# interactuando con el usuario

No debemos perder el foco y tenemos que tener nuestro horizonte a la vista. Buscamos desarrollar aplicaciones que interactúen con el usuario. No es menor ahondar en qué significa esto.

Los usuarios son aquellos que dan uso a nuestras aplicaciones. Y obviamente, si bien somos nosotros al programar quienes damos luz a nuestras aplicaciones, **son estas quienes interactúan con el usuario.**

# interactuando con el usuario

De este modo es que las aplicaciones que programamos tienen que estar capacitadas para interactuar con el usuario. Pero, **¿qué es interactuar con el usuario?**

Dijimos que el usuario es una persona que usa nuestra aplicación. Somos usuarios de muchas aplicaciones (incluso de las que creamos nosotros) por lo que sabemos que **como usuarios actuamos en las aplicaciones**. Producimos eventos en ellas: *movemos el mouse, apretamos una tecla, hacemos un click, pasamos el cursor encima de algo, scrolleamos la pantalla*. Acciones a las que algunas veces, la aplicación responde y algunas a las que no.

# interactuando con el usuario

Estas acciones que las aplicaciones realizan en base a las acciones que hace el usuario (***eventos***) las llamaremos **reacciones**. ¿Por qué? Porque serán acciones que estarán basadas en la acción original.

*Solo si el usuario apretó la tecla espacio, la aplicación pausará el video*. Es decir, la reacción de pausar el video ocurrirá sólo si hay una acción previa: apretar la tecla espacio.

# ¿qué tipos de acciones existen?

A su vez, las acciones que un usuario puede hacer están divididas en tipos de acciones. Para poner un ejemplo, *apretar la tecla espacio y apretar la tecla R pertenecen al mismo tipo de acción: apretar una tecla*.

Si bien existen más, se destacan los siguientes:

- **click**
- **mousedown** (poner el cursor encima de)
- **keypress** (apretar una tecla)

# ¿cómo reaccionar?

Como desarrolladores, debemos preparar nuestra aplicación entonces, para reaccionar ante determinadas acciones. En particular Javascript nos presenta su ***sistema de notificicación de eventos*** que lo que hace, justamente, es notificarnos cuando un usuario lleva a cabo una acción enviándonos un evento avisando tanto ***qué*** acción llevó a cabo (apretó una tecla) como ***cómo*** la llevó a cabo (qué tecla).

Por lo tanto, como programadores, debemos estar alerta y a la espera de que Javascript nos notifique el suceso de un nuevo evento para, si queremos, *reaccionar a él*.

# ¿cómo reaccionar?

Como dijimos, *todo sucede en un elemento*. Esto también se aplica, claro, a los eventos. Ocurren sobre ellos y esto es útil. Nos van a interesar aquellos eventos que ocurran en determinados elementos. *No nos interesa si el usuario hace un click en el párrafo sino que nos interesa si lo hace en un botón*.

Por lo que nuevamente, para estar alerta a los eventos ocurridos en un elemento, debemos primero identificar a este elemento.

# ¿cómo reaccionar?

Los elementos hemos dicho que *son representados en un objeto de Javascript*. Una de las propiedades que este objeto tiene es una función llamada ***addEventListener***. Su objetivo es activar esta escucha hacia un tipo de evento en particular, y establecer la reacción a este evento.

Para hacer esto, obviamente **tenemos que tener previamente definido a qué tipo de evento queremos reaccionar y cómo.**

# ¿cómo reaccionar?

La función ***addEventListener*** definirá la escucha de un tipo de evento producido sobre un elemento y recibirá ***dos parámetros***. El primero será el **tipo de evento** a escuchar (*click*, *mousedown*, *keypress*) y el segundo, una función que se ejecutará cada vez que este evento ocurra en este elemento.

```
window.onload = function() {
  const boton = document.querySelector('.boton')

  function capturarClick() {
    console.log('¡capturé el click!')
  }

  boton.addEventListener('click', capturarClick)
}
```

# ¿qué es un callback?

Esta función, de ahora en más llamada ***callback***, será la reacción. *Cada vez que el usuario ejecuta la acción definida sobre el elemento se ejecutará el callback.*

Este callback será una función que recibirá un único parámetro: el evento en cuestión. Recordemos que por ejemplo, hasta ahora pudimos definir que se ejecute un callback cuando el usuario apretó una tecla, pero nunca pudimos saber qué tecla apretó. Esto será lo que nos permita **este parámetro del callback**, el ***evento*** propiamente dicho.

# ¿qué es un callback?

**Este evento será un objeto que explique qué es lo que hizo el usuario**.

Sus propiedades variarán según el tipo de evento que se ejecutó. No tendremos qué tecla apretó el usuario si lo que hizo fue un click.

Teniendo en cuenta estas propiedades podremos diferenciar las reacciones que va a tener nuestra aplicación. **Es decir, reaccionar en base a las características de la acción que ocurrió.**

# ¿qué es un callback?

```
window.onload = function() {
  const nombre = document.querySelector('.nombre')

  function capturarTecla(event) {
    if (event.key === 'a') {
      console.log('¡apretó a!')
    }
  }

  nombre.addEventListener('keypress', capturarTecla)
}
```

El código en cuestión captura el evento ***keypress*** (*cuando el usuario aprieta una tecla*) de **un elemento cuya clase es *nombre*.** La reacción es evaluar si la tecla apretada (*propiedad **key** del objeto **event***) es la ***a***, y en ese caso, imprimir en consola un mensaje acorde.

# ejercicio de eventos en dom ✍️

1. Crear una aplicación que tenga *un campo para ingresar una edad* y *un botón*. Si la edad agregada es menor a 18, mostrar un **cartel rojo** que diga que es menor de edad. En el caso contrario, mostrar un **cartel verde** que le de la bienvenida.

2. Al apretar ***enter*** dentro del campo para ingresar la edad, el usuario debe poder ver el cartel mencionado en el primer ejercicio.

3. Prohibir que el usuario ingrese algo que no sea un número en el campo para ingresar la edad.