Guía de Java Script

Índice

1. [Seleccionando elementos](#seleccionandoelementos)
2. [document.getElementById()](#elemetbyid)
3. [querySelector()](#queryselector)
4. [Manipulando elementos](#manipulandoelementos)
5. [document.createElement()](#createelement)
6. [appendChild()](#appendchild)

1. [textContent](#textcontent)
2. [innerHTML](#innerhtml)
3. [append()](#append)
4. [push()](#push)
5. [Manipulando los estilos de un elemento](#manipulandolosestilosdeunelemento)
6. [style property – propiedad estilo](#styleproperty)
7. [Trabajando con eventos](#trabajandoconeventos)
8. [Introducción a los eventos](#introduccionaloseventos)
9. [Manejando eventos](#manejandoeventos)
10. [Eventos de ratón](#eventosraton)
11. [Eventos de teclado](#eventosteclado)
12. [Eventos de focos (focus)](#focos)
13. Métodos con Strings
14. length()
15. Extraer elementos de un String(slice, substring y substr)
16. replace()
17. replaceAll()
18. toUpperCase()
19. toLowerCase()
20. trim()
21. Extraer caracteres de un String (charAt, charCodeAt, propiedades)
22. split()
23. Métodos con Arrays
24. length()
25. toString()
26. pop()
27. push()
28. shift()
29. unshift()
30. flat()
31. splice()
32. slice()
33. join()
34. delete()
35. concat()

**Seleccionando elementos**

**document.getElementById()**

El elemento document.getElementById() devuelve un Elemento que representa un elemento HTML con una id que coincida con el string.

Si el elemento está vacío, devuelve null

Es el más rápido porque tira de id

A diferencia del método querySelector(), solo esta disponible en el objeto document, no en otros elementos

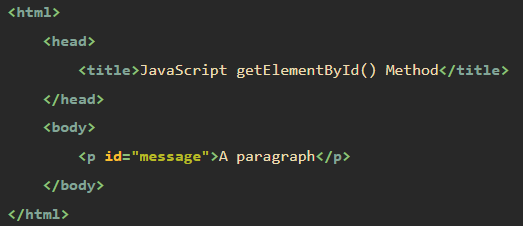
Sintaxis:



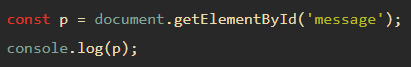
El id es un string que representa la id del elemento a seleccionar, La id es sensible a mayúsculas. Por ejemplo ‘root’ y ‘Root’ son totalmente diferentes.

La id es única dentro de un documento HTML. Sin embargo, HTML es un lenguaje indulgente. Si el documento HTML tiene varios elementos con la misma id, el método document.getElementById() devuelve el primer elemento que encuentra.

Ejemplo



Aquí hay un párrafo p con una id llamada message



Resultado:



Después de seleccionar un elemento, puede agregar estilos al elemento, manipular sus atributos y recorrer los elementos principales y secundarios.

**querySelector()**

El querySelector() es un método de la interfaz Element. El método querySelector() te permite seleccionar el primer elemento que coincida con uno o mas selectores CSS

Sintaxis:



Si el selector no es una sintaxis CSS válida, el método devolverá una excepción de SyntaxError

Si ningún elemento coincide con los selectores CSS, el querySelector() devolverá null

Además de querySelector() puedes usar el método querySelectorAll() para seleccionar todos los elementos que coinciden con un selector CSS o un grupo de selectores CSS

Sintaxis:



El método querySelectorAll() devuelve una NodeList estática de elementos que coinciden con el selector de CSS. Si ningún elemento coincide, devuelve una NodeList vacía.

Tenga en cuenta que NodeList es un objeto similar a un array, no un objeto de array. Sin embargo, en los navegadores web modernos, puedse usar el método forEach() o el bucle for...of.

Selectores de tipo

Para seleccionar elementos por nombre de nodo, utilice el selector de tipo, por ejemplo, a selecciona todos los elementos <a>.

Ejemplo: Encuentra el primer elemento h1 en el documento (document)



Ejemplo: Encuentra todos los elementos h2



Selectores de clase

Para encontrar un elemento de una clase CSS. Se utiliza un punto ‘.’ para identificar que es una clase.

Ejemplo: Encuentra el primer elemento con la clase menu-item



Ejemplo: Encuentra todos los elementos con la clase menu-item



Selector de ID

Para seleccionar un elemento basado en el valor de su id

Sintaxis:



Ejemplo: Encuentra el primer elemento con la id #logo

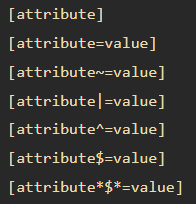


Mientras que la id debe ser única en el documento, el querySelectorAll() no es relevante.

