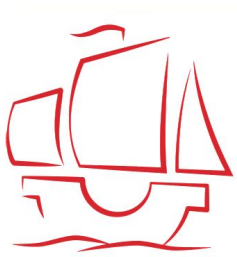


FORMACIÓN PROFESIONAL
MONTECASTELO

Gestión de eventos

Desarrollo Web en Entorno Cliente

Ciclo Superior de Desarrollo de Aplicaciones Web

2023/2024

Eventos

- El entorno notifica a JavaScript en el momento en el que se produce un determinado evento en la página o el navegador.
- El evento está representado por un objeto de JavaScript con propiedades y métodos que permiten acceder a la información de lo que ha ocurrido.
- Un manejador de eventos es una función que está registrada para atender un determinado evento.

Eventos y nodos DOM

- El encargado de gestionar los eventos es el DOM.
- Los manejadores de eventos se registran en un determinado nodo del DOM.
- Los manejadores de eventos son notificados solamente cuando el evento se produce en el nodo para el cual se han registrado.

Registrar un manejador

- El siguiente método puede emplearse para registrar un manejador de eventos en un determinado nodo:

```
nodo.addEventListener(“tipo Evento”, manejador)
```

Registrar un manejador en línea

- También se puede registrar un manejador en el atributo cuyo nombre es el nombre del evento precedido por on.
- Es lo que se denomina un manejador en línea.

```
<p onclick="saludo( )">Pulsa aquí</p>
```

Registrar un manejador en línea

- Pero a través de un atributo solamente se puede registrar un manejador para un determinado evento.
- En cambio `.addEventListener()` permite registrar un número indefinido de manejadores.

Eliminar un manejador

- node.removeEventListener(manejador)
- El nombre del manejador pasado debe ser el mismo que el que se pasó a la función .addEventListener()

Objetos Evento

- Cuando un manejador de eventos es invocado, recibe como argumento un objeto que representa el evento generado.
- El objeto en cuestión contiene información adicional acerca del evento.
- La información depende del tipo de evento, pero todos los eventos tienen una propiedad `type` que almacena una cadena que identifica el tipo de evento: “`click`”, “`mousedown`”, etc.

Propagación de eventos

- Para la mayoría de los eventos, un nodo padre recibe también los eventos que se generan en sus nodos descendientes (propagación).
- Por ejemplo, un párrafo que contiene un botón, también recibe los eventos "`click`" del botón.
- En cualquier caso, de existir varios manejadores, se llaman desde el nodo en el cual se generó el evento hacia sus ancestros.
- En el caso anterior, primero se llamaría al manejador del botón y después al del párrafo (si existiese).

Propagación de eventos

- En cualquier momento, un manejador puede invocar `.stopPropagation()` para detener la propagación de un evento.
- De ese modo, el evento no se seguirá propagando hacia el resto de los ancestros del elemento que detiene la propagación.

`target` de un evento

- La mayoría de los eventos tienen una propiedad `.target` que contiene el nodo en el que se generó el evento.
- Puede ser empleada para evitar gestionar un evento propagado, ya que permite identificar el nodo original sobre el que se produjo el evento.
- Por tanto permite identificar qué nodo concreto ha generado un evento propagado.

Acciones por defecto

- La mayoría de los eventos tienen asociada una acción por defecto.
- Por ejemplo, hacer clic en un enlace por defecto carga una nueva página.
- Los manejadores de JavaScript se invocan primero, y a continuación se llevan a cabo las acciones por defecto.
- El método `.preventDefault()` puede emplearse dentro de un manejador para evitar que un evento ejecute las acciones por defecto.

Evento de carga

- Cuando el navegador termina de cargar una página se lanza un evento `load`.
- Este evento es fundamental, ya que se emplea para llevar a cabo las acciones de inicialización.
- Cabe recordar que el código de JS se ejecuta tan pronto se encuentra la etiqueta <script> que lo contiene, por tanto no se ha cargado todavía contenido situado después de la misma.
- En cambio, cuando se lanza el evento `load`, se tiene la certeza de que todos los elementos del documento han sido cargados.

Eventos de teclado

- Cuando se presiona una tecla se lanza un evento `keydown`.
- Si la tecla se mantiene pulsada, se lanzan periódicamente eventos `keydown`.
- Cuando se libera la tecla, se lanza un evento `keyup`.
- Los eventos de teclado se generan en el elemento que tiene el foco cuando se pulsa la tecla.

Eventos de teclado

- Los eventos de teclado tienen una propiedad `key` que registra la tecla que fue pulsada.
- Además, existen propiedades booleanas `shiftKey`, `ctrlKey`, `altKey` y `metaKey` que detectan las pulsaciones de las teclas asociadas.
- Éstas últimos, se pueden emplear para detectar combinaciones de teclas tanto en eventos de teclado como de ratón.

Eventos de puntero

- En la actualidad existen dos métodos fundamentales para señalar puntos de la pantalla: los ratones y las pantallas táctiles.
- Ambos tipos producen eventos diferentes.
- Los ratones incluyen también dispositivos que se comportan de modo análogo, tales como *touchpads* y *trackballs*.

