WEB STANDARDS E PADRÕES

de usabilidade e de acessibilidade de aplicações web

Senac

WEB STANDARDS E PADRÕES DE USABILIDADE E DE ACESSIBILIDADE DE APLICAÇÕES *WEB*

Nesta unidade curricular (UC) tratamos essencialmente de elementos visuais presentes em aplicações *web*. Entretanto, precisamos entender que um bom *front-end* não deve se focar exclusivamente no visual da página ou na funcionalidade do sistema. É necessário, desde o princípio, pensar na experiência do usuário e na facilidade de uso da aplicação.

A seguir, veremos orientações e aplicações práticas sobre *web standards*, usabilidade e acessibilidade.

Web standards: padrões sugeridos pela W3C e recomendações de boas práticas

Web standards são recomendações publicadas pela World Wide Web Consortium (W3C), a principal organização de padronização da *web*. Elas se referem especificamente a sistemas para a internet.

VOCÊ PODE CONSULTAR O CONTEÚDO *WEBSTANDARDS*: **PRÁTICAS E PADRÕES RECOMENDADOS PELO W3C**, DA UC **CODIFICAR APLICAÇÕES** *WEB*, PARA OBTER UM PANORAMA TEÓRICO MAIS COMPLETO SOBRE OS PADRÕES DEFINIDOS PELA W3C.

Uma recomendação da W3C é constituída por um conjunto de diretrizes elaboradas e avaliadas pelos membros da organização, entre empresas e organizações governamentais e independentes. Entre os padrões sugeridos, estão o próprio HTML (em suas diversas versões) e o CSS, além de guias para uso de JavaScript.

Os padrões determinam, entre outras coisas, que o HTML seja usado especificamente para marcação de conteúdo, sem especificação de estilo, e que o CSS seja responsável pelo visual da página. Por exemplo, a *tag* <h1> deve ser usada quando o conteúdo envolvido for um título, e não apenas para deixar uma fonte com tamanho maior.

Há vantagens na utilização de padrões e de boas práticas em HTML e CSS. Veja:

Maior visibilidade nas buscas

Um código HTML estruturado de maneira correta é mais bem lido pelos buscadores *web*, como o Google, permitindo a interpretação e a indexação correta do *website*.

Velocidade

Os *sites* ficam mais eficientes. Um *site* malformado, ou com códigos adicionais inúteis, aumenta a banda de dados e o processamento no navegador.

Manutenção

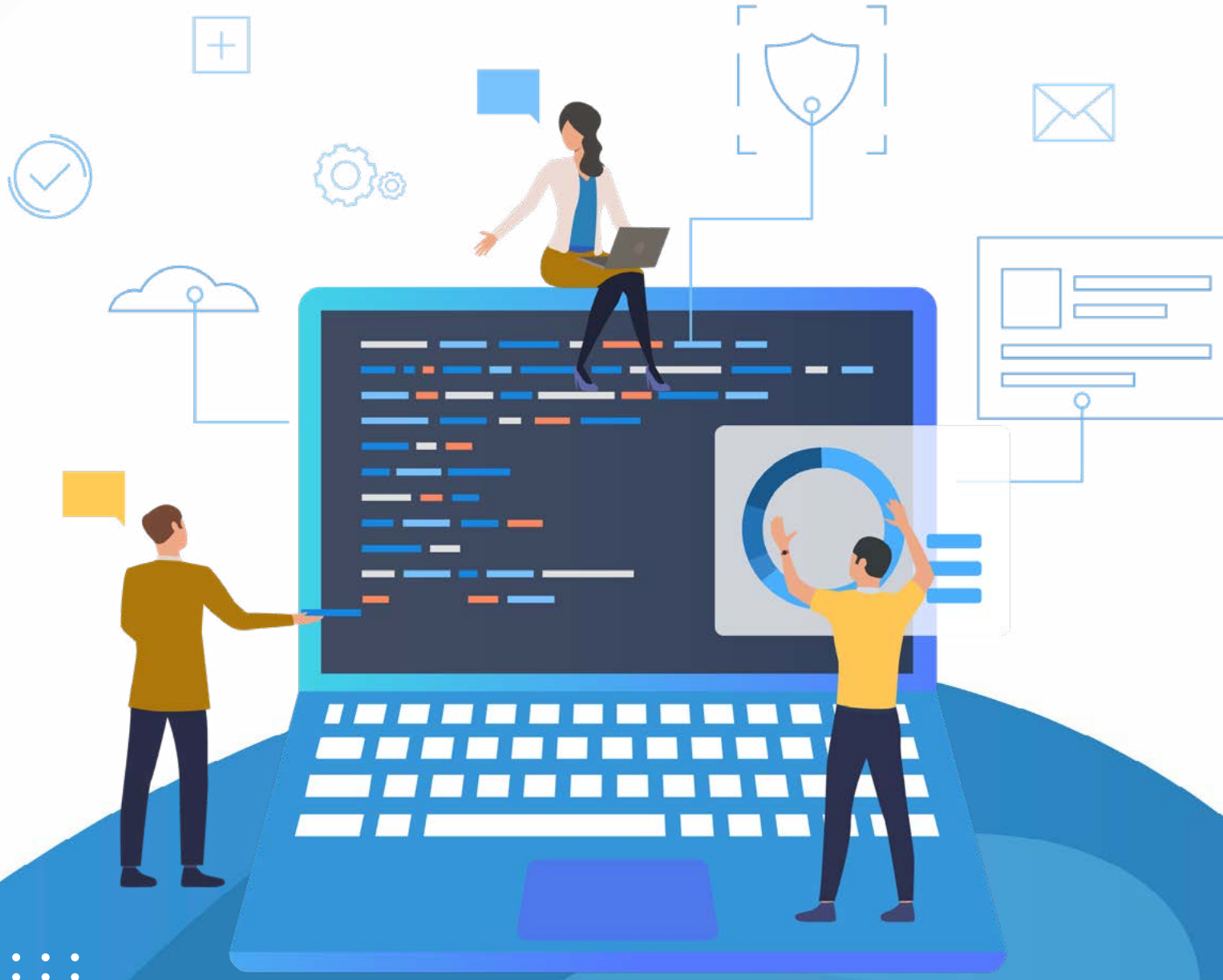
Fica mais fácil redefinir o *design* de uma página se todo o estilo estiver especificado em folhas CSS separadas. Além disso, um código HTML mais legível é essencial para que os demais envolvidos no projeto consigam compreender a página e realizar a sua manutenção.

Algumas boas práticas sobre estrutura HTML

- Separar o estilo CSS em arquivo separado em vez de usar a *tag* <style> permite o reúso de estilos predefinidos e a divisão de responsabilidades na aplicação. Os atributos *style* dos elementos também devem ser evitados.
- A *tag* <table> deve ser usada apenas para tabular dados, e não para definir posicionamento de elementos. Para isto, usam-se **float**, **margin**, **padding**, **position** e **display** no CSS.
- Use a *tag* <p> para parágrafos em vez de texto direto com quebra por
 ou parágrafos formados por .
- Use <header>, <section>, <main> e <nav> em vez de *tags* genéricas, como já recomendado ao longo do curso.
- Use <audio> e <video> em vez de <object>.
- Use <ul> para navegação (lista de *links*) e aplique estilo CSS para adaptar posição e aparência.
- Sempre encerre as *tags*. Preste muita atenção à ordem em que as *tags* alinhadas são iniciadas e encerradas. Veja o exemplo:

```
<div id=”test”>
    <div><a href=”#” title=”test”><img src=”ex.jpg” alt=”exemplo” /></a></
div> <!-- certo -->
    <div><a href=”#” title=”test”><img src=”ex.jpg” alt=”exemplo” /></
div></a> <!-- errado, div iniciou antes de a -->
</div>
```

- *Tags* que se autoencerram terminam com **/>**. Use
, por exemplo, em vez de
.
- É preciso ter cuidado com a formatação e a indentação do código. *Tags*-filhas ficam um nível à frente de *tags*-pais. O Visual Studio Code provê a ferramenta Format Document (por padrão, com atalho **Shift + Alt + F**), que reformata o código e o deixa mais legível.
- Avalie o código usando ferramentas de validação, como o W3C Markup Validation Service. Busque esse termo na *web* para acessar o validador, que é gratuito e simples de usar.
- Utilize a *tag* <meta charset="UTF-8"> em <head> quando houver acentuação nos textos da página.



Algumas boas práticas sobre estilo

Defina estilos genéricos e estilos específicos para cada tamanho de tela. O CSS define a diretiva **@media**, que permite aplicar um determinado estilo a uma condição. Veja o exemplo:

```
@media (min-width:320px) {
        body {
            background: none;
        }
        main {
            width: 300px;
            font-size: 14px;
        }
    }

    @media (min-width:480px) {
        main{
            width: 460px;
            font-size: 18px;
        }
    }
```

Usamos **@media** para definir que telas com largura a partir de 320 px tenham a *tag* <main> com largura de 300 *pixels* e fonte de 14 *pixels*. Já se a tela tiver a partir de 480 *pixels*, a largura muda para 460 *pixels;* e a fonte, para 18 *pixels*.

Essa abordagem faz parte de uma filosofia "*mobile-first design*" (projeto que considera antes as telas *mobile*) e é um dos fundamentos da responsividade.

- Utilize unidades relativas de tamanho. Em vez de *pixels*, cm ou pt, podemos optar por %, em, vw (1/100 da largura da tela) e vh (1/100 da altura da tela). Isso também impacta a responsividade, uma vez que a página poderá ser visualizada em variadas resoluções.

Já do lado do Javascript, há padronização, mas ela não vem pelo W3C, e, sim, pela ECMA (European Computer Manufacturers Association, ou Associação Europeia de Fabricantes de Computadores). Isso tanto é verdade que o nome "oficial" do JavaScript é ECMAScript.

O padrão define, basicamente, a sintaxe e as funcionalidades da linguagem. Ele é atualizado regularmente (em 2019, a versão é a 8, mas a utilizada nos *browsers* é a 6). Um ponto de atenção é quanto à versão do navegador – funcionalidades podem não estar disponíveis em alguns navegadores mais antigos. No entanto, hoje, com as atualizações constantes nesses *softwares*, esse problema deve ser minimizado.

Algumas boas práticas em JavaScript

- Inclua *script* no fim da *tag* <body>. É comum encontrar a inclusão de *scripts* JS na *tag* <head> da página. No entanto, o carregamento do *script* pode travar a renderização do resto da página. Assim, é preferível que a referência ao *script* fique no fim da página – só carregará depois de todos os elementos dela.

A tipagem no JavaScript é dinâmica, então é necessário ter cuidado ao informar um valor de tipo diferente do original.

Prefira usar a comparação === (que verifica se dois valores são idênticos) a uma com o operador == (que analisa se os valores são equivalentes). Veja um exemplo:

```
0 == “”; //true - 0 é considerado “vazio” ou “false”
0 === “”; //false
1 == “1”; //true - 1 é considerado equivalente ao texto “1”
1 === “1”; //false
```

Reduza o uso de variáveis globais: elas ficam disponíveis para todos os outros *scripts*.

```
var nome = ‘abc’;

function doSomething() {
  console.log(nome); //abc
}
```

Caso declare um vetor já preenchido, use a notação **[]**. Veja o exemplo:

```
var vetor = [‘nome1’, ‘nome2’];
//equivale a
var vetor = new Array();
vetor[0] - “nome1”;
vetor[1] - “nome2”;
```

- Evite a função **eval()**: ela interpreta e executa um texto de *script*, mas são raros os casos em que será necessária.
- Utilize ferramentas de validação como o JSLint para verificar a escrita do código JavaScript.

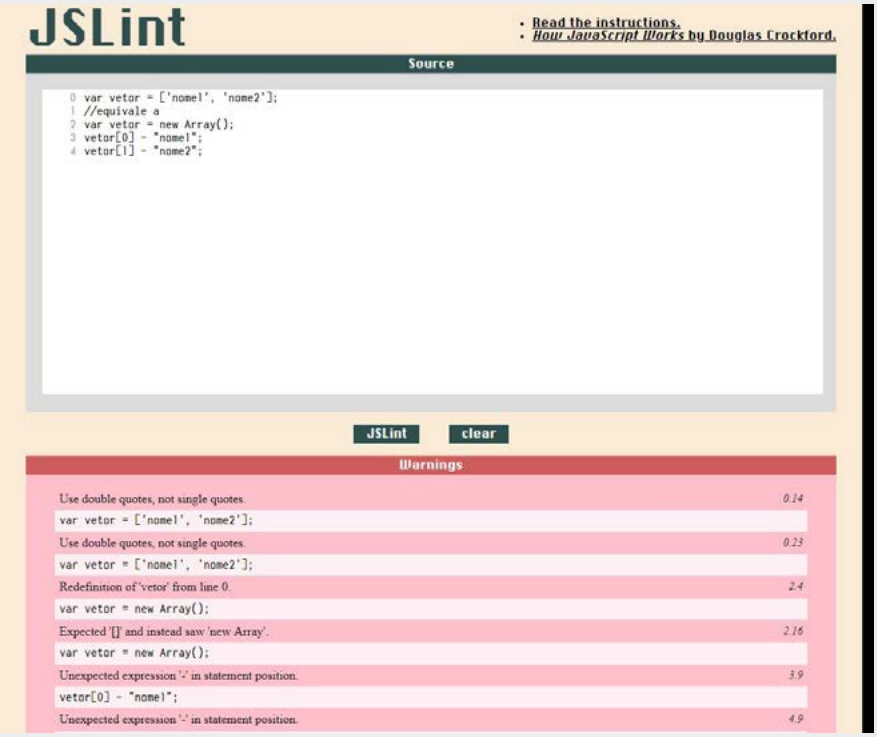


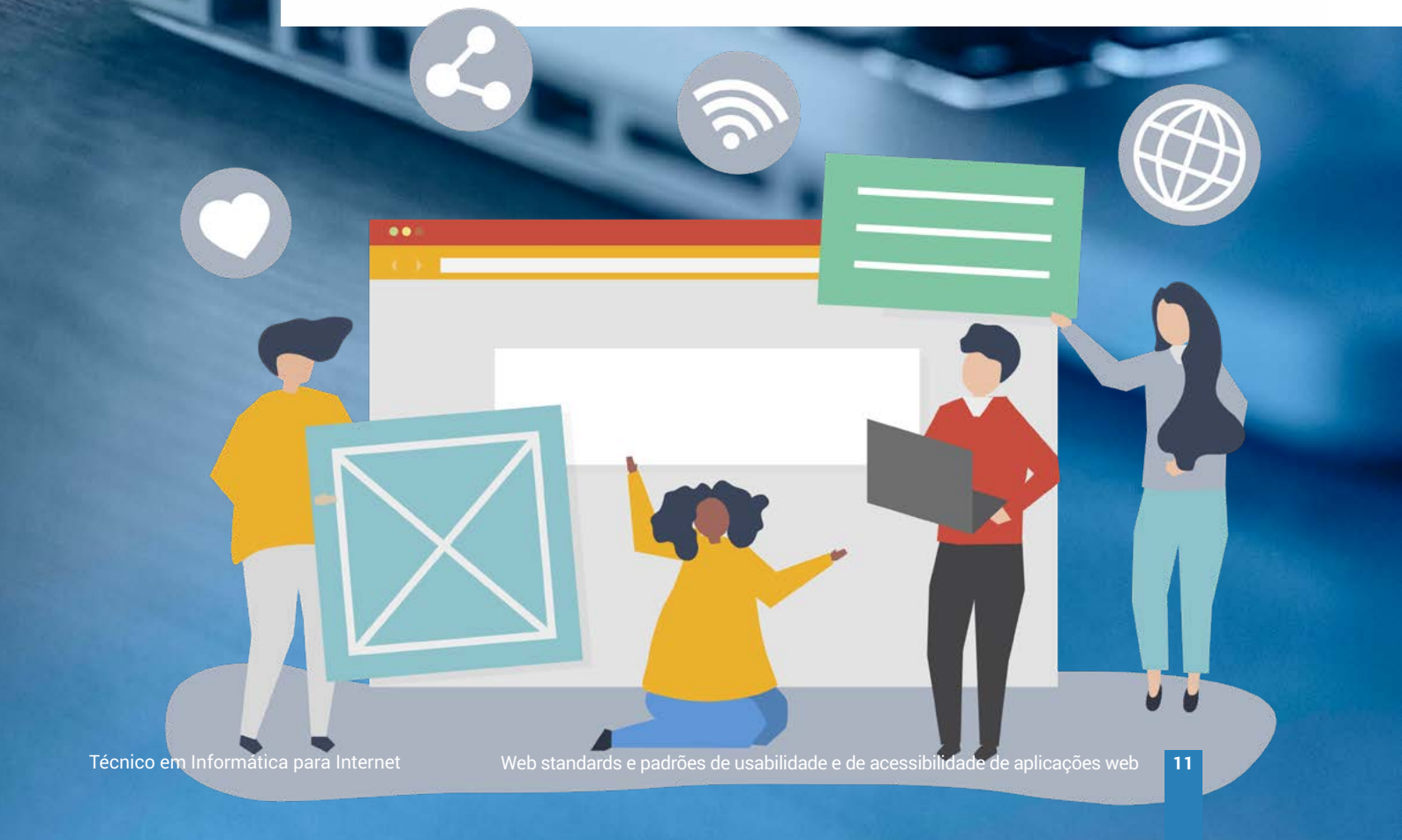
Figura 1 – Página do JSLint mostrando o código validado e o resultado da validação

ACESSIBILIDADE

Acessibilidade é a característica, desejável a um *website*, que permite que pessoas com alguma deficiência possam usá-lo, podendo entender, perceber, navegar e interagir, além de contribuir para a *web*. Ela envolve questões bastante ligadas ao *front-end* da aplicação e que afetam tanto a construção do conteúdo com HTML quanto a sua estilização.

Há algumas situações-chave que servem como exemplo e guia para a implementação de sistemas acessíveis. É preciso considerar que o usuário pode, por exemplo:

- **Ser deficiente visual (pessoa cega ou com baixa visão)**
- **Ser deficiente auditivo (com perda parcial ou total da audição)**
- **Apresentar deficiência cognitiva ou distúrbio de aprendizagem (dislexia, disgrafia, discalculia, dislalia, disortografia, TDHA)**
- **Ter algum tipo de deficiência motora, permanente ou temporária (braço quebrado), situação que impediria a utilização do mouse ou do teclado, por exemplo**
- **Ter dificuldade para distinguir cores (daltonismo)**



A partir de considerações como essa, é fácil perceber que uma variedade de situações deve ser coberta. Para facilitar o desenvolvimento e a verificação de acessibilidade em *sites*, o W3C publicou o *WCAG* (*Web Content Acccessibility Guidelines*, ou diretrizes de acessibilidade para conteúdo da *web*), que em 2018 entrou na versão 2.1.

O documento traz uma série de recomendações para acessibilidade para a *web*. Ele pode ser resumido em 13 recomendações principais, agrupadas em quatro princípios. Veja:

1 Componentes e informações de interface devem ser percebidos por todos (devem ser perceptíveis).

a) Informações não textuais devem trazer um texto alternativo, de modo que seja possível a exibição em um formato ou outro.

No HTML, usa-se o atributo **alt** para imagens, vídeos ou outros objetos, para definir um texto a ser apresentado quando a imagem, por algum motivo, não for carregada. Também poderá ser usado por um leitor de tela para descrever a imagem (em *softwares* de leitura para pessoas cegas).

```
<img src="logo_empresa_xyz.png" alt="Empresa XYZ" />
```

b) Mídias baseadas em tempo (vídeo, áudio) devem ter alternativas.

Vídeos incluídos na página devem trazer opção de legenda. No HTML5, a *tag* <track> pode ser usada com <vídeo> para carregar uma legenda. Isso ajudará pessoas com dificuldade de audição.

```
<video width="320" height="240" controls>
    <source src="video.mp4" type="video/mp4">
    <track src="legenda_en.vtt" kind="subtitles" srclang="en" label="English">
</video>
```

c) O conteúdo da página deve ser apresentado de diferentes maneiras sem a perda de informação ou estrutura.

Usar duas configurações de leiaute, por exemplo, pode ajudar alguém com dificuldades visuais. Para isso, criam-se duas folhas CSS (uma normal e outra de alto contraste, por exemplo), e dinamicamente o *site* poderá trocar por uma ou outra.

Recomenda-se ainda o uso do atributo **title** para elementos sem descrição alternativa, agrupamento de campo por meio de <fieldset> e agrupamento de *links* com <ul> ou <ol>. Esses são recursos que facilitam o trabalho de *softwares* leitores de páginas.

d) Conteúdos visuais e auditivos devem ser fáceis de distinguir.

Cabe aqui também o exemplo anterior de uso de um estilo alternativo de alto contraste para a página. Também é importante que cores não sejam o único meio de comunicar a diferença entre elementos na tela, pois isso afeta diretamente pessoas com dificuldade de visualização de cores. Além da cor, é importante alguma indicação, como um texto (um botão verde para confirmar e um vermelho para refutar ficam melhores com os textos “ok” e “cancelar”, por exemplo).

