TEMA 4

JAVASCRIPT
EVENTOS, EXPRESIONES, COOKIES

Desarrollo web entorno cliente

2 DAW

María Millán Gamero

Índice

1. Introducción a eventos

PROPORCIONA INTERACCIÓN CON EL USUARIO

EVENTOS= idioma en el que JavaScript entiende las acciones del usuario.

- Es el mecanismo por el que se consigue establecer una comunicación bidireccional y en tiempo real entre aplicación y usuarios.

ASÍNCRONO

"Se prepara" el código para cuando se produzca el evento el programa lo captura y se ejecuta.

LOS EVENTOS JAVASCRIPT SE ASOCIAN A ELEMENTOS CONCRETOS DEL DOM

2. Captura de eventos

Para capturar eventos se define un **listener** (“oyente”) → **addEventListener** sobre el elemento que nos interese.

```
document.addEventListener("DOMContentLoaded", init)

function init(){
  contarClics();
}

function contarClics(){
    var nombre= document.getElementById("nombre");
    var num_veces=0;
    var clics = document.getElementById("numero");

    nombre.addEventListener("click", () =>{
        num_veces++;
        console.log(`El usuario ha clicado ${num_veces} veces
en mi nombre`);
        clics.innerText = num_veces;

    })
}
```

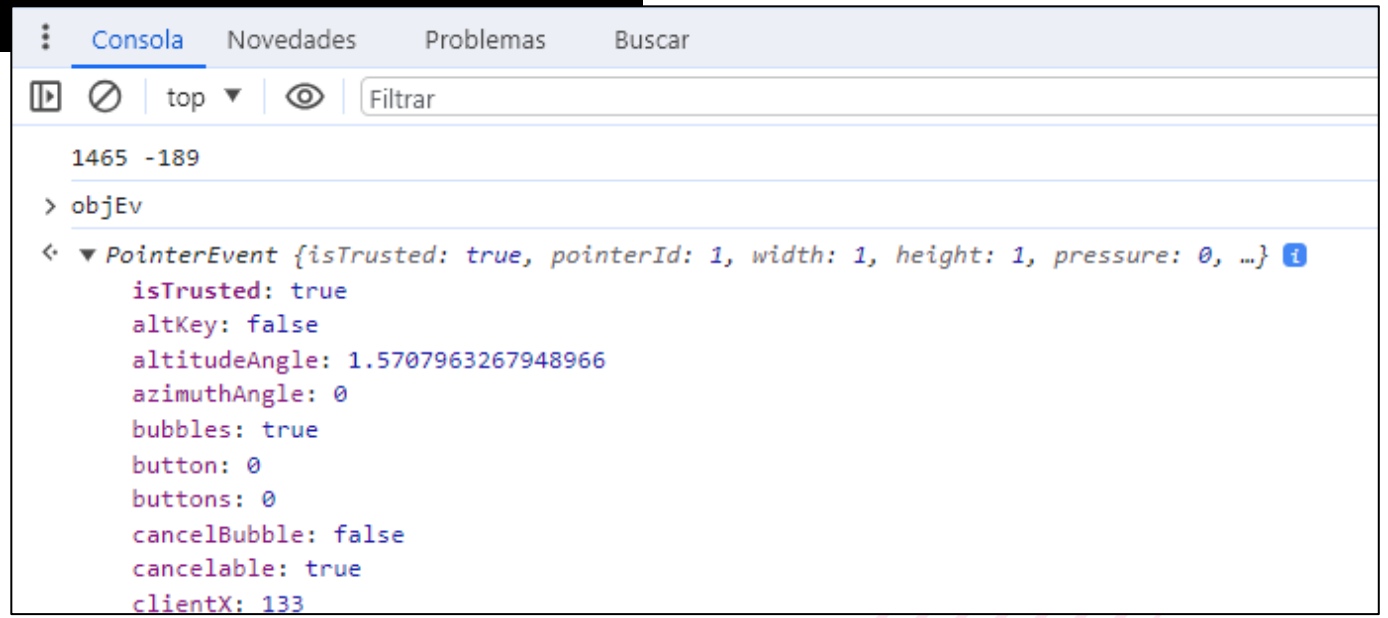
```
<p>Captura de clics en mi
    <span id="nombre"> nombre.</span>
</p>
<p>Número de clics:
    <span id="numero" ></span>
</p>
```