Eventos de click de ratón

- Cuando se hace clic con el ratón se lanza un evento `mousedown`.
- Cuando se libera el botón del ratón, se lanza un evento `mouseup`.
- Después del evento `mouseup` se genera un evento `click`.
- Si presiona el ratón dos veces, se genera un evento `dblclick`.

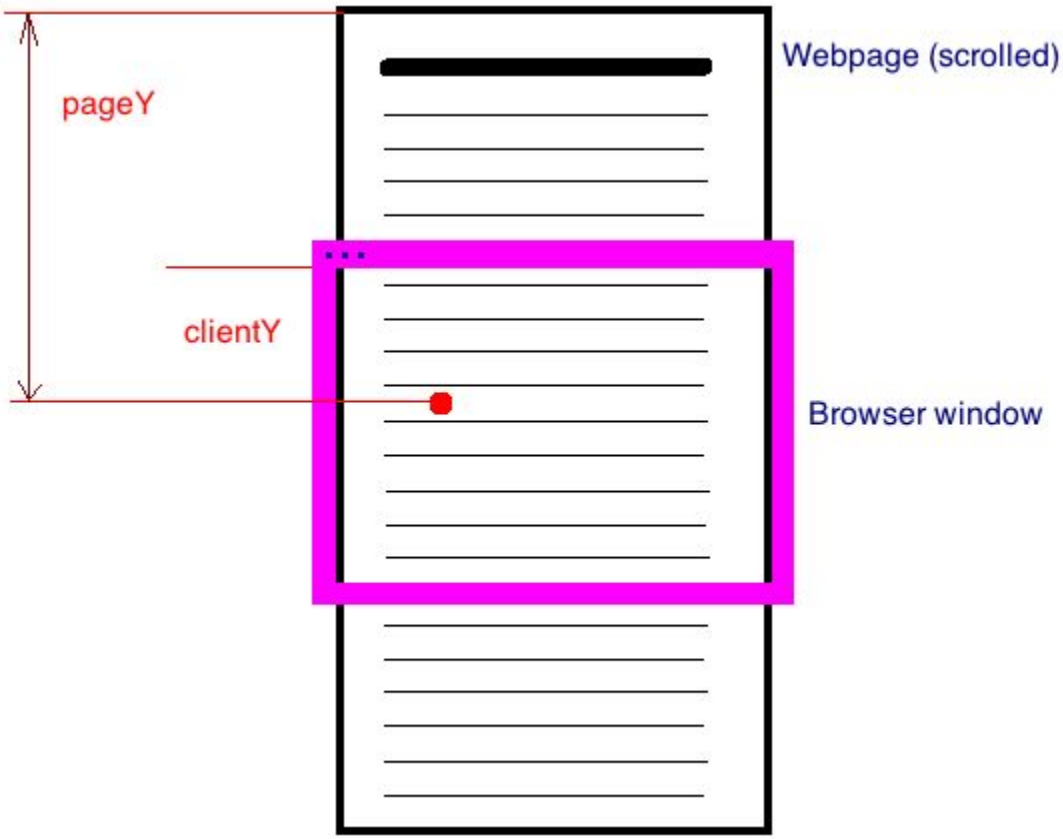
Eventos de click de ratón

- Los eventos del ratón se generan en el elemento que está justo debajo del puntero del mismo.
- En concreto, el evento `click` se genera en el elemento más específico que contenga tanto al elemento de pulsación como al de liberación del botón correspondiente.
- Por ejemplo, si se presiona el ratón en un botón y, sin soltar, se arrastra a otro botón y se suelta, el evento se generará en el primer elemento que contenga a ambos botones.

Eventos de click de ratón

- Para averiguar en qué coordenadas se produjo el evento se pueden emplear las propiedades presentadas a continuación.
- `clientX` y `clientY`, relativas a la parte superior izquierda del área visible del documento.
- `pageX` y `pageY`, relativas a la parte superior izquierda del documento HTML.
- Por tanto, estas propiedades pueden no coincidir si se se ha desplazado (*scroll*) la ventana del navegador.

Eventos de click de ratón



Eventos de movimiento de ratón

- Cada vez que se mueve el puntero del ratón se lanza un evento `mousemove`.
- Este evento puede ser empleado para detectar la posición del ratón, así como los desplazamientos del mismo.

Eventos táctiles

- Los navegadores en pantallas táctiles también soportan los eventos de ratón.
- Además, soportan los eventos específicos descritos a continuación.
- Cuando un dedo comienza a presionar una pantalla se lanza un evento `touchstart`.
- Si se mueve un dedo mientras se desplaza por la pantalla se lanza un evento `touchmove`.
- Cuando el dedo deja de tocar la pantalla se lanza un evento `touchend`.

Eventos de scroll

- Cuando un elemento en pantalla se desplaza se lanza un evento `scroll`.
- Estos eventos pueden emplearse para para conocer qué parte del documento está visualizando el usuario.
- Por ejemplo, para mostrar alguna indicación del progreso a través de una barra de progreso o número de página.

Eventos de foco

- Cuando un elemento en pantalla tiene el foco se lanza un evento `focus`.
- Cuando un elemento pierde el foco se lanza un evento `blur`.
- Al contrario de todos los eventos anteriores, estos eventos **no se propagan**.

Referencias

- JavaScript en w3schools
- JavaScript en Mozilla Developer Network

FORMACIÓN PROFESIONAL

MONTECASTELO