    font-size : 1em;
}
```

```
p{
    color : red;
}
```

Neste exemplo, os parágrafos irão ter o tamanho de fonte de 1em, mas a cor será sobreescrita ficando com a cor vermelha e não azul.

Em alguns casos isto pode não ocorrer, isto quando falamos de especificidade, que ser específico no tratamento do código CSS, e a melhor maneira de dominar isto é pontuando os nossos possíveis seletores, atrabés de sua relevância quanto a especificidade. Confira a pontuação:

SELETOR	PONTUAÇÃO
TAG	1
CLASS	10
ID	100

Ao determinamos um seletor, devemos levar em consideração além da ordem da escrita sua pontuação, que determina a especificidade. Por exemplo:

```
div#noticias p{
    color : black;
}
```

Este seletor possui duas tags e um ID, se utilizarmos a tabela de pontuação de especificidade podemos pontuá-lo com 102 pontos.

```
div p{
    color : red;
}
```

Este outro seletor com apenas duas tags irá somar apenas 2 pontos.

Neste caso devido a especificidade, mesmo o segundo seletor ter sido declarado por últimо, a cor dos parágrafos da div com id “noticias” será preta, conforme declaração do primeiro seletor.

CAPÍTULO 6

TABLELESS

Tableless é uma metodologia de trabalho que se baseia em semântica e onde tabelas são usadas apenas para dasdos tabulares e não mais para estruturar a página. Hoje, os Padrões Web (Web Standards) estão bem mais maduros e já se pode utilizá-los com bem menos dificuldades do que anteriormente.

Benefícios

Redução de custos

Ao contrário do método tradicional, que quase sempre é linear, utilizando Web Standards a equipe de negócio, design e programação podem atuar simultânea e independentemente. Isso acelera bastante a produção e torna a manutenção de um projeto web muito simples. Além da economia de tempo, há redução de custos na compra de licenças de software.

Desenvolvimento mais ágil

Significativa parte do tempo no processo de desenvolvimento da maioria dos websites é desperdiçada em retrabalho de design para que o HTML fique com aparência profissional. O desenvolvimento Web Standards corta o retrabalho praticamente a zero.

Tecnologia acessível

Um designer treinado em Web Standards não depende de um programador para saber o que pode ou não ser implementado em um projeto web; e vice-versa.

Facilidade para criar versões de layout

Uma vez que conteúdo, programação e design estão em camadas separadas, é incomparavelmente mais rápido desenvolver versões de layout para uma mesma tela.

Paz entre designers e programadores

Com a separação entre conteúdo e layout promovida pelos Web Standards, designers e programadores podem trabalhar simultaneamente e de maneira independente.

Perfeita integração com .NET, Java, PHP, ASP etc.;

Web Standards & Tableless têm total integração com as principais tecnologias de mercado. Pequenas mudanças precisam ser feitas em uma ou outra plataforma.

Lembrando sempre, que cada caso é um caso.

Por exemplo: se você tem um grande portal, provavelmente a maior vantagem que você vai obter, será a economia de banda.

Controle sobre o projeto

Com a metodologia Web Standards, evita-se que apenas um membro da equipe tenha domínio exclusivo sobre o desenvolvimento, assegurando que qualquer desenvolvedor poderá dar continuidade ou realizar futuras manutenções.

Melhor visibilidade no Google

Optar pelos Web Standards significa valer-se da estrutura semântica simples e coerente do HTML, fator primordial para ter um melhor posicionamento nas ferramentas de busca.

Velocidade do website

O código HTML se torna muito mais compacto ao se separar conteúdo, design e programação, conforme os Web Standards. Além disso, a tecnologia Tableless permite que o navegador interprete as informações de layout (em um arquivo CSS) de 30% a 70% mais rapidamente.

Acessibilidade

O uso de Web Standards facilita muito a aplicação de normas de acessibilidade. Isso garante o acesso ao website, não importando que combinação de navegador e plataforma o usuário possua.

Cabe agora aos desenvolvedores atualizarem seus conhecimentos e revolucionar a Internet. Como nós dizemos: primeiro, a internet mudou a vida das pessoas; agora, são as pessoas que vão mudar a internet.

CAPÍTULO 7

Ajax Acessível

Atualmente, muitas entidades estão envolvidas com a criação de padrões e metodologias para desenvolvimento de aplicações acessíveis. Um dos aspectos mais críticos é o Ajax (Asyncronous JavaScript and XML). Muitos grupos de desenvolvimento espalhados por aí hoje trabalham afim de criar esses padrões, entre muitas estão a WAI (Web Accessibility Initiative's), WAT-C (Web Accessibility Tools Consortium) entre outras. Mas para podermos ter uma idéia de como deixar uma aplicação com Ajax mais acessível, devemos nos interar sobre o comportamento de alguns leitores de tela.

Comportamento dos Leitores de Tela

Basicamente um leitor de tela consiste em um software que que faz uma varredura da página, coloca seu conteúdo no seu buffer virtual, feito isso ele libera o conteúdo que foi gravado em seu buffer para o usuário. Sem esse buffer, o leitor de tela seria limitado a acessar somente parte dos elementos de um site, como âncoras e elementos da interface, ficando praticamente impossibilitado de fornecer ao usuário condições de interagir com os demais elementos e seus nós descendentes no conteúdo, tais como listas, imagens, tabelas entre outros.

Como informar ao leitor de tela que ocorreu uma mudança no conteúdo

Se o conteúdo for alterado via Script, de algum modo isso deve ser informado ao leitor de telas. Como não há ainda um mecanismo que detecte essa umdança no conteúdo, um usuário de leitor de tela nunca ficará sabendo da mudança, ou ainda, será informado, mas precisará ler todo o documento pra descobrir o que foi alterado.

A última linha de leitura é aquela que contém o elemento ativado pelo usuário para produzir o novo conteúdo, dessa maneira, o usuário não terá a mínima idéia de como localizar a mudança de conteúdo. Uma maneira para resolver isso é utilizando o método focus, que é padrão ECMAScript, que pode ser utilizado para colocar o foco no local da página onde o conteúdo foi alterado, porém, para que isso funcione, o elemento alvo deve ser habilitado a receber o foco. No HTML e XHTML os elementos habilitados a receberem foco são: a, area, button, input, object, select e textarea.

Em XHTML 2 todos os elementos podem receber foco, mas nas atuais especificações para HTML 4.01 e XHTML 1.x somente elementos âncora e elementos de interface podem receber foco. Para tentar contornar isso, a WAI-PF[1], propôs a Dynamic Accessible Web Content Roadmap[2], onde é sugerido que se use -1 para o tabindex, em elementos que não podem receber foco, e formalizou tal proposta em States and Adaptable Properties Module[3]. A necessidade de que todos elementos possam receber foco foi reconhecida pela Web Applications 1.0[4].

O atributo tabindex aceita valores entre 0 e 32767. Um valor positivo determina a ordem em que um elemento deve ser acessado quando a navegação se dá pelo teclado. Um valor igual a 0 siginifica que esse elemento deve ser acessado de acordo com sua ordem dentro do arquivo HTML. Atribuir

o valor 0 para o tabindex em um elemento que não seja âncora ou que não seja da interface, significa que ele está sendo habilitado a receber foco, e atribuir o valor -1 para o mesmo, significa habilita-lo a receber foco via ECMAScript, contudo, isso pode causar certa confusão para usuários que navegam por meio do teclado, pois ao atribuir -1 para o tabindex desse elemento, ele estará habilitado para receber foco via ECMAScript, mas não será colocado na ordem de tabulação dos elementos do site.

Estruturando conteúdo para aplicações AJAX

Existem várias maneiras de estruturar conteúdos para que uma aplicação Ajax funcione corretamente em um leitor de tela. A mais simples e talvez a melhor seja a de se fazer com que as partes da aplicação que exigem ativação pelo usuário, estejam em elementos para formulários. Esta solução permite informar ao usuário sobre mudanças, focando na parte do documento que foi alterada. Isto requer que o usuário desabilite/habilite o mode buffer virtual, mas é isso exatamente que o uauário espera quando interage com formulários.

Se a parte alterada do documento está no formulário, então surge um problema; the changed part of the content is in the form itself, then this presents a problem; although the user can get at the text, they won't be able to interact with the content as they would if they were in virtual buffer mode. Tomemos como exemplo para ilustrar esta situação uma tabela de dados incorporada em um formulário (uma estruturação destas poderia ser melhorada, mas vamos considerar para efeito de exemplo que seja uma solução plausível). Se um dado em uma célula da tabela for atualizado em resposta ao evento onreadystatechange o foco pode ser dado à célula e o leitor anunciará a atualização. O problema é que a tabela contento o texto atualizado não será mais reconhecida como uma tabela e o usuário será incapaz de localizar os headers da tabela e navegar o restante da tabela. Para reverter esta situação o usuário terá que voltar ao modo buffer virtual. Como os leitores de tela sempre anunciam o atributo title, este poderia ser usado para os controles de formulário que possam receber foco (inclusive elementos que não sejam de interface com um específico valor para o atributo tabindex) e que não estejam associados a um label. A seguir um exemplo de como atualizar o conteudo de uma célula de tabela em resposta ao evento onreadystatechange.

```
var objCurrent = document.getElementById('update');

var objReplacement = document.createElement('td');



// Set the title attribute to prompt the user to change mode

// This should use simpler language than used here, as the user

// isn't likely to understand the concept of a virtual buffer

objReplacement.setAttribute('title', 'Switch to virtual buffer');

// When the element loses focus, remove the attribute for other
```

```
// user agents
objReplacement.onblur = function(){this.removeAttribute('title');};

// Set a negative tabindex attribute value so the element
// can receive focus
objReplacement.tabIndex = -1;

objReplacement.setAttribute('id', 'update');
objReplacement.appendChild(document.createTextNode(strResult));