Selector de atributos

Para seleccionar elementos a los que se les han asignado atributos

Sintaxis:



Ejemplo: Encuentra el primer elemento con el atributo [autoplay] con cualquier valor



Ejemplo: Encuentra todos los elementos que tienen el atributo [autplay] con cualquier valor



Selector de grupos

Ejemplo:



**Manipulando elementos**

**document.createElement()**

Para crear un elemento HTML, se usa el método document.createElement()

Sintaxis:



El document.createElement() acepta un nombre de etiqueta HTML y devuelve un Nodo con el tipo Element

Crear un div

Suponiendo que el código es este:



Ejemplo: usar el document.createElement() para crear un elemento <div>



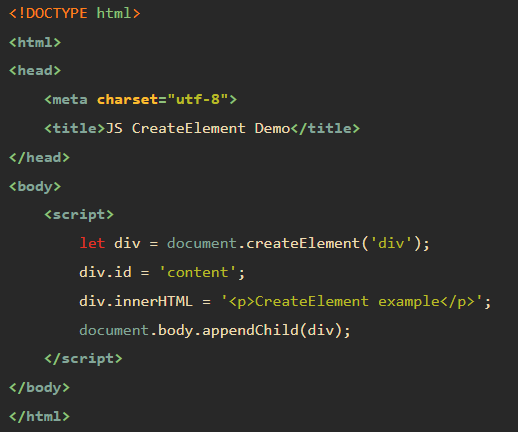
Y agregar un fragmento de código HTML al <div>



Para adjuntar el div al documento, utiliza el método appendChild():



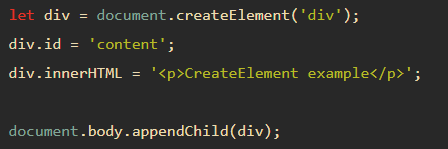
Resultado:



Añadiendo un id al div

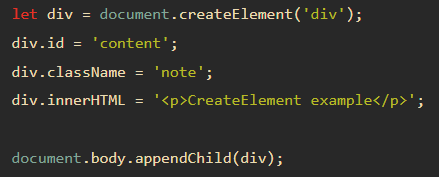
Si quieres añadir una id al div, puedes establecer el atributo id al elemento a un valor.

Ejemplo:



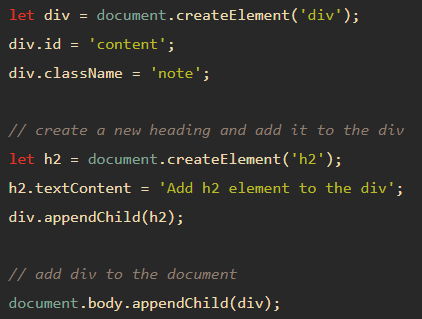
Añadiendo una clase al div

Ejemplo:



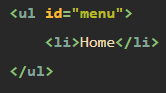
Añadiendo texto a un div

Para añadir un elemento a un div, crea un elemento y lo agregalo al div usando el método appendChild():

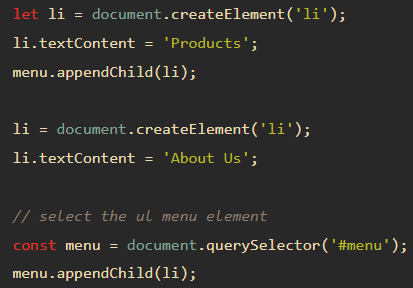


Creando una nueva lista de elementos

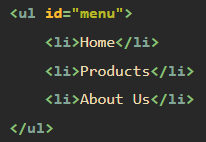
Digamos que tenemos una lista de elementos:



El código añade dos elementos li al ul



Salida o resultado:



**appendChild()**

Es un método de la interfaz Node. El método appendChild() permite añadir un nodo al final de la lista de nodos hijo de un nodo padre especifico.

Sintaxis:

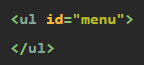


Nodo padre.nodo hijo

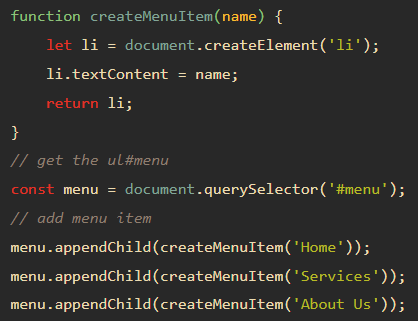
En este método, el childNote es el nodo para anexar al nodo padre dado. El appendChild() devuelve el hijo anexado

Si el childNode es una referencia a un nodo existente en el documento, el método appendChild() mueve el childNode desde su posición actual a la nueva

Según el código.



El siguiente ejemplo usar el método appendChild() para añadir 3 items de lista al elemento <ul>:



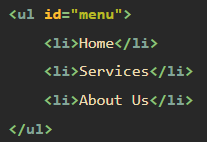
Explicación:

Primero, la función createMenuItem() crea una nuevo item de la lista con un nombre especificado usando el método createElement()

Segundo, selecciona el elemento <ul> con la id menú usando el método querySelector()

Tercero, llama la función createMenuItem() para crear un nuevo item de menú y usa el método appendChild() para anexar el item de menú al elemento <ul>

Salida:



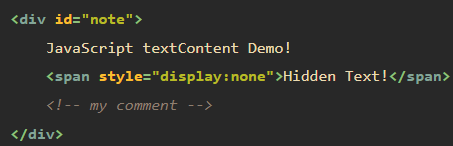
**textContent**

Obtiene el contenido del texto de un nodo y sus descendientes.

