Interacción accesible

JavaScript e accesibilidade

O código JavaScript pode incluírse nunha páxina web sen que presente un obstáculo para ninguén, coa condición de que estea implementado de forma accesible e sexa compatible cos produtos de apoio.

Neste documento vanse mostrar diversas técnicas para facelo.

Independencia do dispositivo

Para que os scripts e elementos de programación poidan ser operados dende calquera dispositivo, e por tanto poidan ser operados non só co rato, senón tamén co teclado ou cunha interface de teclado, é necesario usar correctamente os manexadores de eventos de JavaScript.

Hai manexadores de eventos que non dependen de dispositivos concretos e recoñecen tanto eventos de rato como de teclado: onsubmit, onreset, onfocus, onload, onselect.

Tamén hai manexadores específicos do rato, que teñen o seu equivalente para teclado:

- onmousedown -> onkeydown
- onmouseup -> onkeyup
- onmouseover -> onfocus
- onmouseout -> onblur

Se se usan eventos de teclado xunto con eventos do rato, asegúrase que se poderá interactuar co contido a través dunha ampla variedade de dispositivos.

Hai exemplos de como facelo nas técnicas SCR2 e SCR20 das WCAG 2.1.

O seguinte exemplo está na técnica SCR2.

```
<a href="https://www.w3.org/wai" onmouseover="updateImage('wai', true);"
onfocus="updateImage('wai', true);" onmouseout="updateImage('wai',false);"
onblur="updateImage('wai',false);">
 <img src="greyplus.gif" border="0" alt="" id="wai">W3C Web Accessibility Initiative
</a>
```

No exemplo anterior, cámbiase a imaxe cando o rato pasa por riba. Para proporcionar unha experiencia similar, independente do dispositivo, a imaxe tamén cambia cando se accede co tabulador, utilizando os manexadores de eventos de teclado (onfocus e onblur), é dicir, cando recibe e perde o foco.

Evento “onclick”

O evento **onclick** é o evento da acción por defecto de enlaces e botóns e actívase por teclado e por rato. Polo tanto, con enlaces e botóns non é necesario utilizalo xunto a **onkeypress**, que sería o evento de teclado equivalente.

Sen embargo, cando se usa o evento **onclick** con outros elementos que non sexan enlaces e botóns (, <img>, <td>, etc), para que o código asociado ao evento poida ser invocado tamén por teclado, sí deberá usarse xunto ao seu evento equivalente de teclado **onkeypress**.

Pero non serve de nada incluír o evento de teclado **onkeypress** se estes elementos non poden coller o foco por teclado, é dicir, non se pode chegar a eles co tabulador e polo tanto non poden activarse por teclado.

Por iso debe engadirse a estes elementos o atributo **tabindex**:

- tabindex=”0” para que se poida acceder a dito elemento por teclado mediante tabulador, na orde secuencial de navegación do teclado. A orde de tabulación depende do contido do documento.
- tabindex=”-1” significa que o elemento non pode coller o foco a través do teclado pero si a través de JavaScript ou pulsando co rato. Úsase para crear widgets accesibles con JavaScript.
- Un valor positivo significa que o elemento debe coller o foco na orde secuencial de navegación. A orde está definida polo valor do número. É dicir, tabindex="4" colle o foco antes de tabindex="5" e tabindex="0", pero despois de tabindex="3". Se múltiples elementos comparten o mesmo valor positivo, a súa orde segue a súa posición no documento. O valor máximo de tabindex é 32767. Se non se especifica, o valor por defecto é 0.

Máis información sobre tabindex.

Por exemplo:

```
<img src="arrow.gif"
   onclick="nextPage();"
   onkeypress="nextPage();"
   tabindex="0"
   alt="Ir a la siguiente página” />
```

O exemplo anterior utiliza os eventos **onclick** e **onkeypress** para que funcione co rato e co teclado (coa tecla “enter” ou coa barra espaciadora). Pero isto non sería suficiente. Se non se incluíse o atributo **tabindex=”0”**, a imaxe non podería coller o foco.

O exemplo mostra como permitir pulsar nunha imaxe. De todas formas, sempre é máis recomendable utilizar un elemento “button” ou un “input” de tipo imaxe para facilitar a accesibilidade.

Funcións DOM e contido dinámico

O DOM (Modelo de Obxectos do Documento) representa os documentos nunha estrutura de árbore.

Existen métodos para obter un elemento e os seus atributos, modificalos, eliminalos ou crear novos. É dicir, pode manipularse de forma dinámica o contido do documento.

Ao usar as funcións DOM auméntase a compatibilidade cos axentes de usuario e produtos de apoio, xa que estas aplicacións acceden ao DOM para recuperar o contido.

Non se debe usar o método **document.write** ou as propiedades “innerHTML”, “outerHTML”, “innerText” ou “outerText” para insertar ou modificar contido dinamicamente no documento, xa que non son parte da especificación DOM.

Mellor é usar DOM para xerar e manipular dinamicamente o contido.

Máis información nas técnicas SCR26 e SCR27.

JavaScript non intrusivo

Un principio básico de deseño de programación é a separación entre a presentación, o contido e o comportamento da páxina.

JavaScript non intrusivo permite separar a programación da páxina (comportamento) do contido e presentación.

No seguinte código vese como está misturado o comportamento da páxina co seu contido:

```
<input type=”button” id=”btn” value=”pulsa aquí” onclick=”helloWorld();” />
```

Unha forma de facer o mesmo usando JavaScript non intrusivo sería (podería facerse algo equivalente con JQuery):

ficheiro: myScript.js

```
window.onload = function() {
    document.getElementById(“btn”).onclick = helloWorld;
}

function helloWorld() {
    alert(“hola”);
}
```

ficheiro: index.html

```
...
<input type=”button” id=”btn” value=”pulsa aquí” />
…
```

O uso de JavaScript non intrusivo favorece a escalabilidade, mantemento e deseño de sitios web. O código HTML é máis limpo, lixeiro e semántico e mellora a accesibilidade á páxina.

Referencias

Para a elaboración deste material utilizáronse, entre outros, os recursos que se enumeran a continuación:

- https://www.w3.org/WAI/WCAG21/Techniques/