3. Objeto del evento

- Cada evento tiene asociado un objeto, basado en la interfaz **Event,** que contiene propiedades y métodos para obtener más información sobre lo sucedido (*el evento que se ha producido*).

```
function evento(){
    var miBody = document.getElementsByTagName("body")[0];
    miBody.addEventListener("click", (objEv) =>{
        console.log(`${objEv.screenX}`, `${objEv.screenY}`)
    });
}
```

Nos permite conocer la posición en la que se ha hecho clic en la pantalla



3. Objeto del evento

Propiedades del objeto evento

Propiedad	Utilidad
altKey	Devuelve si la tecla [Alt] fue pulsada durante el evento.
button	Devuelve el botón del ratón que activó el evento: ■ 0: botón principal. ■ 1: botón central. ■ 2: botón secundario. ■ 3 y 4: cuarto y quinto botones (si los hubiera).
charCode	Contiene el valor Unicode de la tecla que se pulsó (evento **keypress**).
clientX	Coordenada X del ratón con respecto a la ventana.
clientY	Coordenada Y del ratón con respecto a la ventana.
ctrlKey	Devuelve si la tecla [Ctrl] fue pulsada durante el evento.
pageX	Coordenada X del evento, relativa al documento completo.
pageY	Coordenada Y del evento, relativa al documento completo.
screenX	Coordenada X del evento con respecto a la pantalla.
screenY	Coordenada Y del evento con respecto a la pantalla.
shiftKey	Devuelve si la tecla [Mayús] fue pulsada durante el evento.
target	Referencia al elemento que lanzó el evento.
timeStamp	Devuelve el momento en el que se creó el evento.
type	Nombre del evento.

4. Cancelación de eventos

Hay eventos:

- **Predeterminados**: como por ejemplo el elemento <a> que por defecto un evento click (href).
- **Personalizados**.

¿Cómo cancelar su comportamiento por defecto?

```
/*CAPTURAMOS EL EVENTO Y
ANULAMOS SU COMPOTAMIENTO PREDETERMINADO*/
function cancelarEvento(){
    var enlace = document.getElementById("enlace");
    enlace.addEventListener("click", (evento)=>{
        evento.preventDefault();
    })
}
```

De esta forma no realiza la acción que tiene definida de forma predeterminada., aunque podemos indicarle que realice otras acciones.

4. Cancelación de eventos

Lo mostrado en la diapositiva 7 es la cancelación del comportamiento por defecto, pero no del evento en sí.

Para anular completamente un evento tenemos que usar:

- **removeEventListener()**


```
document.addEventListener("DOMContentLoaded", init)

function init() {
    var clic = document.getElementById("nombre");
    clic.addEventListener("click", contarClics);
}

var num_veces = 0;
function contarClics() {
    var clics = document.getElementById("numero");
    num_veces++;
    console.log(`El usuario ha clicado ${num_veces} veces en mi nombre`);
    clics.innerText = num_veces;
    if (num_veces == 5) {
        nombre.removeEventListener("click", contarClics);
    }
}
```

5. Tipos de eventos
Eventos de formularios

document.forms → contiene todos los formularios del documento.

- Devuelve un HTMLCollection, por lo que podremos acceder a cada elemento según su posición.

Propiedades y métodos del objeto forms

Propiedad	Utilidad
action	Contiene la URL que recibirá los datos del formulario.
elements	Contiene todos los controles del formulario.
length	Número de controles del formulario.
method	{GET|POST} en función del método elegido para enviarlo.
enctype	Tipo de codificación de los datos del formulario.
acceptCharset	Conjunto de caracteres del formulario.
submit()	Envía los datos del formulario a la URL de **action** usando **method.**
reset()	Devuelve el formulario a su estado inicial.
on*X*	Todos los eventos asociados al formulario, siendo ***X*** el nombre del evento: **onAbort, onBlur, onCancel, onClick…**

5. Tipos de eventos
Eventos de formularios

- Submit → evento más importante relacionado con los formularios, el cual nos permite enviar los datos.
- Las validaciones de datos que queramos hacer hay que hacerlas antes del envío del formulario.
- Si la información no se valida tenemos que cancelar el comportamiento predeterminado de submit para que los datos no se envíen.

```
<form name="formulario">
        Introduzca una edad mayor de 18 y menor de 65:
        <input type="text" name="edad" placeholder="Edad" id="edad">
        <input type="submit"name="submit" value="Enviar">
</form>
<div id="errores"></div>
```

```
var formulario = document.forms[1];
    var error=document.getElementById("errores");
    formulario.addEventListener("submit", (evento)=>{
        var edad = document.getElementById("edad").value;
        var edad = formulario.elements["edad"].value;
        if(edad<=18 || edad>65){
            evento.preventDefault();
            error.innerHTML='<p>La edad debe contenerse entre 18 y 65</p>'
        }
    })
```