// Replace the existing node with the new node
objCurrent.parentNode.replaceChild(objReplacement, objCurrent);
// Set focus to the element
objReplacement.focus();
```

Se a interação com a aplicação não for por formulário, o mais importante é avisar ao usuário para que ele desabilite o modo buffer virtual (tal como ele faria para interagir com um formulário). Isto não é tão simples como parece, pois será necessário usar instruções muito claras ao usuário do leitor de tela que na sua maioria desconhece os detalhes técnicos do software que estão utilizando — da mesma forma que usuários de navegadores visuais não necessariamente conhecem como redimensionar os textos no navegador. Uma solução simplista é a de disponibilizar uma página de ajuda, facilmente localizável na aplicação, explicando buffer virtual e contendo uma tabela mostrando os atalhos para habilitá-lo e desabilitá-lo nos leitores de tela mais populares. A tabela a seguir mostra os atalhos de teclado para habilitar e desabilitar o buffer virtual dos leitores de tela que nós usamos para testes.

Links:

[1] - http://www.w3.org/WAI/PF/

[2] - http://www.w3.org/WAI/PF/roadmap/

[3] - http://www.w3.org/WAI/PF/roadmap/DHTMLRoadmap040506.html#focus

[4] - http://whatwg.org/specs/web-apps/current-work/#tabindex0

CAPÍTULO 8

SWFObject

SWFObject é um pequeno Script para a inserção de conteúdo Macromedia/Adobe Flash em páginas web. Ele detecta o plugin do Flash para a maioria das arquiteturas e plataformas, foi projetado para facilitar a inserção de mídias em flash. É amigável com mecanismos de busca, uma vez que, mostra um conteúdo alternativo quando a mídia não puder ser carregada. Pode ser usado tranquilamente com HTML e XHTML 1.1 (desde que seja text/html e não application/xhtml + xml).

Como Utilizar o Script?

Utilizar o script para inserir qualquer conteúdo Flash no site é muito fácil. Primeiro precisamos do código fonte do Swfobject[1], depois devemos incluí-lo em nosso código HTML. Abaixo segue exemplo com explicação de cada detalhe do processo.

Assim inserimos o script no site

<script type=”text/javascript” src=”swfobject.js”></script>

Elemento que receberá o conteúdo flash, já como texto alternativo, no caso do usuário possuir o flash player instalado, ele não verá esse conteúdo, porém será facilmente indexado por qualquer mecanismo de busca.

Esse texto será substituído por conteúdo Flash

E após isso, podemos criar um novo objeto da classe swfobject e fazer com que ele coloque o flash onde quisermos, conforme exemplo abaixo:

```
<script type="text/javascript">
  var so = new SWFObject("movie.swf", "mymovie", "200", "100",
"7",    #336699");
  so.write("flashcontent");
</script>
```

Cria um novo SWFObject passando os parâmetros obrigatórios:

- swf: o caminho e o nome do seu arquivo swf.
- id: o ID do seu objeto ou da tag “embed”. A tag embed também usará esse valor no seu atributo nome para os arquivos que utilizam o swliveconnect.
- width: a largura (em pixels) do seu filme Flash.
- height: a altura (em pixels) do seu filme Flash.

- version: a versão necessária para rodar o seu conteúdo em Flash. Pode ser uma string no formato “maiorVersão.menorVersão.revisão”. Por exemplo: “6.0.65”. Ou você pode apenas exigir a maior versão, como por exemplo “6”.

- background-color: o valor em hexa da cor de fundo do seu filme Flash.

Parâmetros opcionais:

- useExpressInstall: se você deseja atualizar o plug-in dos usuários usando o recurso ExpressInstall, use “true” para esse valor.

- quality: a qualidade com a qual você deseja que seu filme seja executado. O valor padrão é “high”.

- xiRedirectUrl: se você deseja redirecionar os usuários que completarem uma atualização via ExpressInstall, especifique a URL aqui.

- redirectUrl: se você deseja redirecionar os usuários que não têm a versão correta do plug-in, use esse parâmetro.

- detectKey: o nome da variável de URL que o script do SWFObject procurará para contornar a detecção. O padrão é “detectflash”. Por exemplo: para contornar a detecção do Flash e simplesmente escrever o filme Flash na página, você pode adicionar ?detectflash=false na URL do seu documento que contém o filme Flash.

so.write("flashcontent");

Diz ao script do SWFObject para escrever o conteúdo em Flash na página (se a versão correta do plug-in estiver instalada no sistema do usuário), substituindo o conteúdo dentro do elemento HTML especificado.

[1] - http://blog.deconcept.com/swfobject/swfobject_source.js

Adicionando Parâmetros para o Flash

O exemplo supracitado se ateve somente ao básico na inserção de um arquivo swf dentro de uma página HTML. Porém como todos sabem, existem parâmetros adicionais que podemos adicionar aos arquivos Flash. Abaixo segue um exemplo com alguns parâmetros adicionas, mas pode ser obtida no site da Adobe a lista completa de parâmetros[2].

```
<script type="text/javascript">
   var so = new SWFObject("movie.swf", "mymovie", "200", "100%",
"7", "#336699");
   so.addParam("quality", "low");
   so.addParam("wmode", "transparent");
   so.addParam("salign", "t");
   so.write("flashcontent");
</script>
```

[2] - http://www.adobe.com/go/tn_12701

Passar Valores para o Flash utilizando FlashVars

A utlização de FlashVars é a maneira mais comum e fácil de passarmos valores como parâmetros para o arquivo swf dentro de um site HTML, os valores são passados uma vez, quando o arquivo é carregado. O modo normal de inserir esse conteúdo é passar um parâmetro chamado FlashVars, e seu valor um lista de pares nome=valor, como exemplo abaixo:

variavel1=valor1&variavel2=valor2&variavel3=valor3...

O SWFObject torna isso mais prático e objetivo, permitindo que você adicione tantas variáveis quanto necessárias, de uma maneira similar à aquele usada para incluir parâmetros adicionais ao arquivo. Abaixo temos um exemplo da utilização:

```
<script type="text/javascript">
  var so = new SWFObject("movie.swf", "mymovie", "200", "100",
"7", "#336699");
  so.addVariable("variavel1", "valor1");
  so.addVariable("variavel2", "valor2");
  so.addVariable("variavel3", "valor3");
  so.write("flashcontent");
</script>
```

Desse modo, ao ser carregado, imediatamente seu arquivo contará com todos esses valores disponíveis na linha de tempo _root.

Outro recurso interessante do SWFObject é a função que permite extrair variáveis diretamente da URL do site, se tivéssemos a seguinte URL:

http://www.example.com/page.html?variavel1=valor1&variavel2=valor2

Utilizando a função getQueryParamValue(), você pode facilmente pegar esses valores da URL e passá-los para o seu arquivo swf. Abaixo exemplo para a supracitada URL:

```
<script type="text/javascript">
  var so = new SWFObject("movie.swf", "mymovie", "200", "100",
"7", "#336699");
  so.addVariable("variavel1", getQueryParamValue("variavel1"));
  so.addVariable("variavel2", getQueryParamValue("variavel2"));
  so.write("flashcontent");
</script>
```

EXPRESS INSTALL

Um recurso muito interessante do SWFObject é seu completo suporte ao Recurso Adobe Flash Player Express Install. Junto com o script do SWFObject existe um arquivo ActionScript que funciona com o SWFObject, para dar inicio ao processo de atualização do Plugin do Flash.

Para utilizar tal recurso, primeiro você deve incluir o arquivo expressinstall.as no seu .fla e no, inicio do arquivo, fazer uma pequena checagem para ver se o plugin do usuário precisa ser atualizado:

```
#include "expressinstall.as"

var ExpressInstall = new ExpressInstall();

// se o usuario precisa ser atualizado, mostra o botao “iniciar
atualizacao”
if (ExpressInstall.needsUpdate) {

// isto eh opcional, vc pode tambem iniciar a atualizacao
automaticamente,
// chamando ExpressInstall.init() aqui, ao inves das seguintes
linhas

// anexa a msg personalizada de atualizacao, centralizada
var upgradeMsg = attachMovie("upgradeMsg_src", "upgradeMsg", 1);
upgradeMsg._x = Stage.width / 2;
upgradeMsg._y = Stage.height / 2;

// anexa as acoes de botao q iniciarao o atualizador ExpresInstall
upgradeMsg.upgradeBtn.onRelease = function() {
// este metodo eh o q dah inicio a atualizacao
ExpressInstall.init();
}
// se o expressinstall foi invocado, para a linha do tempo
stop();
}
```

É importante ressaltar que não se deve colocar nenhum conteúdo no primeiro quadro ou no que acontecer a checagem do expressinstall.

DOWNLOAD

No blog do autor[3] é possível baixar o conteúdo completo, incluindo o código fonte do JavaScript, arquivos .fla, arquivos .swf, expressinstaller.as e exemplos da implementação do SWFObject em diferentes circunstâncias.

[3] - http://blog.deconcept.com/swfobject/swfobject1-4.zip

UM EXEMPLO DE COMO APLICAR

Entre as tags <head></head> colocar os seguintes códigos, linkando para dois arquivos JavaScript, o primeiro para um arquivo novo onde iremos escrever uma função:

```
<script type="text/javascript" src="js/flash.js">

</script>

<script type="text/javascript" src="js/swfobject.js">

</script>
```

O arquivo swfobject, é sempre o mesmo para todos os projetos. (APENAS COPIAR E COLAR).

O arquivo flash.js, vai conter os códigos que substituirão o id declaro em HTML pelo flash, neste arquivo declaramos também a altura e a largura do flash.

Exemplo de um arquivo flash.js:

```
// JavaScript Document

function flash(arquivo,id,largura,altura){

    if(!document.getElementById(id))

        return;

    var so=new SWFObject(arquivo,id,largura,altura,8,"#FFFFFF");

    // 8 indica a versão do flash;

    // #FFFFFF indica que a cor do fundo padrão branco;

    so.addParam("wmode", "transparent");

    so.write(id);

}

window.onload = function(){

    flash("flash/nome_flash.swf","nome_id",largura,altura);

    flash("flash/banner1.swf","banner_atualizacoes",213,80);

    flash("flash/banner2.swf","banner_aprenda",213,80);

    flash("flash/video.swf","video_flash",230,188);

}
```

CAPÍTULO 9

Acessibilidade no Flash

O Flash já foi um alvo de muitas críticas, incluindo do famoso Jakob Nielsen, pai da usabilidade, sobre suas deficiências ao gerar conteúdos acessíveis.

Bom a Macromedia, atualmente Adobe, contratou o próprio Jakob Nielsen, autor das maiores críticas, como consultor para melhorar o programa. O resultado foi o surgimento de um painel novo, além de novas características em outros painéis.

Vamos ver como utilizar estes painéis da forma correta, aplicando ao documento, menus e campos de texto.

Para este manual básico, estarei utilizando a versão CS3 do Flash, mas as mesmas ferramentas e painéis são encontradas nas versões MX 2004 e 8.

Acessibilizando o documento

Janela Document Properties

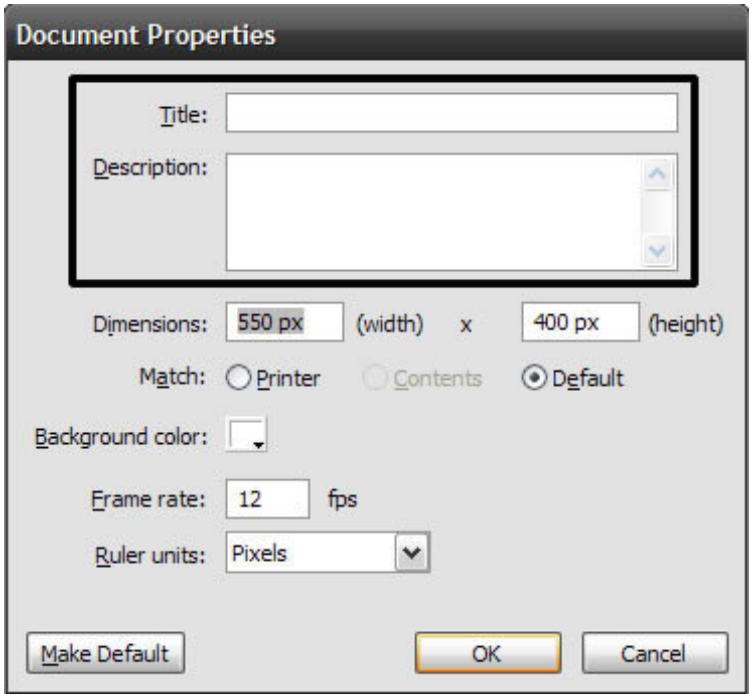

Figura 3: Layout da Janela Document Properties

Para abrir esta janela, vá até o menu Modify > Document.

Para cada arquivo em Flash que for desenvolver é importante que defina duas informações sobre o mesmo:

Title – Este campo serve para determinar um título para o arquivo, uma informação breve, sobre o documento, como por exemplo: “Catálogo de Produtos”.

Description – Neste campo faça uma breve descrição sobre o conteúdo do arquivo e seu propósito. Como por exemplo: “Conheça nossas linhas de produtos”.

Com o documento descrito podemos começar a desenvolver nosso projeto, sabendo que já informará ao usuário que documento está acessando e qual o seu propósito.

Painel Acessibility

Com este painel podemos gerar conteúdo acessível, fornecendo equivalentes textuais aos elementos utilizados no Flash.

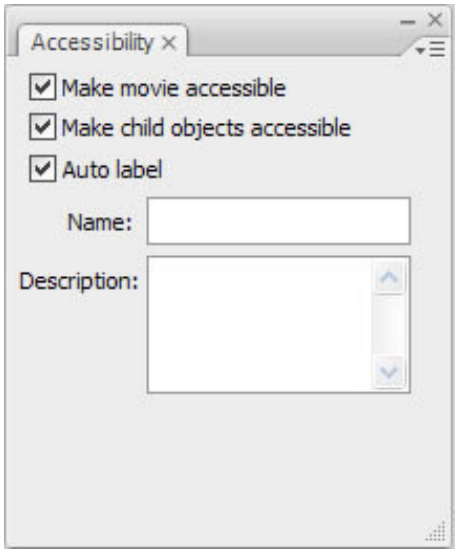

Figura 4: Layout da janela Painel Acessibility do Flash (apresenta os três itens ativados)

Para abrir este painel, vá até o menu Window > Other Panels > Acessibility.

Acessibilizando Textos

Por padrão, o Flash permite o acesso a todos os elementos textuais para os leitores de tela. Não sendo necessário fazer nenhuma modificação. Mas é importante observar que as opções do painel Accessibility mudam de acordo com o tipo de Campo Texto que for utilizado.

Static Text (Texto Estático)

Se escrever qualquer texto utilizando o campo Static Text, vai encontrar no painel Accessibility apenas a mensagem: “Current selection cannot have accessibility applied to it”, ou seja: “A seleção atual não tem acessibilidade aplicada a ela”, o que nos informa que não há opções a serem configuradas para ela, *veja a Figura 5*. Mas é importante deixar ativa a opção “selectable”, que possibilita a seleção do texto, isto permite que usuário selecione o texto e facilita a acessibilidade para alguns leitores de tela.

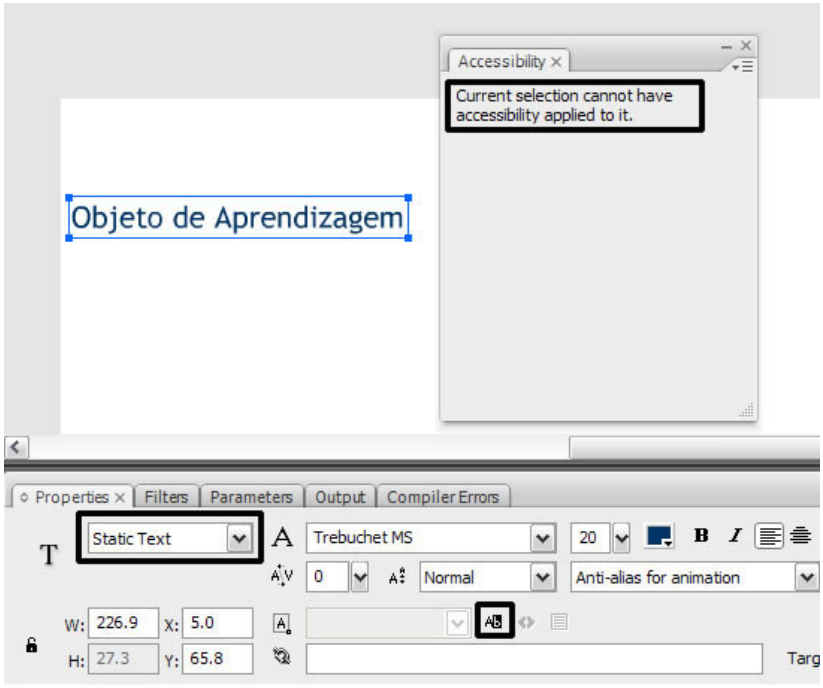

Figura 5: Static Text (Texto Estático) não possui configurações no painel Accessibility

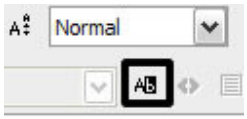

Figura 6: Detalhamento da opção “selectable”

Dynamic Text (Texto Dinâmico)

Os campos de texto do tipo Dynamic Text, quando com a opção “Make object Accessible”, ou seja, “Torne o objeto acessível”, estiver ativada, permitirá ao leitor de tela ler exatamente o conteúdo escrito no mesmo. Mas se achar necessário poderá contar com mais duas configurações:

Description (Descrição) – Utilize esta opção para definir uma pequena descrição do campo, o qual deve ser feito apenas se o campo necessitar de maiores explicações, o que na maioria das vezes não é necessário.

Tab index (Ordem da Tabulação) – Utilizado para definir a ordem que o objeto será selecionado quando o usuário navegar utilizando as teclas TAB e SHIFT+TAB.

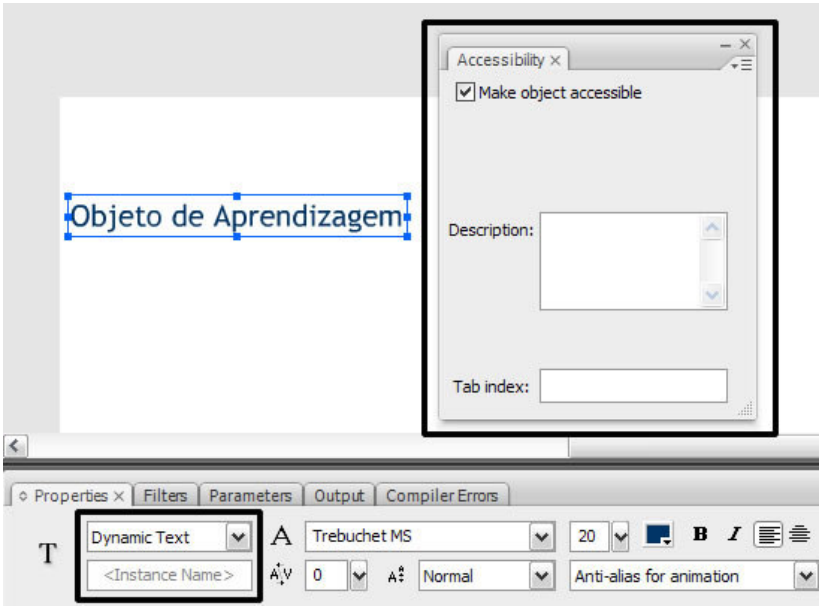

Figura 7: Opções de acessibilidade para Dynamic Text

Input Text (Texto de Entrada)

No campo de texto do tipo Input Text, configuramos o painel Accessibility da mesma forma que fariamos na linguagem (X) HTML, onde é indicado definir um “Name” – um rótulo para o campo de texto, para que o usuário saiba o que está preenchendo, e quando for necessário coloque uma descrição no campo “Description”, como por exemplo: “Campo obrigatório”.

Além destas duas opções ainda temos os campos:

Shortcut – que serve para definir um atalho, um acesso rápido ao campo, o que é feito definindo uma letra ou número, como por exemplo: “A”. O atalho para o usuário será a tela ALT juntamente com a letra ou número definido, no nosso exemplo o atalho é “ALT + A”.

Tab index – No caso de formulários é muito importante definir a ordem da tabulação para que o usuário tenha uma ordem lógica definida ao preencher os campos. Caso contrário ele ficará preso à ordem de criação dos campos.

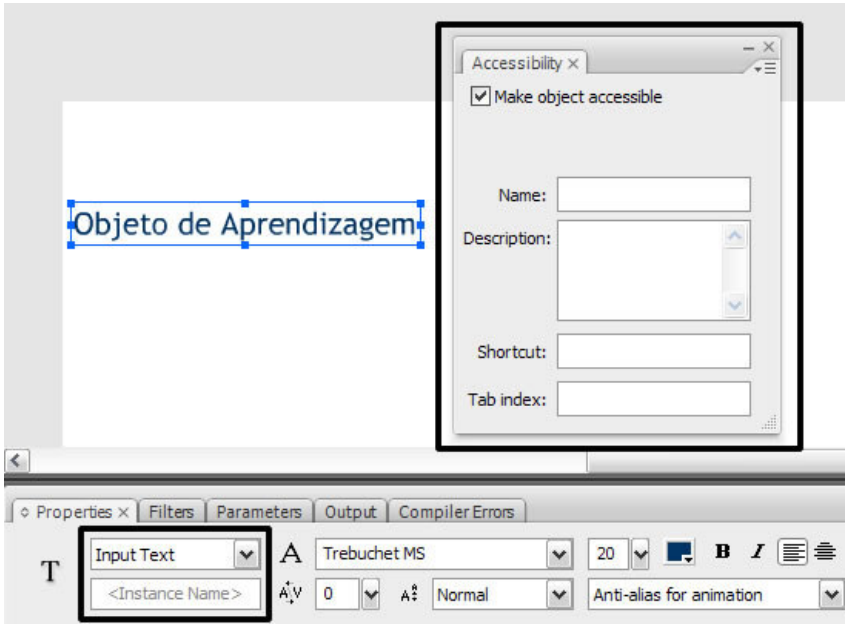

Figura 8: Opções de acessibilidade para Input Text (Texto de Entrada)

Acessibilizando Buttons e Movie Clips

O termo “etiquetar” é bem conhecido por quem desenvolve conteúdo acessível, que significa identificar textualmente um elemento, o que é feito no HTML com imagens através do atributo “alt” e no Flash é feito através do painel Accessibility.

Tanto os Buttons como os MovieClips tem que ser etiquetados, pois mesmo contendo campos de texto inseridos no seu corpo, é um elemento fechado e pode acarretar em não ser acessado pelo leitor de tela.

Para demonstrar como etiquetar um menu de navegação, criei no Flash dois elementos utilizados para este fim, um Button, com a função de voltar e um MovieClip com a função de avançar.

A utilização do Symbol Button é para botões mais simples, sem uma animação mais elaborada, enquanto o Symbol Movie Clip, deve ser utilizado para botões animados e com aspectos visuais mais elaborados.

Acessibilizando um Button

O Button tem as mesmas opções que foram vistas nos campos de texto, então basta configura-lo da mesma forma, mas é importante salientar a importância de determinar o rótulo do botão no campo “Name” do painel Accessibility, para este botão do exemplo o Name seria “Voltar” e nos casos que forem necessários determine a descrição no campo Description, aqui poderíamos utilizar: “Clique para voltar para o slide anterior”.

Para botões com a função de navegação de slides, como “Voltar” e “Avançar” é importante definir corretamente o Tab index e um Shortcut, como exemplificado na imagem abaixo:

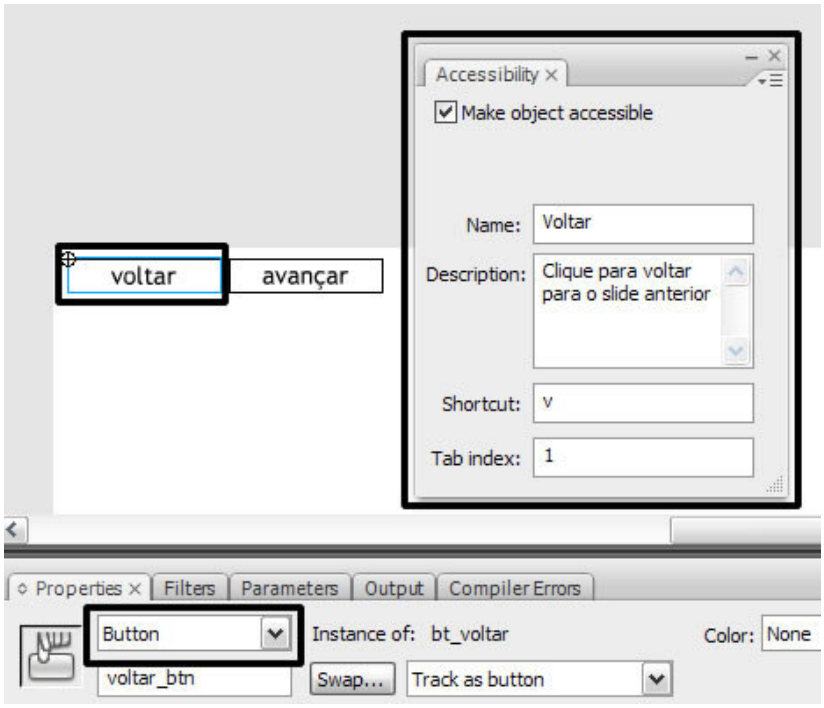

Figura 9: Opções para acessibilizar o Button

Para botões animados desenvolvidos com Movie Clips, tem apenas uma diferença no painel Accessibility para os botões definidos pelo Symbol Button, a opção “Make child objects accessible” ou seja, “Tornar os objetos filhos (ou internos) acessíveis”.

Acessibilizando um Movie Clip

Neste caso como o Movie Clip pode desde ter a simples funcionalidade de um botão até ter a função de recipiente de partes ou de toda uma animação, a opção “Make child objects accessible”, permite que os elementos internos do MovieClip possam ser lidos pelo leitor de tela, os quais devem também estar com as suas opções de acessibilidade definidas.

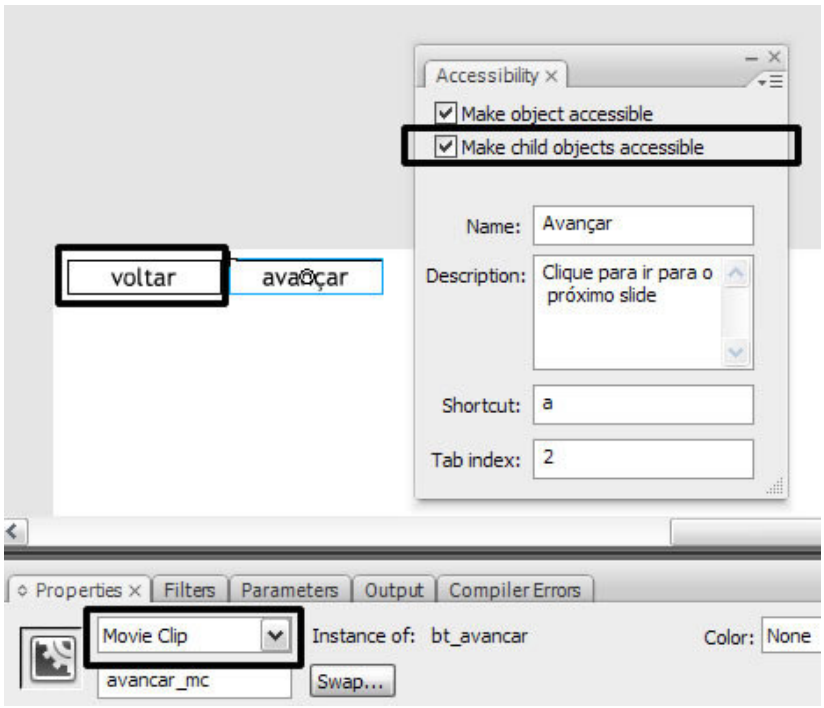

Figura 10: Opções para acessibilizar Movie Clip

REFERÊNCIAS BIBLIOGRÁFICAS

SONZA, Andréa Poletto. Ambientes Virtuais Acessíveis sob a perspectiva de usuários com limitação visual. Tese (Doutorado) – Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Informática na Educação, Porto Alegre, 2008.

SONZA, Andréa Poletto. Acessibilidade de Deficientes Visuais aos Ambientes Digitais Virtuais. Dissertação (Mestrado). Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Educação, Porto Alegre, 2004. 197f.

W3C, 2007. Disponível em <http://www.w3.org/TR/xhtml1> Acesso em nov 2007.

W3SCHOOLS, 2007. Disponível em <http://www.s3schools.com> Acesso em set 2007.

DECONCEPT, 2008. Disponível em <http://blog.deconcept.com/swfobject> Acesso em mar 2008.

SOUWEB, 2008. Disponível em <http://www.souweb.info> Acesso em mar 2008.

BLOGDOXORNA, 2008. Disponível em <http://www.blogdoxorna.com> Acesso em mar 2008.

JUICYSTUDIO, 2008. Disponível em <http://juicystudio.com/article/making-ajax-work-with-screen-readers.php> Acesso em mar 2008.

WIKIPEDIA, 2007[1]. Tableless. Disponível em <http://pt.wikipedia.org/wiki/Tableless> Acesso em nov 2007.

WIKIPEDIA, 2007[2]. Web Semântica. Disponível em <http://pt.wikipedia.org/wiki/Web_sem%C3%A2ntica> Acesso em out 2007.

ZELDMMAN, Jeffrey. Projetando Web Sites Compatíveis. Editora Campus. Rio de Janeiro: Elsevier, 2003.

CAMPOS, Leandro. HTML Rápido e Prático. Editora Terra. Goiânia : Terra, 2004