Sintaxis:



Suponiendo que el siguiente HTML es este:



El ejemplo siguiente usar la propiedad textContent para obtener el texto del elemento <div>:



Explicación:

1. Primero, selecciona el elemento div con la id note usando el elemento getElementById()
2. Después, muestra el texto del nodo accediendo a la propiedad textContent

Salida:



textContent vs innerText

Por otro lado, el textContent tiene en cuenta el estilo CSS y devuelve solo texto legible por humanos. Por ejemplo:



Salida:



El texto oculto y los comentarios no son devueltos.

La propiedad innerText usa el CSS actualizado para calcular el texto, acceder a él desencadenará un reflujo, lo que es computacionalmente costoso.

**innerHTML**

Es una propiedad de Element que te permite obtener o establecer el marcado HTML (DESTRUYE EL DOCUMENTO).

Sintaxis:



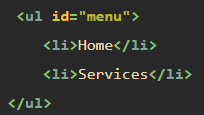
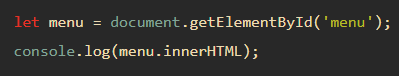
Leyendo la propiedad innerHTML de un elemento

Sintaxis:



Cuando lees el innerHTML de un elemento, el navegador web debe serializar el fragmento HTML de los descendientes del elemento.

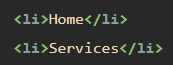
Ejemplo: Usa la propiedad innerHTML para obtener el contenido del elemento lista <u>

Como funciona:

1. Primero, selecciona el elemento <ul> por su id menu usando el metodo getElementByID().
2. Después, obtiene el contenido HTML de <ul> usando el innerHTML

Salida:



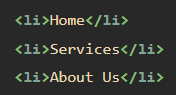
La propiedad innerHTML devuelve el código HTML actual del documento, incluyendo todos los cambios que se han hecho desde que la pagina ha sido cargada.

Ejemplo:



Comentario: Crea un dato de tipo let con el valor de la lista ul con la id menú. Crea un dato de tipo let llamada “li” y utiliza el método document.createElement() para crear un nuevo elemento, en este caso un <li> nuevo en la lista.

Salida:



Explicación:

Primero obtiene el elemento <ul> con el menú id usando el método getElementById().

En segundo lugar, crea un nuevo elemento <li> y lo agrega al elemento <ul> mediante los métodos createElement() y appendChild().

Tercero, obtiene el HTML del elemento <ul> usando la propiedad innerHTML del elemento <ul>. El contenido del elemento <ul> incluye el contenido inicial y el contenido dinámico creado dinámicamente por JavaScript.

Estableciendo la propiedad innerHTML en un elemento

Sintaxis:



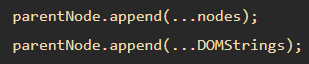
Reemplazará el contenido existente de un elemento con el nuevo contenido.

Ejemplo: Puedes borrar el contenido entero de un documento haciendo esto:



**append()**

El método nodoPadre.append() inserta un set de objetos Nodo u objetos DOMString después del último hijo de un nodo padre.

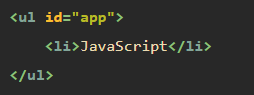


El método append() insertará objetos DOMString como nodos de texto

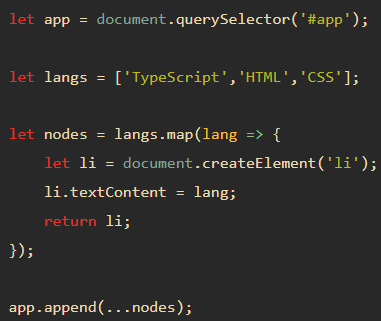
El método append() no tiene devuelve ningún valor. Significa que append() implícitamente devuelve undefined.

Usando el método append() para anexar un elemento

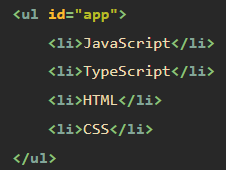
Según el elemento ul:



El siguiente ejemplo muestra como crear una lista de elementos li y los anexa al elemento ul:



Salida HTML:



Explicación:

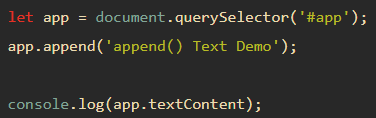
1. Primero, elige el elemento ul por su id usando el método querySelector().
2. Segundo, declara un array de lenguajes
3. Tercero, por cada lenguaje, crea un nuevo elemento li con el textContent asignado a cada lenguaje
4. Finalmente, anexa los elementos del li al elemento ul usando el método append()

Usando el método append() para anexar texto a un elemento

Suponiendo que el código HTML es este:



Se puede usar el método append() para anexar un texto a un elemento



Salida:



append() vs appendChild()

|  |  |  |
| --- | --- | --- |
| Diferencias | append() | appendChild() |
| Devolver valor (return) | undefined | El objeto Nodo anexado |
| Input | Multiples objetos Nodo | Un sólo objeto nodo |
| Tipos de parámetro | Acepta Nodo y DOMString | Solo nodo |