5. Tipos de eventos
Eventos de formularios

CHANGE

Se lanza cuando se realiza un cambio de valor en un control del formulario, y abandona el foco.

```
<form name="formulario_foco">
        Introduzca una edad mayor de 18 y menor de 65:
        <input type="text" name="edad" placeholder="Edad" id="edad">
        <input type="text" name="email" placeholder="Email" id="email" required>
        <input type="text" name="telefono" placeholder="Teléfono" id="telefono"
                                                                  required>
        <input type="submit"name="submit" value="Enviar">
    </form>
```

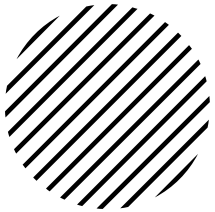
FOCUS | BLUR

Cuando gana el foco (activar control) se lanza el evento **focus** y cuando lo pierde (pierde el control) se lanza el evento **blur**.

```
 var elementos = Array.from(document.forms[2].elements);
    elementos.forEach(element => {
        element.addEventListener("focus", ()=>{
            if(element.hasAttribute("required")){
                document.getElementById(element.id).value="...@gmail.com"
            }
        })

        element.addEventListener("blur", ()=>{
            if(element.hasAttribute("required")){
                document.getElementById(element.id).style.borderColor="red";
                document.getElementById(element.id).title="Campo requerido"
            }
        })
    });
```

5. Tipos de eventos
Eventos de ratón



- Se crean con el ratón o con el dedo en las pantallas táctiles.

```
<div id="raton" style="background-color: brown;">
    <h1>Evento de ratón</h1>
</div>
```

```
var titulo = document.getElementById('raton');
titulo.addEventListener('mouseover', function ()
{
    titulo.style.backgroundColor = 'blue';
});
titulo.addEventListener('mouseleave', function ()
{
    titulo.style.backgroundColor = 'brown';
});
```

Evento	Funcionamiento
click	Hacer clic sobre el botón principal del dispositivo.
dblclick	Hacer doble clic sobre el botón principal del dispositivo.
mousedown	Cuando se pulsa y justo antes de soltarlo, se lanza el evento.
mouseup	El evento se lanza cuando se suelta el botón.
mousenter	El evento se lanza cuando el puntero se sitúa sobre el elemento que captura el evento.
mouseleave	El evento se lanza cuando el puntero deja de situarse sobre el elemento que captura el evento.
mousemove	Mientras se está dentro del elemento, el evento se lanza cada vez que se mueve el puntero.
mouseover	El evento se lanza cuando el puntero se sitúa sobre el elemento que lo captura o sobre cualquiera de sus hijos.
mouseout	El evento se lanza cuando el puntero deja de situarse sobre el elemento que lo captura o sobre cualquiera de sus hijos.
contextmenu	El evento se lanza cuando se solicita un menú contextual.

5. Tipos de eventos

Eventos de teclado

Evento	Funcionamiento
keypress	El evento se lanza tras pulsar y soltar una tecla.
keydown	El evento se lanza tras pulsar y antes de soltar la tecla.
keyup	El evento se lanza tras soltar la tecla.

```
let todoBody = document.getElementsByTagName("body")[0];
todoBody.addEventListener("keypress",(evento)=>{
        console.log(`Lanzado KEYPRESS con la tecla ${evento.key}`);
});
todoBody.addEventListener("keydown",(evento)=>{
        console.log(`Lanzado KEYDOWN con la tecla ${evento.key}`);
});
todoBody.addEventListener("keyup",(evento)=>{
        console.log(`Lanzado KEYUP con la tecla ${evento.key}`);
});
```

```
Lanzado KEYDOWN con la tecla d
Lanzado KEYPRESS con la tecla d
Lanzado KEYUP con la tecla d
>
```

5. Tipos de eventos
Eventos de arrastrar y soltar

- **Drag&Drop** = Son eventos que se lanzan al utilizar elementos definidos como arrastrables → atributo draggable = true.

- Como se trata de una acción con un origen y un destino, deben diferenciarse los dos conjuntos de eventos.

Evento	Funcionamiento
Elemento que se arrastra	
drag	Cada vez que se arrastra una vez que se ha iniciado el arrastre.
dragstart	Al comenzar a arrastrar el elemento.
dragstop	Al finalizar el arrastre del elemento.
Elemento donde puede soltarse	
dragenter	Cuando el elemento que se arrastra entra en el elemento destino.
dragleave	Cuando el elemento que se arrastra sale del elemento destino.
dragover	Cada vez que continúa el arrastre sobre el elemento de destino.
drop	Cuando el elemento que se arrastra se suelta sobre el elemento destino.

Ordena la lista según tus preferencias

- Pizza
- Hamburguesa
- Macarrones

Lista desordenada

5. Tipos de eventos

Eventos de arrastrar y soltar

```
 <h2>Ordena la lista según tus preferencias</h2>
    <ul id="origen">
        <li class="drag" draggable="true" id="origen1" ondragstart="drag(event)" >Pizza</li>
        <li class="drag" draggable="true" id="origen2" ondragstart="drag(event)" >Hamburguesa</li>
        <li class="drag" draggable="true" id="origen3" ondragstart="drag(event)" >Macarrones</li>
    </ul>
    <h2>Lista desordenada</h2>
    <div id="destino" ondrop="drop(event)" ondragover="allowDrop(event)">
```

```
function allowDrop(ev) {
    ev.preventDefault();
}

function drag(ev) {
  ev.dataTransfer.setData("idTransitorio", ev.target.id);
}

function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("idTransitorio");
  ev.target.appendChild(document.getElementById(data));
}
```

```
  <style>
        #destino {
          width: 350px;
          height: 70px;
          padding: 10px;
          border: 1px solid #aaaaaa;
          margin: 3%;
        }
    </style>
```

6. Eventos de reproducción multimedia

Evento	Momento en el que se activa
canplay	Cuando se detecta que el contenido se puede comenzar a reproducir.
canplaythrough	Cuando se estima que el contenido tiene cargados datos suficientes como para poder reproducirse.
durationchange	Cuando se modifica el atributo **duration** del medio.
emptied	Cuando se vacía de datos el medio.
ended	Cuando se detiene la reproducción, sin intervención del usuario.
loadeddata	Cuando se ha cargado la primera imagen de un vídeo.
loadedmetadata	Cuando se han cargado los metadatos del medio.
pause	Cuando se pausa el medio de reproducción.
play	Cuando se reanuda la reproducción tras una pausa.
playing	Cuando el medio está listo para reproducirse tras una parada.
ratechange	Cuando se modifica el **rate** del vídeo en reproducción.
seeked	Cuando finaliza la búsqueda en el medio.
seeking	Cuando se inicia la búsqueda en el medio.
stalled	Cuando se produce un fallo en la carga, pero la carga continúa.
suspend	Cuando se suspende la carga del medio.
timeupdate	Cuando se modifica el valor de **currentTime** del medio.
volumechange	Cuando se modifica el volumen.
waiting	Cuando la reproducción se detiene por falta de datos.

7. Otros eventos

Evento	Momento en el que se activa
Carga de elementos	
abort	Cuando se cancela la carga de un elemento.
DOMContentLoaded	Cuando se ha cargado el documento HTML.
error	Cuando se produce un error en la carga.
load	Cuando se ha cargado el documento y los elementos externos que incorpora: imágenes, hojas de estilo…
progress	Cuando se está produciendo la carga.
readystatechange	Cuando se modifica el valor del atributo **readystate.**
Historial de la sesión	
popstate	Cuando se cambia el historial.
pagehide	Cuando se esconde la página actual mientras se visualizan las distintas páginas de la sesión.
pageshow	Cuando el navegador muestra el documento en la ventana.
Ventana	
scroll	Cuando se desplaza la ventana por medio de las barras de desplazamiento.
resize	Cuando se modifica el tamaño de la ventana.

Evento	Momento en el que se activa
Animación	
animationend	Cuando finaliza la animación.
animationiteration	Cuando se repite la animación.
animationstart	Cuando comienza la animación.
Impresión	
afterprint	Cuando ha comenzado la impresión o se ha cerrado la previsualización de la impresión.
beforeprint	Cuando va a comenzar la impresión o se ha abierto la previsualización de la impresión.
Portapapeles	
copy	Cuando se va a copiar justo antes de ejecutar la acción.
cut	Cuando se va a cortar justo antes de ejecutar la acción.
paste	Cuando se va a pegar justo antes de ejecutar la acción.

8. Expresiones regulares

Las **expresiones regulares** son un sistema para buscar, capturar o reemplazar texto utilizando patrones.

Estos patrones permiten realizar una búsqueda de texto de una forma relativamente sencilla y abstracta, de forma que abarca una gran cantidad de posibilidades que de otra forma sería imposible o muy extensa y compleja.

Estos patrones se representan mediante una cadena de texto, donde ciertos símbolos tienen un significado especial.

Ejemplo

Detectar si un nombre empieza con las letras «p» o «s» y además termina con las letras «o» o «a».

```
<h2>Lista de nombres</h2>
<ul id="nombres">

</ul>
```

```
document.addEventListener("DOMContentLoaded", init)

function init(){
    var lista = document.getElementById("nombres")
    var names = ["Pedro", "Sara", "Miriam", "Nestor", "Adrián", "Sandro"];
    names.forEach(function(name) {
        const regex = /^(p|s).+(o|a)$/i;
        if (regex.test(name)) {
            var elemento = document.createElement("li");
            elemento.textContent=`El nombre ${name} cumple las restricciones.`;
            lista.appendChild(elemento);
        }
    });
}
```

8. Expresiones regulares

```
/^(p|s).+(o|a)$/i;
```

Búsqueda

^: Esto representa el principio de una cadena. La expresión regular buscará una coincidencia al comienzo de la cadena.

(p|s): Este es un grupo de captura (encerrado en paréntesis) que busca una coincidencia con "p" o "s". Significa que la cadena debe comenzar con "p" o "s".

.+: Esto coincide con uno o más caracteres (cualquier carácter excepto una nueva línea). El punto (.) coincide con cualquier carácter, y el signo más (+) indica que debe haber al menos uno de esos caracteres. Esto permite que haya cualquier cantidad de caracteres entre el primer grupo de captura y el segundo grupo de captura.

(o|a): Otro grupo de captura que busca una coincidencia con "o" o "a". Esto significa que la cadena debe terminar con "o" o "a".

$: Esto representa el final de la cadena. La expresión regular buscará una coincidencia al final de la cadena.

i: La "i" al final de la expresión regular indica que la búsqueda debe ser insensible a mayúsculas y minúsculas, lo que significa que coincidirá con letras mayúsculas o minúsculas.

8. Expresiones regulares

¿Cuándo usar expresiones regulares?

Por norma general, siempre se debería intentar resolver los problemas sin utilizar expresiones regulares, salvo en casos particulares donde el uso de la expresión regular nos proporcione una ventaja muy clara.

Por ejemplo, situaciones donde hay que controlar tantos casos, que sin expresión regular el código sería exageradamente extenso.

8. Cómo crear expresiones regulares

En Javascript podemos crear una expresión regular de varias formas:

```
// Notación literal (forma preferida)
const regexp1 = /.a.o/i;

// Notación de objeto
const regexp2 = new RegExp(".a.o", "i");
const regexp3 = new RegExp(/.a.o/, "i");
```

- Las barras **/** (azul) son los delimitadores de las partes de una expresión regular.
- La definición de la expresión regular (verde) es un texto con símbolos especiales que indica que textos va a incluir.
- Los flags (amarillo), son una serie de caracteres que indican cómo funcionará la expresión regular.

9.1. Propiedades de las expresiones

- Cada expresión regular creada, tiene unas propiedades definidas, donde podemos consultar ciertas características de la expresión regular.
- Además, también tiene unas propiedades de comprobación para saber si un flag determinado está activo o no.

Propiedades	Descripción
STRING .source	Devuelve la **expresión regular** original definida (sin los flags).
STRING .flags	Devuelve los **flags** activados en la expresión regular.
NUMBER .lastIndex	Devuelve la posición donde detectó una ocurrencia en la última búsqueda. (Ver más adelante)

Busca cualquier texto que contenga una "M" seguida de dos carácteres cualquiera, seguida de una "z".

```
const regexp = /M..z/ig;

regexp.source;      // "M..z"
regexp.flags;       // "ig"
```

9.2. Flags de las expresiones

- Los flags son uno o varios caracteres especiales que se escriben en un tras la segunda barra / delimitadora de una expresión regular, o en el segundo parámetro del new RegExp():

```
const regexp1 = /reg/;                    // No tiene ningún flag activado
const regexp2 = /reg/i;                   // Tiene el flag "i" activado
const regexp3 = new RegExp(/reg/, "gi");  // Tiene el flag "g" activado
```

Propiedades	Flag	Descripción
BOOLEAN .global	g	Búsqueda global. Permite seguir buscando coincidencias en lugar de pararse al encontrar una.
BOOLEAN .ignoreCase	i	Le da igual mayúsculas y minúsculas. Se suele denominar **insensible a mayús/minús**.
BOOLEAN .multiline	m	Multilínea. Permite a ^ y $ tratar los finales de línea \r o \n.

9. Flags de las expresiones

- Los flags son uno o varios caracteres especiales que se escriben en un tras la segunda barra / delimitadora de una expresión regular, o en el segundo parámetro del new RegExp():

```
const regexp5 = /manz/gi;

regexp5.global;                 // true
regexp5.flags.includes("g");    // equivalente al anterior
regexp5.ignoreCase;             // true
regexp5.flags.includes("i");    // equivalente al anterior
regexp5.multiline;              // false
regexp5.flags.includes("m");    // equivalente al anterior
```

10. Métodos de las expresiones

El método .test() siempre te devolverá un para indicar si la expresión regular ha encontrado un texto que encaja con el patrón definido.

```
const regexp = /.a.o/i;

regexp.test("gato");    // true
regexp.test("pato");    // true
regexp.test("perro");   // false
regexp.test("DATO");    // true
```

11. Ayuda para crear expresiones

CLASES BÁSICAS

Caracter especial	Descripción
.	Comodín, significa **cualquier caracter** (letra, número, símbolo...), pero que ocupe sólo **1 carácter**.
\	Precedido de un carácter especial, lo invalida (se llama «escapar»).

CONJUNTO DE CARACTÉRES

Caracter especial	Descripción
[]	Rango de carácteres. Cualquiera de los caracteres del interior de los corchetes.
[^]	Que no exista cualquiera de los caracteres del interior de los corchetes.
|	Establece una alternativa: lo que está a la izquierda o lo que está a la derecha.

11. Ayuda para crear expresiones

RANGO DE CARACTÉRES

Caracter especial	Alternativa	Descripción
[0-9]	\d	Un dígito del 0 al 9.
[^0-9]	\D	No exista un dígito del 0 al 9.
[A-Z]		Letra mayúscula de la A a la Z. Excluye ñ o letras acentuadas.
[a-z]		Letra minúscula de la a a la z. Excluye ñ o letras acentuadas.
[A-Za-z0-9]	\w	Carácter alfanumérico (letra mayúscula, minúscula o dígito).
[^A-Za-z0-9]	\W	No exista carácter alfanumérico (letra mayúscula, minúscula o dígito).
[\t\r\n\f]	\s	Carácter de espacio en blanco (espacio, TAB, CR, LF o FF).
[^ \t\r\n\f]	\S	No exista carácter de espacio en blanco (espacio, TAB, CR, LF o FF).
	\xN	Carácter hexadecimal número N.
	\uN	Carácter Unicode número N.

11. Ayuda para crear expresiones

ANCLAS

Caracter especial	Descripción
^	Ancla. Delimita el inicio del patrón. Significa **empieza por**.
$	Ancla. Delimita el final del patrón. Significa **acaba en**.
\b	Límite de una palabra separada por espacios, puntuación o inicio/final.
\B	Opuesta al anterior. Posición entre 2 caracteres alfanuméricos o no alfanuméricos.

CUANTIFICADORES

Caracter especial	Descripción
*	El carácter anterior puede aparecer 0 o más veces.
+	El carácter anterior puede aparecer 1 o más veces.
?	El carácter anterior puede aparecer o no (es opcional).
{n}	El carácter anterior aparece n veces.
{n,}	El carácter anterior aparece n o más veces.
{n,m}	El carácter anterior aparece de n a m veces.

12. Otras utilidades

- El objeto **Date()** es una utilidad que proporciona JavaScript para crear fechas y horas.
- Una vez creado un objeto de tipo fecha, es posible manipularlo para obtener información o realizar cálculos con las fechas.

```
var fechaHora = new Date();
document.getElementById("reloj").innerHTML = fechaHora;
```

- La función **setTimeout()** permite ejecutar una función una vez que haya transcurrido un periodo de tiempo indicado.

```
function init(){
    setTimeout(reloj, 1000);
}

function reloj(){
    var fechaHora = new Date();
    document.getElementById("reloj").innerHTML =
fechaHora;
}
```

- Para ejecutar una función de forma periódica, se utiliza **setInterval().**

```
function init(){
    //Cambia la hora continuamente
    setInterval(reloj, 1000);
}

function reloj(){
    var fechaHora = new Date();
    document.getElementById("reloj").innerHTML =
fechaHora;
}
```

¿ A L G U N A D U D A ?

G R A C I A S