2 A interface de usuário deve ser operável e com navegação utilizável.

a) Todas as funcionalidades devem ser acessíveis via teclado.

A navegação no *site* deve poder ser realizada por teclado. A tecla **tab**, por padrão, salta entre elementos de interação da página, como *links*, botões e caixas de texto.

Pode-se usar o atributo **tabindex** para definir a ordem de foco de cada elemento. Também é importante usar atributos **alt**, **title**, **value** ou elemento de texto em *tags* que recebem foco, pois os leitores de tela usam essa informação.

b) Usuários não devem ser forçados a realizar ações em um limite de tempo.

Estes cuidados se referem mais a comportamentos dinâmicos próprios da página ou ao tempo de validade de sessões.

c) Devem-se evitar estilos que provoquem tontura ou convulsão.

Evite cores piscantes ou elementos com comportamento dinâmico (que se movam com rapidez).

d) Guias e ajuda para navegação no site devem ser fornecidas ao usuário.

As *tags* de <title> e os títulos <h1> a <h6> (quando usados na ordem correta) ajudam o usuário a compreender a navegação do *site*. O já citado atributo **tabindex** também pode ser empregado para satisfazer esta regra de acessibilidade.

e) Facilite a operação do site usando modalidades de entradas de dados além do teclado.

Considere que eventos de *input* na página podem ocorrer tanto por *mouse* e teclado como por toque ou movimento. O uso de eventos de alto nível e independentes de entrada, como "focus", "blur" e "click", é recomendável.

3 Conteúdo e operação da interface de usuário devem ser inteligíveis (compreensível).

a) O conteúdo textual deve ser legível e compreensível.

Vários aspectos são levados em consideração nesta regra, como a linguagem do texto e as definições de termos específicos. O atributo **lang** pode ser usado para o primeiro caso, como neste exemplo:

```
<p>isto é português</p>
<p lang="en">This is English</p>
```

As definições podem ser contruídas com a *tag* <abbr> (abreviaturas) ou <dl> (lista de definição)

```
<dl>
    <dt>Metro</dt>
    <dd>Unidade que representa 100 cm</dd>
    <dt>Kilomêtro</dt>
    <dd>Unidade que representa 1000 m</dd>
</dl>
```

b) Os conteúdos devem aparecer e funcionar de maneira previsível.

Conceitos de usabilidade podem ser aplicados para que as ações padrão esperadas de fato aconteçam (espera-se que um botão <submit>, por exemplo, envie dados informados pelo usuário em um formulário).

c) Deve-se fornecer assistência ao usuário para que evite e corrija erros.

Mensagens de erro devem ser explícitas e textuais. O uso de *tag* <label> com seu atributo **for** corretamente preenchido (apontando para o campo ao qual se refere) pode ajudar o leitor de tela a identificar um erro (uma validação, por exemplo), o campo no qual ocorreu e descrição deste.

4 O conteúdo deve ser robusto, para poder ser usado de maneira confiável em qualquer tipo de agente de usuário, como tecnologia assistiva.

a) A compatibilidade entre os agentes de usuário atuais e futuros deve ser maximizada.

Uma regra básica para atender esta regra é escrever HTML válido, sintaticamente correto e semanticamente significativo. Podem-se usar validadores disponíveis na *web* para verificar essa robustez.

DICA

A INICIATIVA TODOS PELA ACESSIBILIDADE CRIOU UM *TOOLKIT* QUE PODE AJUDAR BASTANTE O DESENVOLVEDOR A ENTENDER, RELEMBRAR E APLICAR OS PRINCÍPIOS DA WCAG. TRATA-SE DE UM CONJUNTO DE CARTAS COM DEFINIÇÕES RESUMIDAS E DIRETAS DAS RECOMENDAÇÕES. BUSQUE NA INTERNET POR "ACESSIBILIDADE *TOOLKIT*" PARA OBTER MAIS INFORMAÇÕES.

USABILIDADE

De mãos dadas com a acessibilidade, mas constituindo um conceito diferente, está a usabilidade. A usabilidade é a qualidade de uma página ou produto que é utilizada(o) por um usuário médio de maneira que não cause frustração ou tédio.

Com a proliferação de *sites* e a popularização da *web*, o usuário que visita uma página não tem paciência para buscar uma informação por muito tempo ou utilizar uma interface que não seja intuitiva. Em vez disso, ele pulará para outra que ofereça uma experiência melhor.

Boa parte das questões de usabilidade convergem para a retenção do usuário (ele deve permanecer na página). Para isso, algumas características podem ser observadas no desenvolvimento de interface *web*.

- O aplicativo ou *site* deve ser autoexplicativo. As informações devem comunicar de maneira rápida e óbvia o propósito do *software* e o seu conteúdo.
- O usuário quer ir diretamente ao ponto, ao objetivo do *site*, sem ter que pensar nem se perder nas páginas. Além disso, se o *site* oferecer algo que realmente funciona, é provável que o usuário se torne fiel. Não é necessário que ele busque uma alternativa, embora, caso uma melhor lhe seja apresentada, ele possa usá-la.
- As pessoas não querem perder tempo. Então, se a página demorar para carregar, a paciência do usuário também tende a se esgotar rapidamente. Este é um item bastante técnico, pois envolve a escolha das imagens a serem usadas no *site* (as grandes implicam mais tempo de carga). *Scripts* (código JavaScript) não otimizados e páginas HTML malformadas também podem levar a uma demora.
- O usuário não pode ficar preso. Ele deve poder voltar (pelo navegador mesmo) às páginas anteriores. Além disso, um *link* para a página inicial do *site* deve estar sempre disponível.
- Buscas e mapas de *site* são importantes para a navegação do usuário.

VOCÊ PODE SE APROFUNDAR EM *USER EXPERIENCE* E USABILIDADE NO CONTEÚDO **CARACTERÍSTICAS FUNCIONAIS DO PROJETO DA APLICAÇÃO WEB**, DESTA UC.

De maneira geral, a usabilidade está muito mais ligada ao desenvolvimento de leiaute do que à programação de *front-end* em si. Veja algumas dicas que podem guiá-lo na construção de uma página ou na sua avaliação de usabilidade:

Menus de *links* posicionados em um local padrão no *site* (nas laterais, acima ou abaixo) e visíveis em todas as páginas facilitam a navegação do usuário. No ASP.NET Core MVC, podemos obter isso montando *partial views* e definindo menus padrão no leiaute.

A imagem mostra a imagem da página de um site

Figura 2 – <Uat.edu>: o leiaute é bonito, mas as animações atrapalham a usabilidade (os *links* dos cursos flutuam, e só é possível clicar neles quando estão à frente, não havendo outro menu com as mesmas informações)

- O uso pensado e informativo de cores, com contraste entre o fundo e os elementos de frente, diferenciando determinadas ações, faz o usuário compreender rapidamente o *site*. Definir no CSS cores diferentes para *links* e botões de submissão de dados, por exemplo, é uma maneira de informar que esses elementos realizam ações diferentes.

- É preciso usar cuidadosamente caracteres maiúsculos e minúsculos, espaçamentos e títulos. O bom uso das *tags* da família <h> novamente se faz necessário, uma vez que elas ajudam a dar importância a um tema e a um subtema da página de maneira imediata.

Figura 3 – Uso confuso de navegação, de fontes e de cores no *site* <Arngren.net>

- *Feedback* (resposta ao usuário após cada ação que ele toma) é muito importante. Um caso comum é a validação de dados informados em formulário. Deve ficar claro ao usuário que os dados estão incorretos. Para isso, podem-se usar JavaScript e CSS.

- Animações e vídeos devem guiar o usuário e nem sempre são bem-vindos. Veja, na figura 4, o exemplo de um conceito de animação para o cupom do PayPal. Ela é bonita, mas note que o valor final só é mostrado 3 segundos após o carregamento da página. Também não são bem-vindas animações em repetição que distraiam o usuário durante a navegação.

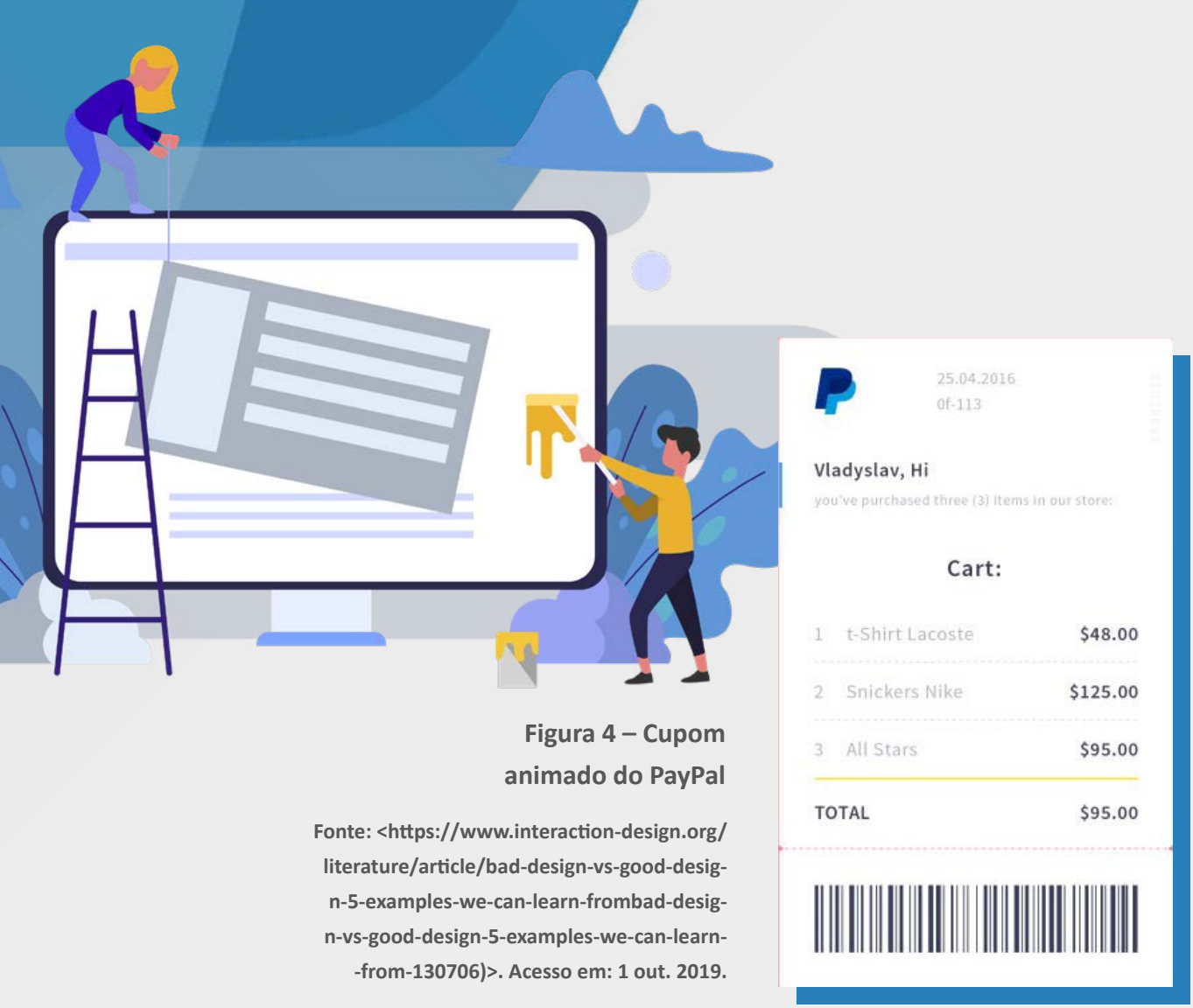


Figura 4 – Cupom animado do PayPal

Fonte: <https://www.interaction-design.org/literature/article/bad-design-vs-good-design-5-examples-we-can-learn-frombad-design-vs-good-design-5-examples-we-can-learn-from-130706)>. Acesso em: 1 out. 2019.

Em suma

A partir das recomendações da W3C e de outras entidades, pode-se garantir qualidade em diversos aspectos da aplicação, sejam técnicos, sejam de operacionalidade.

Ao se adotarem as regras de acessibilidade e de usabilidade, o foco deve ser sempre o cliente. Do que ele realmente precisa? E quais restrições as pessoas que usarão o *software* apresentam? As respostas a essas perguntas nortearão o esforço a ser aplicado ao projeto de *software*.

O certo é: quanto mais acessível for o *software*, melhor. Assim, mais pessoas poderão usá-lo e ter uma boa experiência.