**push()**

Se utiliza para añadir los elementoa de un array. Sintaxis:

arrayOriginal.push(elementosNuevos)

**Manipulando los estilos de un elemento**

**style property – propiedad estilo**

Para establecer el estilo de un elemento, se utiliza la propiedad style de ese elemento:



La propiedad style devuelve el objeto CSSStyleDeclaration de solo lectura que contiene una lista de propiedades CSS. Por ejemplo, para establecer el color de un elemento en rojo, se haría asi:



También se puede hacer así, al igual que se haría en CSS:



O con el método setAtribute()



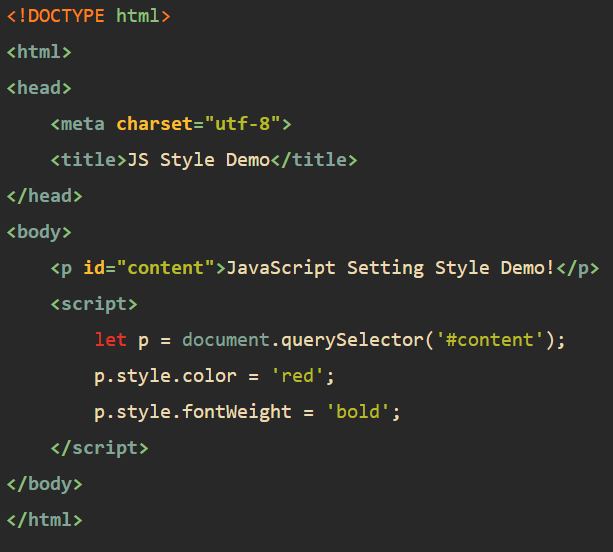
Una vez que se configure el estilo, se puede modificar una o más propiedades de CSS:



Si no quieres sobrescribir las propiedades CSS existentes, puedes concatenar la nueva propiedad CSS al cssText:



El siguiente ejemplo usa el objeto style para establecer las propiedades CSS de un párrafo con el id content:



Explicación:

1. Primero selecciona el elemento p cuya id es content usando el método querySelector()
2. Luego, establecer las propiedades color y peso de fuente del párrafo con las propiedades color y fontWeight del objeto style

Para obtener todos los estilos aplicados en un elemento, usa el método window.getComputedStyle()

**Trabajando con eventos**

**Introducción a los eventos**

Un evento es una acción que ocurre en el navegador web, que el navegador web le informa para que pueda responder. Por ejemplo, cuando los usuarios hacen clic en un botón de una página web, es posible que desee responder a este evento de click mostrando un cuadro de diálogo.

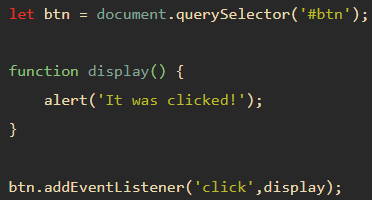
Cada evento puede tener un controlador de eventos que es un bloque de código que se ejecutará cuando ocurra el evento.

Un controlador de eventos también se conoce como detector de eventos. Escucha el evento y se ejecuta cuando ocurre el evento.

Supongamos que tiene un botón con el id btn:



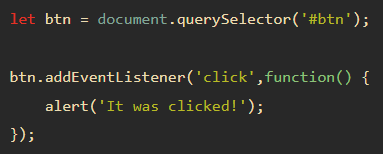
Para definir el código que se ejecutará cuando se haga clic en el botón, debe registrar un controlador de eventos mediante el método addEventListener():



Explicación:

1. Primero, selecciona el botón con la id btn usando el método querySelector()
2. Segundo, define una función llamada display() como un controlador de eventos
3. Finalmente, registra un controlador de eventos usando el addEventListener(), para que los usuarios cuando hagan clic en el botón, la función display() sea ejecutada

Una forma corta de registrar un controlador de eventos es poner todo el código en una función anónima, como esto:



Flujo del evento

Asumiendo que tenemos el siguiente documento HTML:



Cuando haces clic en el botón, no solo estas clicando el botón, también su contenedor div y la página entera.

El flujo del evento explica el orden en el que los eventos son recibidos en la página desde el elemento donde el evento ocurre y se propaga por el árbol DOM.

Hay dos modelos de eventos principales. Difusión de eventos (event bubbling) y captura de eventos (event capturing).

Difusión de eventos

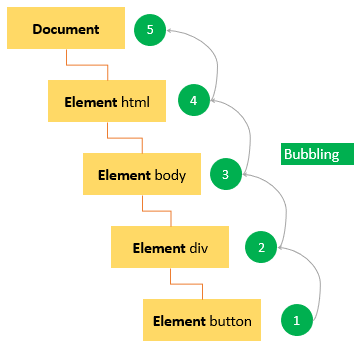
En el modelo de difusión de eventos, un evento comienza en el elemento más específico y luego fluye hacia arriba, hacia el elemento menos específico (el document o incluso la ventana (window)).

Cuando haces clic en el botón, el evento click ocurre en el siguiente orden:

1. Button
2. div con el contenedor id
3. body
4. html
5. document

El evento click ocurre primero en el botón que es el elemento en el que se hizo clic. Luego, el evento click sube en el árbol DOM y se dispara en cada nodo a lo largo de su camino hasta que alcanza el objeto document.

La siguiente imagen muestra el efecto del Event bubbling cuando el usuario pulsa el botón:

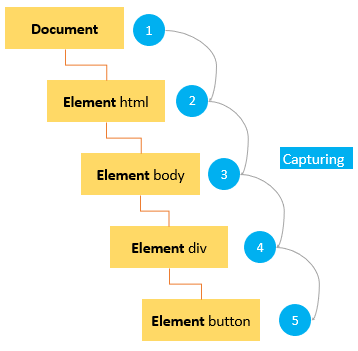


Captura de eventos

En el modelo de captura de eventos, un evento comienza en el elemento menos específico y fluye hacia abajo, hacia el elemento más específico. Cuando se hace clic en el botón, el evento click ocurre en el siguiente orden:

1. document
2. html
3. body
4. div con el contenedor id

La siguiente imagen ilustra el efecto de captura de eventos:



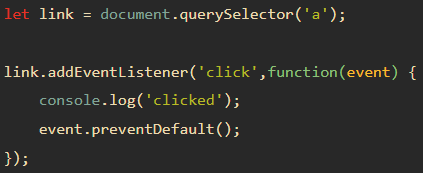
preventDefault()

Para prevenir el comportamiento por defecto de un evento, se usa el método preventDefault().

Por ejemplo, cuando presionas un link, el navegador te lleva a la URL especificada en el atributo href:



Sin embargo, puedes prevenir este comportamiento usando el método preventDefault() del objeto event



El método preventDefault() no detiene el evento de una difusión de estos. Un evento puede ser cancelado cuando la propiedad cancelable es true

**Manejando eventos**

Cuando ocurre un evento, puedes crear un controlador de eventos. Es un fragmento de código que se ejecutará para responder a ese evento. Un controlador de eventos también se conoce como **detector de eventos**. Escucha el evento y responde en consecuencia a los disparos del evento.

Un escuchador de eventos es una función con un **nombre explícito si es reutilizable** o una **función anónima en caso de que se use una sola vez**.

Un evento puede ser manejado por uno o varios controladores de eventos. Si un evento tiene varios controladores de eventos, todos los controladores de eventos se ejecutarán cuando se active el evento.

Existen 3 maneras para asignar controladores de eventos:

Atributos del controlador de eventos HTML

Los controladores de eventos normalmente tienen nombres que empiezan con on. Por ejemplo el controlador de eventos para el evento click es onclick.

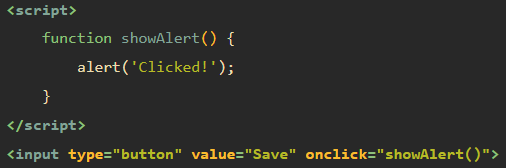
Para asignar un controlador de eventos a un evento asociado con el elemento HTML, puedes usar un atributo HTML con el nombre del controlador del evento. Por ejemplo, para ejecutar un código cuando un botón es presionado, se usa el siguiente código:



En este caso, cuando el botón es presionado, la alerta se mostrará

Cuando asignas código JavaScript como el valor de un atributo onclick, necesitas “escapar” de los caracteres HTML como los ampersand (&), comillas dobles (“), menor que (<), etc., de lo contrario tendrás un error de sintaxis.

Un controlador de evento definido en el HTML puede llamar una función asignada en un script. Por ejemplo:



En este ejemplo. El botón llama la función showAlert() cuando es presionada.

La función showAlert() es una función definida en un elemento <script> separado, y puede ser colocado en un archivo externo de JavaScript

Importante

1. El código en un controlador de evento puede acceder al objeto event sin definirlo explícitamente



1. El valor this dentro del controlador de evento es equivalente al elemento target del evento



1. El controlador de evento puede acceder a las propiedades del elemento



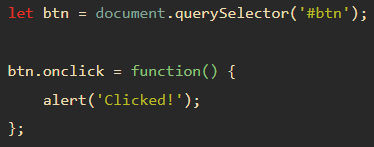
Desventajas de usar atributos del controlador de eventos HTML

La asignación de controladores de eventos utilizando atributos de controlador de eventos HTML se considera una mala práctica y debe evitarse en la medida de lo posible por las siguientes razones:

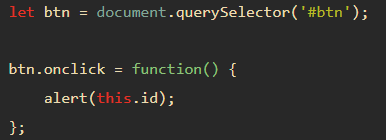
1. En primer lugar, el código del controlador de eventos se mezcla con el código HTML, lo que hará que el código sea más difícil de mantener y ampliar.
2. En segundo lugar, es un problema de tiempo. Si el elemento se carga por completo antes que el código JavaScript, los usuarios pueden comenzar a interactuar con el elemento en la página web, lo que provocará un error.

Controladores de eventos DOM de nivel 0

Cada elemento tiene propiedades de controlador de eventos como onclick. Para asignar un controlador de eventos, se establece la propiedad en una función como se muestra en el ejemplo:



En este caso, la función anónima se convierte en el método de un elemento button. Por lo tanto, el valor this es equivalente al elemento. Se puede acceder a las propiedades del elemento dentro del controlador de eventos:



Salida:



Usando el valor this dentro del controlador de eventos, te permite acceder a las propiedades del elemento y métodos.

Para eliminar un controlador de eventos, estableces el valor de la propiedad del controlador del elemento en null.



Controladores de eventos DOM de nivel 2

Los controladores de eventos DOM de nivel 2 ofrecen 2 métodos principales para lidiar con el registro y “desregistro” de escuchadores de eventos:

* addEventListener()
* removeEventListener()

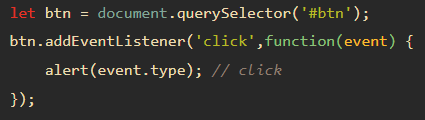
Estos métodos están disponibles en todos los nodos del DOM.

Método addEventListener()

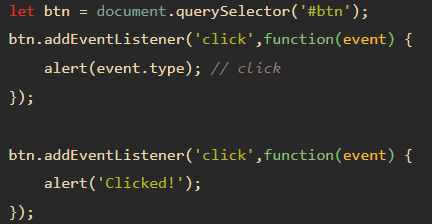
El método addEventListener() acepta tres argumentos:

* Un **nombre de evento**
* Una **función de controlador de eventos**
* Un **valor booleano** que indica al método que llame al controlador de eventos durante la fase de captura (true) o durante la fase de difusión (burbuja) (false).

Por ejemplo:

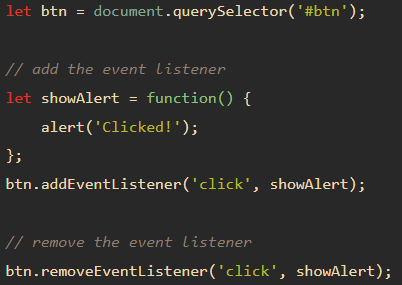


Es posible añadir múltiples controladores de eventos para controlar un solo evento, como esto:

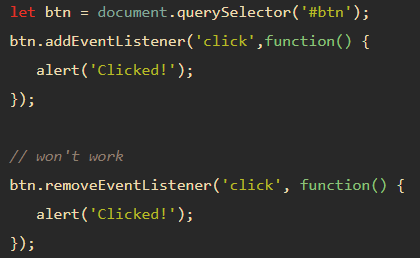


El método removeEventListener()

Elimina el escuchador de un evento que fue añadido por el addEventListener(). Sin embargo, necesitas pasar los mismos argumentos que han sido pasados al addEventListener(). Por ejemplo:



Usando una escuchador de eventos anónimo como el siguiente no funcionará:



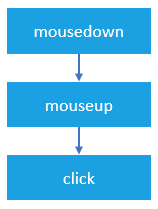
**Eventos de ratón**

Los eventos del ratón se activan cuando usa el ratón para interactuar con los elementos de la página. Los eventos DOM de nivel 3 definen nueve eventos de mouse:

mousedown, mouseup y click

Cuando haces click en un elemento, no hay mas que tres elementos en la siguiente secuencia:

1. El mousedown se activa cuando presionas el botón del ratón de un elemento
2. El mouseup se activa cuando suelta el botón del ratón de un elemento
3. El click se activa cuando mousedown y mouseup es detectado en un elemento



doblclick

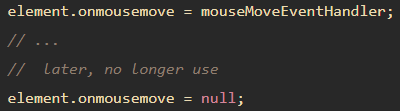
En la práctica, raramente se usa el evento dblclick. El evento dblclick se activa cuando haces doble clic en un elemento.

Se utilizan dos eventos de clic para hacer que un evento de dblclick se active. El evento dblclick tiene 4 eventos activados en el siguiente orden:

1. mousedown
2. mouseup
3. click
4. mousedown
5. mouseup
6. click
7. dblclick

mousemove

El evento mousemove se activa repetidamente cuando mueve el cursor del ratón alrededor de un elemento. Incluso cuando mueve el mouse un píxel, el evento mousemove aún se activa. Hará que la página sea lenta, por lo tanto, solo registra el controlador de eventos mousemove solo cuando lo necesita e inmediatamente elimina el controlador de eventos tan pronto como ya no se usa, así:



mouseover/mouseout

El mouseover se activa cuando el cursor del ratón está fuera del elemento y luego se mueve dentro de los límites del elemento.

El mouseout se dispara cuando el cursor del ratón está sobre un elemento y luego mueve otro elemento.

mouseenter / mouseleave

El mouseenter se dispara cuando el cursor del ratón está fuera de un elemento y luego se mueve dentro de los límites del elemento.

El mouseleave se dispara cuando el cursor del ratón está sobre un elemento y luego se mueve hacia el exterior de los límites del elemento.

Tanto mouseenter como mouseleave no burbujean ni se disparan cuando el cursor del mouse se mueve sobre elementos descendientes.

Registrando controladores de eventos de ratón

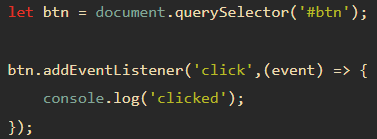
Para registrar un evento de ratón, se usan estos pasos:

1. Primero, selecciona el elemento usando el querySelector() o getElementById().
2. Luego, registra el evento de ratón usando el método addEventListener()

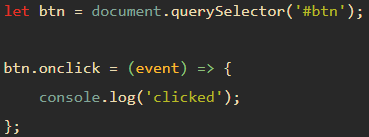
Por ejemplo, supongamos que está el siguiente botón:



Para registrar un controlador de eventos de clic del ratón, se usa el siguiente código:



O también se puede asginar un controlador de eventos de ratón a la propiedad del elemento:



Es una buena practica usar siempre el addEventListener() para registrar un controlador de evento de ratón

Obteniendo las coordenadas de la pantalla

Las propiedades screenX y screenY del evento pasado al controlador de eventos de ratón devuelven las coordenadas de la pantalla de la localización del ratón en relación a la pantalla completa

Por otra parte, las propiedades clientX y clientY ofrecen las coordenadas horizontal y vertical con el área del cliente de la aplicación donde el evento del ratón este sucediendo

**Eventos de teclado**

Los 3 ventos de teclado principales son los siguientes:

* keydown: Se activa cuando presionas una tecla del teclado y se activa repetidamente mientras se esté presionando dicha tecla
* keyup: Se activa cuando sueltas una tecla del teclado
* keypress: Se activa cuando presionas un carácter del teclado como a, b, o c, no la tecla flecha izquierda, home o FIN del teclado. El keypress también se activa repetidamente cuando mantienes pulsada la tecla del teclado.

Los eventos de teclado normalmente se activan en el cuadro de texto, aunque todos los elementos lo admiten.

Cuando presionas una tecla carácter en el teclado, 3 eventos de teclado son activados en el siguiente orden:

1. keydown
2. keypress
3. keyup

Los eventos keydown y keypress se activan antes de que se realice cualquier cambio en el cuadro de texto, mientras que el evento keyup se activa después de que se hayan realizado los cambios en el cuadro de texto. Si mantiene presionada una tecla de carácter, la pulsación de tecla y la pulsación de tecla se activan repetidamente hasta que suelte la tecla.

Cuando presionas una tecla que no es de carácter, primero se activa el evento keydown, seguido del evento keyup. Si mantiene presionada la tecla que no es de carácter, la pulsación de tecla se activa repetidamente hasta que suelte la tecla.

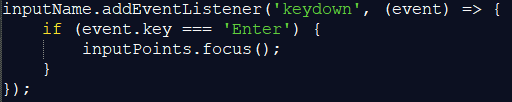
Manejando los eventos de teclado

Para manejar un evento de teclado, se deben seguir los siguientes pasos:

1. Primero, seleccionar el elemento en el que se activará el evento del teclado. Por lo general, es un cuadro de texto.
2. Luego, usar el element.addEventListener() para registrar un controlador de evento

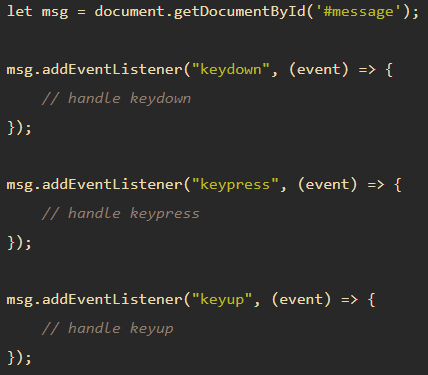
Suponiendo que tenemos el siguiente cuadro de texto con el id message:





Explicación: Al pulsar una tecla, después definida en la condición de abajo, hace el focus del input de puntos del ejercicio del simulacro de examen

Aquí se ilustra como registrar eventos de escucha de teclado



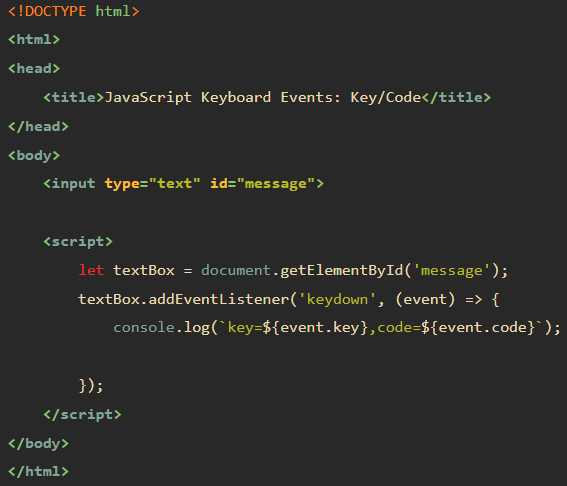
Si presionas una tecla carácter, todos los 3 controladores de eventos serán llamados.

Propiedades de un evento de teclado

El evento de teclado tiene dos importantes propiedades: key y code. La propiedad key devuelve el carácter que ha sido presionado donde la propiedad code devuelve el código de la tecla física.

Por ejemplo, si presionas el carácter z, el event.key devuelve z y event.code devuelve KeyZ.

Según el siguiente ejemplo:



Si escribes el carácter z, veras el siguiente mensaje:



Explicación:

1. Primero, selecciona el campo de texto con el id message usando el metodo getElementById().
2. Después, registra un evento de escucha keydown y registra la tecla y el código de la tecla que ha sido presionada

**Eventos de focos (focus)**

Los eventos focus se activan cuando un elemento recibe o pierde foco. Hay dos eventos de foco principales:

* Focus: Se activa cuando un elemento ha recibido focus
* Blur: Se activa cuando un elemento ha perdido focus

El focusin y focusout se activan al mismo tiempo que focus y blur, sin embargo se ‘difusionan’ mientras que el focus y blur no lo hacen

Los siguientes elementos son enfocables:

* La ventana se enfoca cuando estás en ella usando Alt + Tab o haciendo clic en ella y pierde el focus cuando la devuelve
* Links cuando usas un ratón o un teclado
* Campos de formulario como inputs de texto cuando usas un ratón o teclado.
* Elementos con tabindex, también cuando usas un teclado o ratón.

Ejemplos de eventos de focos (focus)

El siguiente ejemplo muestra como manejar los eventos focus y blur: Cuando mueves el focus al campo password, el fondo cambia a yellow. Si mueves el ratón al campo username, el fondo cambia a white

**Métodos con Strings**

**length()**

La propiedad length devuelve la longitud de un String

Ejemplo:

Texto

Descripción generada automáticamente

**Extraer elementos de un String**

**slice()**

Extrae una parte de un String y devuelve la parte extraída en un nuevo String. Tiene 2 parámetros. Posición inicial (start position) y la posición final (opcional). Las dos formas siguientes de extrar un String no modificar el valor del String del que se va a extraer.

Ejemplo

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Explicación:

Corta una porción de un string desde la posición 7 a la 13

Resultado: Banana

Se puede omitir el segundo parámetro y cortar el resto del String, ya que es opcional.

Ejemplo:

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Resultado: Banana, Kiwi

Si un parámetro es negativo, la posición se cuenta desde el final del String

Ejemplo:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Resultado: Banana, Kiwi

**substring()**

Similar a slice(). La diferencia es que los valores iniciales y finales menores que 0 son tratados como 0.

Ejemplo:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Resultado: Banana

Si omites el segundo parámetro, cortará el resto del String.

**substr()**

Similar a slice(). La diferencia es que el segundo parámetro especifica la longitud (length) de la parte extraída.

Ejemplo:

Una captura de pantalla de un celular

Descripción generada automáticamente con confianza media

Resultado: Banana

Si omites el segundo parámero, cortará el resto del String. Si el primer parámetro es negativo, la posición comienza desde el final del String.

**replace()**

Reemplaza un valor especificado con otro valor en un String. No modifica el valor del que se está haciendo el replace(). Devuelve un nuevo String aparte. Sólo reemplaza la primera coincidencia. Para reemplazar todas las coincidencias, usar una expresión regular (regExp)

Por defecto, solo cogerá la primera coincidencia. Es case sensitive (sensible a mayúculas), ya que si no coinciden los caracteres, no funncionará.

Ejemplo:

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

Resultado: Please, visit W3Schools!

Para reemplazar todas las coincidencias, se usa una expresión regular (regExp) con la /g (global match, se hace de forma global en todo el texto).

Ejemplo:

Texto

Descripción generada automáticamente

**toUpperCase()**

Pasar a mayúsculas. Lo hace en una variable aparte

Ejemplo:

Imagen que contiene naranja, oscuro, pelota, jugador

Descripción generada automáticamente

Resultado: HELLO, WORLD!

**toLowerCase()**

Pasar a minúsculas. Lo hace en una variable aparte

Ejemplo:



Resultado: hello, world!

**trim()**

Elimina los espacios en blanco de ambos lados de un String. Lo hace en una variable aparte

Ejemplo:



Resultado: Hello, World!

**trimStart()**

Igual que trim(), pero solo elimina los espacios desde el comienzo de un String

**trimEnd()**

Igual que trim(), pero solo elimina los espacios desde el final de un String

**Extraer caracteres de un String**

**charAt()**

Devuelve el carácter de un de un índice especificado (posición) en un String

Ejemplo:

Texto

Descripción generada automáticamente

Resultado: H

**charCodeAt()**

Igual que charAt. Devuelve el unicode de un carácter de un de un índice especificado en un String. El método devuelve un código en UTF-16 (un entero entre 0 y 65535)

Ejemplo:

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

Resultado: 72

**Accediendo desde la propiedad**

Ejemplo:

Texto

Descripción generada automáticamente

Resultado: H

**split()**

Un string puede ser convertido a un Array con split().

Ejemplo:

Un reloj digital en la pantalla

Descripción generada automáticamente con confianza baja

Resultado:

