03

Js do Carousel

Transcrição

Pessoalmente, considero crucial que a primeira notícia do carrossel seja realmente a primeira, sem haver troca de ordem ou algo do tipo. O `h3` "Lista de artigos" poderia se tornar "Lista de artigos mais vistos na semana" ou algo similar, portanto, se a ordem importa, isto deve ser seguido, e neste caso o `ul` deveria ser substituído por `ol`.

Porém, a semântica que queremos passar é muito subjetiva, e depende da mensagem a ser passada. Sendo assim, manteremos `ol`, o que não interfirá no JavaScript.

Como dito anteriormente, iremos refazer a funcionalidade do carrossel, abrindo `listaDeArtigos.css`, em que temos:

```
.listaDeArtigos-item:not(:target):first-child {
    display: inline-block;
}

.listaDeArtigos-item:target {
    display: inline-block;
}
```

O trecho acima indica que queremos tudo que **não** seja o que está em `target` (alvo, em inglês). Vamos deixar este código comentado (caso prefira, você pode deletar). Voltaremos ao navegador e veremos que na parte do Blog nada funciona, portanto vamos refazê-lo em JavaScript.

Queremos que este comportamento de exibição de link, imagem e texto, aconteça direito mas, por padrão, o usuário precisa de pelo menos a primeira notícia - com o `display: block`, no caso. O que está acontecendo é que todas as notícias estão definidas com `display: none`.

Vamos alterar isto pelo menos para a primeira delas. Como fazemos isto via JS?

Abriremos `carousel.js` e, logo no início, acrescentaremos:

```
document.querySelector('#new0').style.display = 'block'
```

O JavaScript, como a maioria das linguagens de programação, começa a contagem do zero (`0`).

Ao retornarmos ao navegador e testarmos, a primeira notícia é exibida, porém o carrossel continua não funcionando, apesar do link âncora estar funcionando perfeitamente.

Queremos, de alguma forma, identificar que um *bullet* específico foi clicado. Então, o objetivo é que a segunda notícia fique com `display: block`, e o restante, com `display: none`, justamente para esconder o conteúdo.

Vamos voltar ao Brackets para abrir o `index.html`. O `nav` com `listaDeArtigos-slider` possui um atributo customizável, `data-sliderItem`, de valores `0`, `1` e `2`. Esta parte do código já estava pronta, para que pudéssemos focar um pouco mais na acessibilidade.

Salvaremos e abriremos `carousel.js`, e queremos que algo seja verificado quando apertarmos um botão, portanto teremos:

```
// Percorre todos os botoes controladores
btns.forEach(function(btn) {
    btn.addEventListener('click', function() {


    console.log(btn)
```

Testaremos isto no navegador, atualizando-o e verificando o Console, que nos traz justamente `data-slideritem="1"` ao passarmos pelo segundo *bullet*, e `data-slideritem="0"` quando o foco está no primeiro. É disto que precisávamos, e já temos uma forma de identificar esse `btn`.

Se o substituirmos em `console.log(btn)` por `this`, quem é que está sendo clicado, exatamente? O botão! Isto é, funcionará da mesma maneira. Vamos alterar o código para acessarmos o atributo `data-sliderItem`. Queremos checar se o primeiro botão é o `0`, e por isso usaremos o `if()`.

Como precisaremos acessar as três notícias, criaremos três variáveis, uma para cada:

```
// Variaveis

var btns = document.querySelectorAll('.listaDeArtigos-slider-item');

var new0 = document.querySelector('#new0');
var new1 = document.querySelector('#new1');
var new2 = document.querySelector('#new2');

new0.style.display = 'block'
```

Caso o `if()` seja verdadeiro, queremos que se pegue a primeira notícia e seja utilizado o `display: block`, enquanto o restante deverá ser escondido com o `display: none`. Caso isto não seja verdadeiro, precisaremos pegar o atributo `data-sliderItem` do botão que for clicado, para saber se é o `1` e, se este for o caso, queremos que se dê `display: none` em todos menos no `1`.

Por fim, precisaremos incluir a terceira condição também:

```
// Percorre todos os botoes controladores
btns.forEach(function(btn) {
    btn.addEventListener('click', function() {

     if (this.getAttribute('data-sliderItem') === '0') {
        new0.style.display = 'block';
        new1.style.display = 'none';
        new2.style.display = 'none';
     } else if (this.getAttribute('data-sliderItem') === '1') {
        new0.style.display = 'none';
        new1.style.display = 'block';
        new2.style.display = 'none';
     } else {
        new0.style.display = 'none';
        new1.style.display = 'none';
```

```
            new2.style.display = 'block';
        }
```

Tudo isso que estamos verificando é justamente para vermos qual botão foi clicado. No lugar de `getAttribute`, poderíamos usar `id`.

Salvaremos para testar as alterações no navegador, e confirmaremos que o carrossel funciona bem, porém, se utilizamos "Shift + Tab", o foco vai para cima, apesar da classe de seleção estar de acordo com o que queríamos definir.

O NVDA lê para onde está indo, pois trata-se de um link:

> cabeçalho nível 2
>
> 5 dicas para melhorar a acessibilidade em sua interface
>
> Será que acessibilidade é algo tão complicado que vale a pena eu abrir mão de 25% dos meus usuários? Acessibilidade normalmente é um algo não muito aplicado por aí, falta de informação talvez?
>
> linque
>
> Continuar lendo o post 3

Poderíamos melhorar o texto do link "Continuar lendo o post 3", colocando um `aria-labelledby` e trocando `3` pelo título da notícia, mas por ora focaremos mais no comportamento do carrossel em si.