Como a Web Semântica promove Acessibilidade na Web para Deficientes Visuais

Michel dos Santos Kuguio[1], Prof. Me. Geraldo Henrique Neto[1]

[1]Faculdade de Tecnologia de Ribeirão Preto (FATEC)
Ribeirão Preto, SP – Brasil

michel.kuguio@gmail.com, geraldo.henrique@fatec.sp.gov.br

Resumo. *A maioria das pessoas com deficiência visual tem dificuldade em acessar um site na web. Isto acontece pelo fato do mesmo não está preparado para essas pessoas, uma maneira de prover esse acesso é aumentando a web semântica de um site, neste artigo será demonstrado formas de melhorar a web semântica, como essas mudanças são vista pelos navegadores e como isso pode beneficiar pessoas com deficiência visual no que tange uma navegabilidade satisfatória.*

Abstract. *Most people with visual impairments have difficulty accessing a website. This happens because it is not prepared for these people, one way to provide this access is to increase the semantic web of a website, in this article we will demonstrate ways to improve the semantic web, how these changes are seen by browsers and how it can benefit visually impaired people in terms of satisfactory navigability.*

1. Introdução

Em 2012 cerca de 45,5 milhões de pessoas apontam ter uma ou mais tipo de deficiência, o que corresponde 23,9% da população brasileira no mesmo ano, além disso a deficiência visual foi a mais apontada atingindo 18,8% da população, uma quantidade muito significativa e que por serem deixado de lado perde o privilégio de terem boas experiencia com a maioria das aplicações e consequentemente é um público que o mercado acaba perdendo por não prover as acessibilidades básicas para essa parte da população. (Leal & Thomé, 2012)

Para garantir uma acessibilidade na web é necessário criar aplicações com o código rico semanticamente, proporcionando assim mais informações para que programas para deficientes visuais possam transmitir melhor o conteúdo apresentado na tela.

O presente trabalho tem a proposta de apresentar recursos para melhorar semântica de aplicações web e avaliar o impacto de tais melhoras para os deficientes visuais.

2. Tecnologias

Neste capítulo, será apresentado as seguintes tecnologias HTML5 (HyperText Markup Language), Dados Estruturado e WAI-ARIA (*Web Accessibility Initiative - Accessible Rich Internet Applications*). Mostraremos como utilizar elas e como o sistema vê uma aplicação web que faz uso de tais tecnologias.

2.1. HTML5

O HTML (Linguagem de Marcação de HiperTexto) é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web. (Mozilla, 2019) Desenvolvido para ser processado por navegadores e apesar de ser o elemento base de uma página web é possível fazer uma página visualmente normal porem com uma péssima semântica.

```
<!DOCTYPE html>
<html lang=”pt-BR”>
  <head>
    <meta charset=”UTF-8” />
    <title>Título</title>
  </head>
  <body>
    conteúdo da aplicação
  </body>
</html>
```

Figura 1. Estrutura básica do HTML

Conforme podemos visualizar na Figura 1 acima, um documento em HTML possui a cabeça (*head*), local onde vamos colocar informações que serão carregadas inicialmente; o corpo (*body*), local onde será incluído todo o conteúdo de nossa aplicação e objeto de nosso estudo.

Nos padrões mais antigos era comum usar apenas a implementação de um div (elemento de divisão do HTML) pois havia somente elas para podermos trabalhar. Embora essa estrutura seja invisível para o usuário, os navegadores e buscadores a leem. Na versão mais atual HTML5 é possível criar um código mais estruturado pois foram adicionadas novas *tags* para poder aumentar a semântica do HTML. Com isso podemos garantir uma experiência melhor pelo fato de o navegador identificar o que cada parte do documento será responsável.

```
<div>Cabeçalho da pagina</div>
<div>Conteudo relacionado</div>
<div>
  <a href="./pag1">pagina 1</a>
  <a href="./pag2">pagina 2</a>
</div>
<div>
  <div>
    Titulo do artigo
  </div>
  <div>
    <img src="./img.jpg">
    <p>conteudo</p>
  </div>
  <div>rodapé do artigo</div>
</div>
<div>Rodapé da pagina</div>
```

```
<header><h1>Cabeçalho da pagina</h1></header>
<aside>Conteudo relacionado</aside>
<nav>
  <a href="./pag1">pagina 1</a>
  <a href="./pag2">pagina 2</a>
</nav>
<section>
  <header>
    <h1>Titulo do artigo</h1>
  </header>
  <article>
    <img src="./img.jpg" alt="logo do artigo">
    <p>conteudo</p>
  </article>
  <footer>rodapé do artigo</footer>
</section>
<footer>Rodapé da pagina</footer>
```

Figura 2. Comparação de estrutura HTML e HTML5

Na Figura 2, podemos constatar que enquanto o primeiro código é mais limpo, o segundo é mais verboso e, portanto, de mais fácil leitura pelo navegador ou buscador pelo fato de incorporar uma maior semântica. A Figura 3 abaixo apresenta de uma forma simplificada como o navegador analisaria a estrutura do corpo de uma página feita somente com uso de *divs*, em contrapartida com uma página estruturada com as *tags* específicas do HTML5 (Mozilla, 2019)

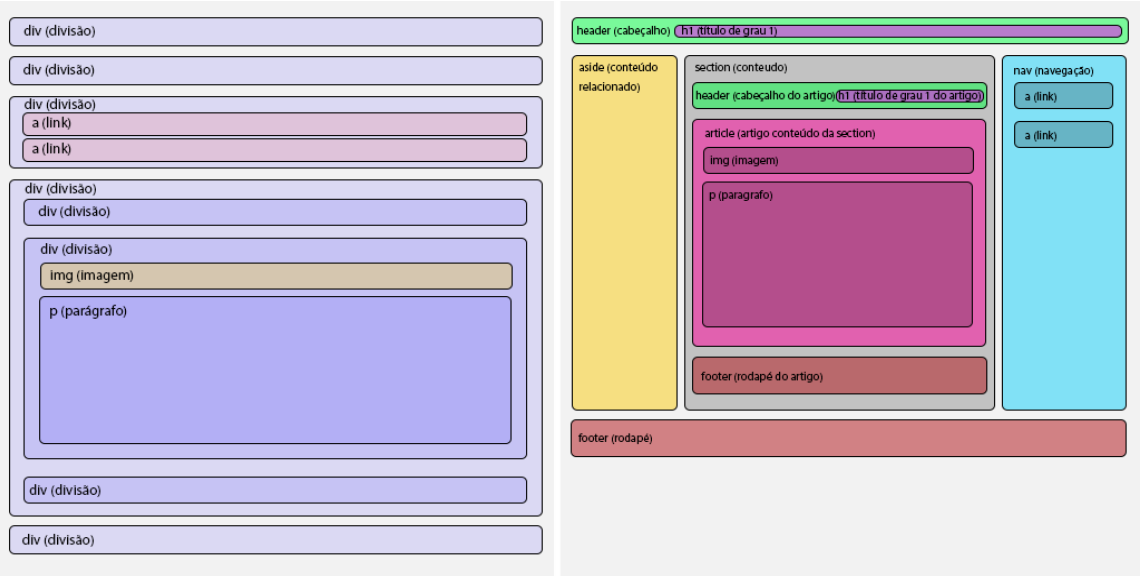

Figura 3. Visualização do HTML e HTML5

No primeiro modo, o navegador lê vários elementos de divisão, já o segundo modo - que faz uso das *tags* específicas - faz o navegador entender cada elemento e a que parte ele pertence. Dessa forma, acreditamos que a melhor estruturação da página beneficiará tanto os navegadores quantos as ferramentas para o deficiente visual. (W3C, 2020)

2.2. Dados Estruturados

De acordo com a empresa (Google, 2020)"Os dados estruturados são um formato padronizado para fornecer informações sobre uma página e classificar o conteúdo dela." Por exemplo, em uma página de evento, quando irá acontecer, onde será, e quem participará, podendo até sugerir a adição do evento na agenda do internauta.

Há várias formas de trabalhar com os dados estruturados. São elas:

- **Microformats** é uma forma mais antiga fazendo uso de classes em CSS (Cascading Style Sheets) para passar os padrões que serão utilizados (Microformats, 2020).
- **RDFa (Resource Description Framework in Attributes)** é uma extensão do HTML5 no qual usa atributos de *tags* para passar os padrões que serão utilizados. Esses padrões têm como base um vocabulário que proveniente do schema.org (RDFa, 2020).
- **Microdata** é uma especificação do HTML com uma comunidade aberta usada

para aninhar os dados estruturados, assim como o RDFa utiliza atributos de *tags* para passar os padrões que serão utilizados e tem como biblioteca base também schema.org (W3C, 2018).

- **JSON-LD (JavaScript Object Notation – Linked Data)** é uma notação em *JavaScript*, utilizando o JSON (JavaScript Object Notation) para estruturar os dados tem a vantagem de ser lido por buscadores mesmo sendo injetado dinamicamente (JSON-LD, 2020)

```html
<div class="h-card">
  <a class="u-photo u-url" href="http://example.org/photo.png">
    <span class="p-name">Fulano</span>
  </a>
  <div>Tel: <span class="p-tel">(99) 9 9999-9999</span></div>
  <div>
   E-mail: <a class="u-email" href="mailto:fulano@dominio.com">fulano@dominio.com</a>
  </div>
</div>
```

HTML

Figura 4. Microformats

Copy to clipboard

```html
<div vocab="http://schema.org/" typeof="Person">
  <a property="image" href="http://example.org/photo.png">
    <span property="name">Fulano</span>
  </a>,
  <div>Tel: <span property="telephone">(99) 9 9999-9999</span></div>
  <div>
   E-mail: <a property="email" href="mailto:fulano@dominio.com">fulano@dominio.com</a>
  </div>
</div>
```

HTML

Figura 5. RDFa

```html
<div itemscope itemtype="https://schema.org/Person">
  <a itemprop="image" href="http://example.org/photo.png">
    <span itemprop="name">Fulano</span>
  </a>,
  <div>Tel: <span itemprop="telephone">(99) 9 9999-9999</span>
  </div>
  <div>
   E-mail: <a itemprop="email" href="mailto:fulano@dominio.com">fulano@dominio.com</a>
  </div>
</div>
```

HTML

Figura 6. Microdata

```
Copy to clipboard
{
 "@context": "http://schema.org/",
 "@type": "Person",
 "image":"http://example.org/photo.png",
 "name": "Fulano",
 "telephone": "(99) 9 9999-9999",
 "email": "mailto:fulano@dominio.com",
}
JSON
```

Figura 7. JSON-LD

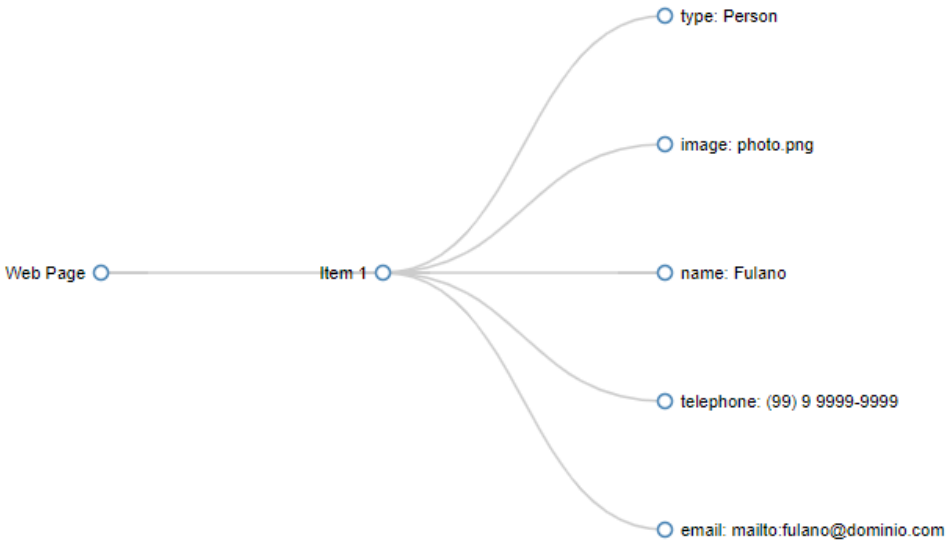

Figura 8. Visualização dos buscadores e navegadores

Nas Figuras 4, 5, 6 e 7 acima, apresentamos as mesmas formas de descrever estruturalmente porem, implementando uma padronização distinta. Conforme mencionado acima o Microformats utiliza classe para fazer a padronização dos dados e é necessário usar as os nomes estabelecido por ele como base de biblioteca, já as demais tecnologias apresentadas utilizam o schema.org como base de sua biblioteca.

Usar o a biblioteca é fácil, basicamente escolhe se um tipo e, posteriormente, adiciona suas possíveis descrições, no caso apresentado pela Figura 7, foi escolhido Person/h-card como tipo e os demais como por exemplo name/p-name, email/u-email... como descrições.

Por outro lado, na Figura 8 podemos visualizar como os buscadores e navegadores leem esses dados. Pode se notar para os dados estruturados é criado uma árvore de relacionamentos facilitando a compreensão do conteúdo, assim os resultados de buscas tornam se mais certeiros e enriquecendo da semântica do site oferece uma vantagem para ferramentas que usam tecnologia assistiva.

2.3. WAI-ARIA

Analisamos anteriormente algumas formas de aumentar a semântica da página web, porém, se acontecer uma ação como a abertura de um *pop-up*, uma notificação ou até mesmo se precisamos ter que mudar de abas ou movimentar algo, não conseguimos transmitir o real estado dessa ação apenas com as outras tecnologias apresentadas. Para solucionar esse problema foi criado uma tecnologia chamada WAI-ARIA, que significa *Web Accessibility Initiative* (Iniciativa pela Acessibilidade na Web) e *Accessible Rich Internet Applications* (Aplicações Ricas para uma Internet Acessível). Essa tecnologia provê maior semânticas pensando principalmente em informar locais importantes ou comportamento e ações da aplicação acessada. (Mozilla, 2020)

Essa tecnologia divide a semântica em duas partes (Eis, 2013):

- **Roles (Regras)** tipo do elemento interagido pelo usuário.
- **States/Properties (Propriedades/Estados)** estado e definição que o elemento está tendo.

Através dessas duas partes podemos dizer que um elemento do tipo (*Roles*) barra de progresso, que tem como propriedades (*Properties*) de início e fim (0-100), está no estado atual (*State*) de 50% concluído, ou que a elemento do tipo notificação, está no estado atual aberto apresentando algumas informações, desse modo o usuário tem uma representação melhor do que está acontecendo.

2.3.1 Roles

Dividido em 6 tipos, com cada tipo responsável por um determinado gênero de elemento. (W3C, 2017)

- **Abstract** utilizado para conceito geral e advertido que não deve se usar para representar um conteúdo.
- **Widgets** utilizado para definir elementos soltos, como *botões, links, tabs* etc.
- **Document Structure** utilizado para definir a estrutura de uma página não interativa, como *header, footer, table, article, list* etc.
- **Lendmarks** utilizado para definir conteúdos de importância para o usuário, como uma busca, conteúdo principal, navegação, formulários etc.
- **Live Region** utilizado para definir regiões ativas e que podem ser modificadas tais como um *timer* ou *status*.
- **Window** utilizado para definir janelas do navegador, como notificações.

```html
<header role="heading"><h1>Cabeçalho da pagina</h1></header>
<aside role="complementary">Conteudo relacionado</aside>
<nav role="navigation">
  <a href="./pag1" role="link">pagina 1</a>
  <a href="./pag2" role="link">pagina 2</a>
</nav>
<section role="main">
  <header role="heading">
    <h1>Titulo do artigo</h1>
  </header>
  <article role="article">
    <img src="./img.jpg" alt="logo do artigo">
    <p>conteudo</p>
  </article>
  <footer role="contentinfo">rodapé do artigo</footer>
</section>
<footer role="contentinfo">Rodapé da pagina</footer>
```

Figura 9. Código HTML com ROLES WAI-ARIA

Na Figura 9 podemos notar que acrescentamos algumas roles no HTML, em alguns casos acaba sendo redundando igual <article com role=”article”> e a semântica natural do HTML é sempre a mais indicada e que prevalece, mas no caso de algum programa desatualizado o segundo caso com o ARIA vem como um reforço de semântica, independente se for redundante é sempre bom reforçar, pois estaremos garantindo uma acessibilidade maior para nossa aplicação

2.3.2 States/Properties

Abaixo iremos demonstrar através de um código como usar uma States/Properties:

```html
<!DOCTYPE html>
<html lang=”pt-BR”>
  <head>
    <meta charset=”UTF-8” />
    <title>Título</title>
  </head>
  <body>
    <progress
      id="barraDeProgresso"
      role="progressbar"
      aria-valuemin="0"
      aria-valuemax="100"
      aria-valuenow="0"
      value="0"
      max="100">
      0% complete
    </progress>
  </body>
</html>
```

Figura 10. Código HTML com WAI-ARIA

Na Figura 10 temos uma barra de progresso representada pela role *progressbar* com as seguintes propriedades aria-valuemin 0 e aria-valuemax 100 com o estado atual representado por aria-valuenow 0, para que esse valor seja alterado utilizamos um código simples em *JavaScript,* ora apresentado pela Figura 11 a seguir. Esse código basicamente cria um *loop* que se repete por 100 vezes no intervalo de 200ms (sigla) atualizando o valor e o estado da barra de progresso.

```javascript
const barraDeProgresso = document.getElementById("barraDeProgresso");
let valorAtual = 0;
let intervalo = setInterval(atualizarProgresso, 200);
function atualizarProgresso() {
  valorAtual++;

  barraDeProgresso.setAttribute("aria-valuenow", valorAtual);
  barraDeProgresso.setAttribute("value", valorAtual);
  barraDeProgresso.textContent = valorAtual+ "% complete";

  if(valorAtual==100)
  clearInterval(intervalo);
}
```

JavaScript

Figura 11. Visualização dos buscadores e navegadores

3. Conclusão

É notável que utilizando as tecnologias descritas anteriormente, a web semântica fica rica em informações, informações essas que beneficia não só deficientes visuais e sim a web como um todo, visto que buscadores se beneficiam desses recursos para melhora a exatidão da busca. Logo o internauta ganha e a *webpage* também pois terá um posicionamento melhor.

Apesar dos dados estruturados fornecerem várias informações, não foi encontrado nenhum estudo que comprovasse uma melhora na acessibilidade do conteúdo pelos deficientes visuais. Uma das propostas deste artigo era avaliar o quanto uma página que utiliza essa padronização pode impactar positivamente a experiência de um usuário com deficiência visual. Sendo assim, desenvolvedores podem aproveitar essa tecnologia pois, mesclada com a inteligência artificial, seria mais fácil perguntar alguma coisa sobre o conteúdo para a aplicação ou até mesmo pedir para salvar um endereço nos contatos.

Infelizmente, devido a situação atual de pandemia do Covid-19, não foi possível fazer a avaliação do impacto, considerando o público alvo, deficientes visuais. A avaliação seria feita através da escala Likert, com perguntas sobre se a população com deficiência visual, considerando a diferença na navegação com páginas feitas nesse estudo usando tecnologias exploradas e, se no caso houvesse uma loja virtual 100% acessível, qual a confiança atribuídas por eles, na realização de um processo de compra on-line.

Apesar de não conseguirmos realizar essa análise, foi possível concluir que mesmo utilizando um documento HTML correto ou um dado estruturado, faz necessário o uso do WAI-ARIA pois ele garante que ações animadas sejam representadas para ferramentas assistivas assim garantindo uma acessibilidade maior para pessoas que utilizam dessas ferramentas.

Referências

Eis, D. (18 de Novembro de 2013). *Tableless.* Fonte: Tableless: https://tableless.com.br/wai-aria-estendendo-o-significado-das-interacoes/

Google. (17 de Julho de 2020). *Google developers.* Fonte: Google developers: https://developers.google.com/search/docs/guides/intro-structured-data?hl=pt-br

JSON-LD. (20 de Julho de 2020). *JSON-LD.* Fonte: https://json-ld.org/

Leal, L. N., & Thomé, C. (29 de junho de 2012). *Estadão*. Fonte: Estadão: https://www.estadao.com.br/noticias/geral,brasil-tem-45-6-milhoes-de-deficientes,893424

Microformats. (20 de julho de 2020). Fonte: http://microformats.org/

Mozilla. (2 de Setembro de 2019). *MDN web docs.* Fonte: MDN web docs: https://developer.mozilla.org/pt-BR/docs/Sections_and_Outlines_of_an_HTML5_document

Mozilla. (10 de Setembro de 2019). *MDN Web Docs*. Fonte: MDN Web Docs: https://developer.mozilla.org/pt-BR/docs/Web/HTML

Mozilla. (23 de Julho de 2020). *MDN Web Docs.* Fonte: MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Web_applications_and_ARIA_FAQ

RDFa. (20 de julho de 2020). *RDFa.* Fonte: https://rdfa.info/

W3C. (14 de Dezembro de 2017). *W3C.* Fonte: W3C: https://www.w3.org/TR/wai-aria-1.1/#role_definitions

W3C. (26 de Abril de 2018). *W3C.* Fonte: W3C: https://www.w3.org/TR/microdata/

W3C. (17 de julho de 2020). *W3C.* Fonte: W3C: https://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf