UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Programador Web Inicial
Front End Developer

JS - Funciones

Construye tu propia función

Aprendizaje activo: construyamos una función.

La función personalizada que vamos a construir se llamará `displayMessage()`. Mostrará un cuadro de mensaje personalizado en una página web y actuará como un reemplazo personalizado para la función de `alert()` incorporada de un navegador.

Escriba lo siguiente en la consola de JavaScript de su navegador, en la página que desee:

```
alert('This is a message');
```

La función `alert` tiene un argumento — el string que se muestra en la alerta. Prueba a variar el string para cambiar el mensaje.

La función `alert` es limitada: puede cambiar el mensaje, pero no puede cambiar de manera sencilla nada más, como el color, icono o cualquier otra cosa.

La función básica

Para empezar, vamos a poner juntos una función básica.

Comience accediendo al archivo function-start.html y haciendo una copia local. Verás que el HTML es simple — el body únicamente tiene un botón.

También hemos proporcionado algunos estilos básicos de CSS para customizar el mensaje y un elemento `<script> (en-US)` vacío para poner nuestro JavaScript dentro.

Luego añade lo siguiente dentro del elemento `<script>`:

```
function displayMessage() {
}
```

Comenzamos con la palabra clave `function`, lo que significa que estamos definiendo

una función. A esto le sigue el nombre que queremos darle a nuestra función, un conjunto de paréntesis y un conjunto de llaves.
Todos los parámetros que queremos darle a nuestra función van dentro de los paréntesis, y el código que se ejecuta cuando llamamos a la función va dentro de las llaves. Finalmente, agregue el siguiente código dentro de las llaves:

```
let html = document.querySelector('html');

let panel = document.createElement('div');
panel.setAttribute('class', 'msgBox');
html.appendChild(panel);

let msg = document.createElement('p');
msg.textContent = 'This is a message box';
panel.appendChild(msg);

let closeBtn = document.createElement('button');
closeBtn.textContent = 'x';
panel.appendChild(closeBtn);

closeBtn.onclick = function() {
  panel.parentNode.removeChild(panel);
}
```

La primera línea usa un función DOM API llamada `document.querySelector()` para seleccionar el elemento `<html>` y guardar una referencia a él en una variable llamada `html`, por lo que podemos hacer cosas para más adelante:

```
let html = document.querySelector('html');
```

La siguiente sección usa otra función del DOM API llamada `Document.createElement()` para crear un elemento `<div>` y guardar una referencia a él en una variable llamada `panel`. Este elemento será el contenedor exterior de nuestro cuadro de mensaje.

Entonces usamos otra función del API DOM llamada `Element.setAttribute()` para configurar un atributo a `class` en nuestro panel con un valor de `msgBox`. Esto es para facilitar el estilo del elemento.

Si echas un vistazo al CSS en la página, verás que estamos utilizando un selector de clases.`msgBox` para dar estilo al contenedor del mensaje.

Finalmente, llamamos a una función del DOM llamada `Node.appendChild()` en la variable `html` que hemos guardado anteriormente, que anida un elemento dentro del otro como hijo de él.

Hemos especificado el panel `<div>` como el hijo que queremos añadir dentro del elemento `<html>`. Debemos hacer esto ya que el elemento que creamos no aparecerá en la página por sí solo — tenemos que especificar donde ponerlo.

```
let panel = document.createElement('div');
panel.setAttribute('class', 'msgBox');
html.appendChild(panel);
```

Las siguientes dos secciones hacen uso de las mismas funciones `createElement()` y `appendChild()` que ya vimos para crear dos nuevos elementos — un `<p>` y un `<button>` — e insertarlo en la página como un hijo del panel `<div>`. Usamos su propiedad `Node.textContent` — que representa el contenido de texto de un elemento: para insertar un mensaje dentro del párrafo y una 'x' dentro del botón. Este botón será lo que necesita hacer clic / activar cuando el usuario quiera cerrar el cuadro de mensaje.

```
let msg = document.createElement('p');
msg.textContent = 'This is a message box';
panel.appendChild(msg);

let closeBtn = document.createElement('button');
closeBtn.textContent = 'x';
panel.appendChild(closeBtn);
```

Finalmente, usamos el manejador de evento `GlobalEventHandlers.onclick` para hacerlo de modo que cuando se haga clic en el botón, se ejecute algún código para eliminar todo el panel de la página, para cerrar el cuadro de mensaje.

Brevemente, el handler `onclick` es una propiedad disponible en el botón (o, de hecho, en cualquier elemento de la página) que se puede configurar en una función para especificar qué código ejecutar cuando se hace clic en el botón.

Estamos haciendo el handler `onclick` igual que una función anónima, que contiene el código para ejecutar cuando se ha hecho click en el botón. La línea dentro de la función usa la función del DOM API `Node.removeChild()` para especificar que queremos

eliminar un elemento secundario específico del elemento HTML— en este caso el panel `<div>`.

```
closeBtn.onclick = function() {
  panel.parentNode.removeChild(panel);
}
```

Básicamente, todo este bloque de código está generando un bloque de HTML que se ve así, y lo está insertando en la página:

```
<div class="msgBox">
  <p>This is a message box</p>
  <button>x</button>
</div>
```

Fue un montón de código con el que trabajar: ¡no te preocupes demasiado si no recuerdas exactamente cómo funciona todo ahora! La parte principal en la que queremos centrarnos aquí es la estructura y el uso de la función, pero queríamos mostrar algo interesante para este ejemplo.

Llamando a la función

Ahora tienes la definición de tu función escrita en tu elemento `<script>` bien, pero no hará nada tal como está.

Intente incluir la siguiente línea debajo de su función para llamarla:

```
displayMessage();
```

Esta línea invoca la función, haciéndola correr inmediatamente. Cuando guarde el código y lo vuelva a cargar en el navegador, verá que el pequeño cuadro de mensaje aparece inmediatamente, solo una vez. Después de todo, solo lo llamamos una vez.

Ahora abra las herramientas de desarrollo de su navegador en la página de ejemplo, vaya a la consola de JavaScript y escriba la línea nuevamente allí, ¡verá que aparece nuevamente! Ahora tenemos una función reutilizable que podemos llamar en cualquier momento que queramos.

Pero probablemente queremos que aparezca en respuesta a las acciones del usuario y

del sistema. En una aplicación real, tal cuadro de mensaje probablemente se llamará en respuesta a la disponibilidad de nuevos datos, a un error, al usuario que intenta eliminar su perfil ("¿está seguro de esto?"), o al usuario que agrega un nuevo contacto y la operación se está completando con éxito ... etc.

En esta demostración, obtendremos el cuadro de mensaje que aparecerá cuando el usuario haga clic en el botón. Elimina la línea anterior que agregaste.

A continuación, seleccionaremos el botón y guardaremos una referencia a él en una variable. Agregue la siguiente línea a su código, encima de la definición de la función:

```
let btn = document.querySelector('button');
```

Finalmente, agregue la siguiente línea debajo de la anterior:

```
btn.onclick = displayMessage;
```

De una forma similar que nuestra línea dentro de la función `closeBtn.onclick`..., aquí estamos llamando a algún código en respuesta a un botón al hacer clic. Pero en este caso, en lugar de llamar a una función anónima que contiene algún código, estamos llamando directamente a nuestro nombre de función.

Intente guardar y actualizar la página: ahora debería ver aparecer el cuadro de mensaje cuando haga clic en el botón.

Quizás te estés preguntando por qué no hemos incluido los paréntesis después del nombre de la función. Esto se debe a que no queremos llamar a la función inmediatamente, solo después de hacer clic en el botón.

Si intentas cambiar la línea a:

```
btn.onclick = displayMessage();
```

Al guardar y volver a cargar, verás que aparece el cuadro de mensaje sin hacer clic en el botón. Los paréntesis en este contexto a veces se denominan "operador de invocación de función". Solo los utiliza cuando desea ejecutar la función inmediatamente en el ámbito actual. Del mismo modo, el código dentro de la función anónima no se ejecuta inmediatamente, ya que está dentro del alcance de la función.

Si has intentado el último experimento, asegúrate de deshacer el último cambio antes de continuar.

Mejora de la función con parámetros.

Tal como está, la función aún no es muy útil, no queremos mostrar el mismo mensaje predeterminado cada vez. Mejoremos nuestra función agregando algunos parámetros, permitiéndonos llamarla con algunas opciones diferentes.

En primer lugar, actualice la primera línea de la función:

```
function displayMessage() {
```

colocando esta:

```
function displayMessage(msgText, msgType) {
```

Ahora, cuando llamamos a la función, podemos proporcionar dos valores variables dentro de los paréntesis para especificar el mensaje que se mostrará en el cuadro de mensaje y el tipo de mensaje que es.

Para utilizar el primer parámetro, actualiza la siguiente línea dentro de su función:

```
msg.textContent = 'This is a message box';
```

a

```
msg.textContent = msgText;
```

Por último, pero no menos importante, ahora necesita actualizar su llamada de función para incluir un texto de mensaje actualizado. Cambia la siguiente línea:

```
btn.onclick = displayMessage;
```

por este código:

```
btn.onclick = function() {
  displayMessage('Woo, this is a different message!');
};
```

Si queremos especificar parámetros dentro de paréntesis para la función a la que estamos llamando, no podemos llamarla directamente, necesitamos colocarla dentro de una función anónima para que no esté en el ámbito inmediato y, por lo tanto, no se llame de inmediato. Ahora no se llamará hasta que se haga clic en el botón.

Vuelva a cargar e intenta el código nuevamente y verás que aún funciona bien, ¡excepto que ahora también puede variar el mensaje dentro del parámetro para obtener diferentes mensajes mostrados en el cuadro!

Un parámetro más complejo.

El siguiente parámetro va a implicar un poco más de trabajo: lo configuraremos de modo que, dependiendo de la configuración del parámetro `msgType`, la función mostrará un icono diferente y un color de fondo diferente.

En primer lugar, descarga los iconos necesarios para este ejercicio (warning y chat) de GitHub. Guárdalos en una nueva carpeta llamada icons en la misma localización que tu HTML.

A continuación, encuentra el CSS dentro de tu archivo HTML. Haremos algunos cambios para dar paso a los iconos. Primero, actualiza el ancho de `.msgBox` desde:

```
width: 200px;
```

por lo siguiente:

```
width: 242px;
```

Luego, añade las siguientes líneas dentro de la regla.`msgBox p { ... }`:

```
padding-left: 82px;
background-position: 25px center;
background-repeat: no-repeat;
```

Ahora necesitamos añadir código a la función `displayMessage()` para manejar la visualización de los iconos. Agrega el siguiente bloque justo encima de la llave de cierre `(})` de tu función:

```
if (msgType === 'warning') {
  msg.style.backgroundImage = 'url(icons/warning.png)';
  panel.style.backgroundColor = 'red';
} else if (msgType === 'chat') {
  msg.style.backgroundImage = 'url(icons/chat.png)';
  panel.style.backgroundColor = 'aqua';
} else {
  msg.style.paddingLeft = '20px';
}
```

Aquí, si el parámetro `msgType` se establece como `'warning'`, se muestra el icono de advertencia y el color de fondo del panel se establece en rojo. Si se establece en `'chat'`, se muestra el icono de chat y el color de fondo del panel se establece en azul aguamarina.
Si el parámetro `msgType` no está configurado en absoluto (o en algo diferente), entonces la parte `else { ... }` del código entra en juego, y al párrafo simplemente se le da un relleno predeterminado y ningún icono, sin el conjunto de colores del panel de fondo ya sea. Esto proporciona un estado predeterminado si no se proporciona ningún parámetro `msgType` , lo que significa que es un parámetro opcional.

Vamos a probar nuestra función actualizada, prueba a actualizar la llamada a `displayMessage()` de esto:

```
displayMessage('Woo, this is a different message!');
```

por uno de estos:

```
displayMessage('Your inbox is almost full -- delete some mails', 'warning');
displayMessage('Brian: Hi there, how are you today?','chat');
```

Puedes ver cuán útil se está volviendo nuestra (ahora no tan) poca función.