








JavaScript - Funciones y Eventos

Llevando tu código al siguiente nivel

Fundamentos de la Web - Clase 8

Objetivos de Hoy

¿Qué aprenderemos?

-  Crear y usar objetos en JavaScript
-  Entender qué son las funciones y cómo crearlas
-  Usar parámetros y valores de retorno (return)
-  Crear funciones fábrica (factory functions)
-  Manejar eventos de click en elementos HTML
-  Usar "this" para manipular elementos del DOM
-  Hacer páginas web interactivas con JavaScript





Objetos en JavaScript



¿Qué es un Objeto?

Piensa en una hamburguesa...

Imagina que queremos describir una hamburguesa. Tiene muchas características:

-  Tipo de pan
-  Tipo de carne
-  Tipo de queso
-  Ingredientes extras

En JavaScript, un **objeto** nos permite agrupar toda esta información relacionada en un solo lugar.



Creando un Objeto

Sintaxis básica:

```
var nombreObjeto = {  
  propiedad1: valor1,  
  propiedad2: valor2,  
  propiedad3: valor3  
};
```

Ejemplo real:

```
var hamburguesa = {  
  pan: "pan de hamburguesa",  
  carne: "carne de res",  
  queso: "cheddar",  
  ingredientes: ["lechuga", "tomate", "cebolla", "pepinillos"]  
};
```



Accediendo a Propiedades

Usar el punto (.) para acceder

```
var hamburguesa = {  
  pan: "pan de hamburguesa",  
  carne: "carne de res",  
  queso: "cheddar",  
  ingredientes: ["lechuga", "tomate", "cebolla"]  
};  
  
// Acceder a las propiedades  
console.log(hamburguesa.pan);    // "pan de hamburguesa"  
console.log(hamburguesa.carne);  // "carne de res"  
console.log(hamburguesa.queso);  // "cheddar"
```

Notación punto: `objeto.propiedad`



Ejemplo Completo: Objeto Hamburguesa

```
var hamburguesaClasica = {  
  "pan": "pan con semillas de sésamo",  
  "carne": "carne de res 100%",  
  "queso": "queso cheddar",  
  "extras": ["lechuga", "tomate", "cebolla", "salsa especial"]  
};  
  
// Mostrar información  
console.log("Pan: " + hamburguesaClasica.pan);  
console.log("Carne: " + hamburguesaClasica.carne);  
console.log("Queso: " + hamburguesaClasica.queso);  
console.log("Extras: " + hamburguesaClasica.extras);
```

💡 **Recuerda:** Las propiedades pueden tener cualquier tipo de dato (strings, números, arreglos, etc.)

? Pregunta Rápida

¿Qué imprimirá este código?

```
var persona = {  
  nombre: "Ana",  
  edad: 25,  
  ciudad: "Santiago"  
};  
  
console.log(persona.nombre);
```

Opciones:

- a) Ana
- b) "nombre"
- c) undefined
- d) Error

Piensa unos segundos... 🤔



Respuesta

```
var persona = {  
  nombre: "Ana",  
  edad: 25,  
  ciudad: "Santiago"  
};  
  
console.log(persona.nombre);
```

Respuesta correcta: a) Ana

¿Por qué?

- `persona.nombre` accede a la propiedad `nombre` del objeto
- El valor de esa propiedad es el string `"Ana"`
- `console.log()` muestra ese valor sin las comillas

¡Excelente si acertaste! 🎉

Métodos en Objetos

¿Qué son los Métodos?

Los objetos no solo guardan datos, ¡también pueden hacer cosas!

Un **método** es una función que pertenece a un objeto.

Piensa en esto:

- Una hamburguesa tiene **propiedades** (pan, carne, queso)
- Una hamburguesa puede **hacer algo** (mostrar su información)

```
var hamburguesa = {  
  pan: "pan de hamburguesa",  
  carne: "carne de res",  
  // Este es un método ↓  
  mostrarInfo: function() {  
    console.log("Hamburguesa lista!");  
  }  
};
```



Método en Acción

```
var hamburguesaClasica = {  
  "pan": "pan con semillas de sésamo",  
  "carne": "carne de res 100%",  
  "queso": "queso cheddar",  
  "extras": ["lechuga", "tomate", "cebolla", "salsa especial"],  
  
  // Método para mostrar información  
  "infoHamburguesa": function() {  
    console.log("Pan: " + this.pan);  
    console.log("Carne: " + this.carne);  
    console.log("Queso: " + this.queso);  
    console.log("Extras: " + this.extras.join(", "));  
  }  
};  
  
// Llamar al método  
hamburguesaClasica.infoHamburguesa();
```

La Palabra Mágica: this

¿Qué es "this"?

`this` se refiere al objeto actual. Es como decir "yo mismo" o "este objeto".

```
var persona = {  
  nombre: "Carlos",  
  edad: 30,  
  
  saludar: function() {  
    // "this" se refiere al objeto "persona"  
    console.log("Hola, soy " + this.nombre);  
    console.log("Tengo " + this.edad + " años");  
  }  
};  
  
persona.saludar();  
// Hola, soy Carlos  
// Tengo 30 años
```

 **Importante:** `this` evita que tengamos que repetir el nombre del objeto

? Pregunta Rápida

¿Qué mostrará este código?

```
var coche = {  
  marca: "Toyota",  
  modelo: "Corolla",  
  mostrarInfo: function() {  
    console.log(this.marca + " " + this.modelo);  
  }  
};  
  
coche.mostrarInfo();
```

Opciones:

- a) Toyota Corolla
- b) undefined undefined
- c) this.marca this.modelo
- d) Error

Piensa unos segundos 🤔



Respuesta

```
var coche = {  
  marca: "Toyota",  
  modelo: "Corolla",  
  mostrarInfo: function() {  
    console.log(this.marca + " " + this.modelo);  
  }  
};  
coche.mostrarInfo();
```

Respuesta correcta: a) **Toyota Corolla** ¿Por qué?

- **this** se refiere al objeto **coche**
- **this.marca** es **"Toyota"**
- **this.modelo** es **"Corolla"**
- La concatenación resulta en **"Toyota Corolla"**




¡Muy bien si acertaste! 🎉

Funciones en JavaScript

¿Qué es una Función?

Una función es como una receta de cocina

Imagina que tienes una receta para hacer galletas:

1.  Escribes los pasos una sola vez
2.  Puedes hacer galletas cuando quieras
3.  Reutilizas la receta muchas veces

En programación:

- Una función es un bloque de código que puedes reutilizar
- Se ejecuta solo cuando la llamas
- Evita repetir el mismo código una y otra vez



Sintaxis de una Función

Estructura básica:

```
function nombreFuncion() {  
    // Código que se ejecutará  
}
```

Ejemplo simple:

```
function saludar() {  
    console.log("¡Hola, amigo!");  
}  
  
// Hasta aquí solo CREAMOS la función  
// No se ejecuta automáticamente
```

💡 **Importante:** Crear una función NO la ejecuta. Es como escribir una receta pero no cocinar todavía.

Llamando una Función

Para ejecutar una función, debemos "llamarla"

```
// 1. Primero creamos la función
function saludar() {
    console.log("¡Hola, amigo!");
}

// 2. Luego la llamamos (ejecutamos)
saludar(); // ← Los paréntesis son necesarios

// Resultado: ¡Hola, amigo!
```

Pasos:

1. `function nombreFuncion()` → Crear/definir
2. `nombreFuncion()` → Llamar/invocar/ejecutar



Reutilizando Funciones

El poder de las funciones: llamarlas múltiples veces

```
function saludar() {  
    console.log("¡Hola, amigo!");  
}  
  
// Podemos llamarla cuantas veces queramos  
saludar(); // ¡Hola, amigo!  
saludar(); // ¡Hola, amigo!  
saludar(); // ¡Hola, amigo!
```

Ventaja: Escribimos el código una vez, lo usamos muchas veces.

Sin funciones, tendríamos que escribir:

```
console.log("¡Hola, amigo!");  
console.log("¡Hola, amigo!");  
console.log("¡Hola, amigo!");
```

? Pregunta Rápida

¿Cuántas veces se imprime "Hola Mundo"?

```
function mensaje() {  
    console.log("Hola Mundo");  
}  
  
mensaje();
```

Opciones:

- a) 0 veces (no se imprime nada)
- b) 1 vez
- c) 2 veces
- d) Error

Piensa unos segundos... 🤔



Respuesta

```
function mensaje() {  
    console.log("Hola Mundo");  
}  
  
mensaje();
```

Respuesta correcta: b) 1 vez

¿Por qué?

- La función `mensaje()` se define con un `console.log()`
- La función se llama UNA vez con `mensaje()`
- Por lo tanto, imprime "Hola Mundo" una sola vez

Nota: Si quisiéramos imprimir 3 veces, llamaríamos `mensaje()` tres veces.

¡Perfecto si acertaste! 🎉



Parámetros en Funciones

¿Qué son los Parámetros?

Los parámetros hacen las funciones más flexibles

Sin parámetros:

```
function saludar() {  
    console.log("¡Hola, amigo!");  
}  
saludar(); // Siempre saluda igual
```

Con parámetros:

```
function saludar(nombre) {  
    console.log("¡Hola, " + nombre + "!");  
}  
saludar("Ana"); // ¡Hola, Ana!  
saludar("Carlos"); // ¡Hola, Carlos!  
saludar("María"); // ¡Hola, María!
```


12 34 Parámetros vs Argumentos

Terminología importante:

```
// "nombreAmigo" es el PARÁMETRO
function saludar(nombreAmigo) {
    console.log("¡Hola, " + nombreAmigo + "!");
}

// "Luis" es el ARGUMENTO
saludar("Luis");
```

Diferencia:

- **Parámetro:** La variable en la definición de la función (al crearla)
- **Argumento:** El valor real que pasamos al llamar la función

Múltiples Parámetros

Una función puede recibir varios parámetros

```
function sumar(a, b) {  
    var resultado = a + b;  
    console.log("El resultado es: " + resultado);  
}  
  
sumar(5, 3);    // El resultado es: 8  
sumar(10, 20); // El resultado es: 30
```

Con 3 parámetros:

```
function presentar(nombre, edad, ciudad) {  
    console.log("Hola, soy " + nombre);  
    console.log("Tengo " + edad + " años");  
    console.log("Vivo en " + ciudad);  
}  
  
presentar("Ana", 25, "Santiago");
```



Ejemplo Práctico: Calculadora

```
function multiplicar(num1, num2) {  
    var resultado = num1 * num2;  
    console.log(num1 + " x " + num2 + " = " + resultado);  
}  
  
multiplicar(5, 4);    // 5 x 4 = 20  
multiplicar(7, 3);    // 7 x 3 = 21  
multiplicar(10, 10);  // 10 x 10 = 100
```

Ventaja: Una sola función puede hacer muchos cálculos diferentes.

? Pregunta Rápida

¿Qué imprimirá este código?

```
function calcularEdad(añoNacimiento, añoActual) {  
    var edad = añoActual - añoNacimiento;  
    console.log("Tienes " + edad + " años");  
}  
  
calcularEdad(1995, 2025);
```

Opciones:

- a) Tienes 1995 años
- b) Tienes 2025 años
- c) Tienes 30 años
- d) Error

Piensa unos segundos... 🤔

✓ Respuesta

```
function calcularEdad(añoNacimiento, añoActual) {  
  var edad = añoActual - añoNacimiento;  
  console.log("Tienes " + edad + " años");  
}  
  
calcularEdad(1995, 2025);
```

Respuesta correcta: c) Tienes 30 años

¿Por qué?

- añoNacimiento recibe 1995
- añoActual recibe 2025
- $edad = 2025 - 1995 = 30$
- Se imprime "Tienes 30 años"

¡Excelente razonamiento si acertaste! 🎉



Return: Devolviendo Valores

¿Qué es Return?

Return devuelve un valor desde la función

Analogía de la heladería:

- Sin `return` : El heladero te muestra el helado pero nunca te lo da 😞
- Con `return` : El heladero te entrega el helado para que lo disfrutes 🍦


```
// SIN return (solo muestra)
function sumar(a, b) {
    var resultado = a + b;
    console.log(resultado);
}

// CON return (devuelve el valor)
function sumar(a, b) {
    var resultado = a + b;
    return resultado;
}
```

Usando Return

Return permite usar el resultado de la función

```
function sumar(a, b) {  
    return a + b;  
}  
  
// El valor retornado se puede guardar en una variable  
var total = sumar(5, 3);  
console.log("El total es: " + total); // El total es: 8  
  
// O usarlo directamente  
console.log("10 + 20 = " + sumar(10, 20)); // 10 + 20 = 30
```

 **Clave:** `return` "devuelve" el valor para que podamos usarlo fuera de la función

Ejemplo: Encontrar el Máximo

```
// Función que compara dos números y devuelve el mayor
function encontrarMaximo(a, b) {
    if (a > b) {
        return a; // Si a es mayor, devolvemos a
    } else {
        return b; // Si no, devolvemos b
    }
}

var numero1 = 10;
var numero2 = 7;

// La función devuelve el número mayor
var maximo = encontrarMaximo(numero1, numero2);

console.log("El máximo entre", numero1, "y", numero2, "es:", maximo);
// El máximo entre 10 y 7 es: 10
```

Return Detiene la Función

Importante: return termina la ejecución

```
function verificarEdad(edad) {  
  if (edad < 18) {  
    return "Eres menor de edad";  
    console.log("Este mensaje NUNCA se verá");  
  }  
  return "Eres mayor de edad";  
}  
  
console.log(verificarEdad(15)); // Eres menor de edad  
console.log(verificarEdad(25)); // Eres mayor de edad
```

Regla: Cuando se ejecuta `return`, la función termina inmediatamente.



Ejercicio Práctico: Return

```
function calcularAreaRectangulo(base, altura) {  
    var area = base * altura;  
    return area;  
}  
  
function calcularPerimetroRectangulo(base, altura) {  
    var perimetro = 2 * (base + altura);  
    return perimetro;  
}  
  
// Usando las funciones  
var miArea = calcularAreaRectangulo(5, 3);  
var miPerimetro = calcularPerimetroRectangulo(5, 3);  
  
console.log("Área: " + miArea);           // Área: 15  
console.log("Perímetro: " + miPerimetro); // Perímetro: 16
```

? Pregunta Rápida

¿Qué valor tendrá la variable `resultado`?

```
function multiplicar(x, y) {  
    return x * y;  
}  
  
var resultado = multiplicar(6, 7);
```

Opciones:

- a) 6
- b) 7
- c) 42
- d) undefined

Piensa unos segundos... 🤔

✓ Respuesta

```
function multiplicar(x, y) {  
    return x * y;  
}  
  
var resultado = multiplicar(6, 7);
```

Respuesta correcta: c) 42

¿Por qué?

- La función `multiplicar` recibe 6 y 7
- Calcula `6 * 7 = 42`
- `return 42` devuelve el valor 42
- La variable `resultado` recibe ese valor (42)

¡Perfecto cálculo si acertaste! 🎉



Factory Functions (Funciones Fábrica)



¿Qué es una Factory Function?

Una función que crea y devuelve objetos

Problema: Crear muchos objetos similares es repetitivo

```
var sandwich1 = { pan: "integral", proteina: "pollo", queso: "suizo" };  
var sandwich2 = { pan: "blanco", proteina: "jamón", queso: "cheddar" };  
var sandwich3 = { pan: "centeno", proteina: "pavo", queso: "provolone" };  
// Mucho copy-paste...
```

Solución: Una función que crea objetos

```
function crearSandwich(pan, proteina, queso) {  
    return { pan: pan, proteina: proteina, queso: queso };  
}
```



Creando una Factory Function

Paso a paso:

```
function sandwichFactory(pan, proteína, queso, salsas) {  
  // 1. Crear un objeto vacío  
  var sandwich = {};  
  
  // 2. Agregar propiedades con los parámetros  
  sandwich.pan = pan;  
  sandwich.proteína = proteína;  
  sandwich.queso = queso;  
  sandwich.salsas = salsas;  
  
  // 3. Devolver el objeto completo  
  return sandwich;  
}
```




Usando la Factory Function

```
function sandwichFactory(pan, proteína, queso, salsas) {  
  var sandwich = {};  
  sandwich.pan = pan;  
  sandwich.proteína = proteína;  
  sandwich.queso = queso;  
  sandwich.salsas = salsas;  
  return sandwich;  
}  
  
// Crear diferentes sándwiches fácilmente  
var s1 = sandwichFactory("trigo", "pavo", "provolone",  
                          ["mostaza", "cebolla frita", "rúcula"]);  
  
var s2 = sandwichFactory("integral", "pollo", "suizo",  
                          ["lechuga", "tomate", "mayonesa"]);  
  
console.log(s1);  
console.log(s2);
```

Ejercicio: Pizza Factory

¡Hora de practicar!

Crea un archivo `pizza-factory.js` con:

1. Una función `pizzaOven` que reciba:
 - `tipoCorteza` (ej: "estilo Chicago")
 - `salsa` (ej: "tradicional")
 - `quesos` (arreglo, ej: ["mozzarella"])
 - `toppings` (arreglo, ej: ["pepperoni", "salchicha"])
2. La función debe devolver un objeto pizza
3. Crea al menos 3 pizzas diferentes

 **Tiempo:** 15 minutos

Solución: Pizza Factory




```
function pizzaOven(tipoCorteza, salsa, quesos, toppings) {  
  var pizza = {};  
  pizza.tipoCorteza = tipoCorteza;  
  pizza.salsa = salsa;  
  pizza.quesos = quesos;  
  pizza.toppings = toppings;  
  return pizza;  
}  
  
// Pizza 1: Estilo Chicago  
var pizza1 = pizzaOven("estilo Chicago", "tradicional",  
  ["mozzarella"],  
  ["pepperoni", "salchicha"]);  
  
// Pizza 2: Vegetariana  
var pizza2 = pizzaOven("lanzada a mano", "marinara",  
  ["mozzarella", "feta"],  
  ["champiñones", "aceitunas", "cebollas"]);  
  
console.log(pizza1);  
console.log(pizza2);
```

Eventos en JavaScript




¿Qué son los Eventos?

Los eventos son las interacciones del usuario con la página

Piensa en una fiesta:

-  Alguien toca el timbre → respondes abriendo la puerta
-  Suena tu teléfono → respondes contestando
-  Llega la pizza → respondes comiendo

En una página web:

-  Click en un botón → ejecutas código JavaScript
-  Escribir en un input → validar el texto
-  Pasar el mouse → cambiar el estilo

Los eventos permiten que las páginas sean **interactivas**.

El Evento Click

El evento más común: onclick

```
// HTML
<button id="miBoton">¡Haz clic aquí!</button>

// JavaScript
document.getElementById("miBoton").addEventListener("click", function() {
    alert("¡Me hiciste clic!");
});
```

Componentes:

1. `document.getElementById("miBoton")` → Seleccionar el elemento
2. `.addEventListener("click", ...)` → Escuchar el evento click
3. `function() { ... }` → Código que se ejecuta al hacer click

Estructura de Archivos HTML + JS

Conectando HTML con JavaScript

index.html:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Mi Página con JS</title>
  <script src="script.js" defer></script>
</head>
<body>
  <h1>Haz clic en el botón</h1>
  <button id="miBoton">¡Clic aquí!</button>
</body>
</html>
```

⚠ **Importante:** La palabra `defer` hace que el JavaScript se cargue DESPUÉS del HTML



Ejemplo: Cambiar Color de Fondo

index.html:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Cambia el color de fondo</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js" defer></script>
</head>
<body>
  <h1>Haz clic en el botón para cambiar el color de fondo</h1>
  <button id="colorButton">¡Cambia el color!</button>
</body>
</html>
```




Ejemplo: CSS (style.css)

```
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
}  
body {  
  font-family: Arial, sans-serif;  
  text-align: center;  
  transition: background-color 0.5s ease;  
}  
button {  
  padding: 10px 20px;  
  font-size: 18px;  
  background-color: dodgerblue;  
  color: white;  
  border: none;  
  cursor: pointer;  
  margin-top: 20px;  
  border-radius: 5px;  
}
```



Ejemplo: JavaScript

script.js:

```
document.getElementById("colorButton").addEventListener("click", function() {
    cambiarColor();
});

function cambiarColor() {
    var colores = ["#FF6347", "#66CDAA", "#9370DB",
                  "#FFD700", "#4682B4", "#FFA07A"];

    // Elegir un color aleatorio
    var colorAleatorio = colores[Math.floor(Math.random() * colores.length)];

    // Cambiar el color de fondo del body
    document.body.style.backgroundColor = colorAleatorio;
}
```

Desglosando el Código

```
document.getElementById("colorButton").addEventListener("click", function() {  
    cambiarColor();  
});
```

Paso a paso:

1. `document` → El documento HTML completo
2. `.getElementById("colorButton")` → Buscar elemento con id="colorButton"
3. `.addEventListener("click", ...)` → Escuchar eventos de click
4. `function() { ... }` → Función que se ejecuta al hacer click
5. `cambiarColor()` → Llamar a nuestra función personalizada

Números Aleatorios

Math.random() en acción:

```
var colores = ["#FF6347", "#66CDAA", "#9370DB",  
              "#FFD700", "#4682B4", "#FFA07A"];  
  
// Math.random() devuelve un número entre 0 y 1  
// Ejemplo: 0.7382  
  
// Multiplicamos por la cantidad de elementos  
// 0.7382 * 6 = 4.4292  
  
// Math.floor() redondea hacia abajo  
// Math.floor(4.4292) = 4  
  
var indice = Math.floor(Math.random() * colores.length);  
var colorAleatorio = colores[indice];
```

? Pregunta Rápida

¿Qué hace este código?

```
document.getElementById("boton").addEventListener("click", function() {  
    alert("¡Hola!");  
});
```

Opciones:

- a) Muestra "¡Hola!" inmediatamente al cargar la página
- b) Muestra "¡Hola!" cuando se hace clic en el elemento con id="boton"
- c) Cambia el texto del botón a "¡Hola!"
- d) Error

Piensa unos segundos... 🤔

✓ Respuesta

```
document.getElementById("boton").addEventListener("click", function() {  
    alert("¡Hola!");  
});
```

Respuesta correcta: b) Muestra "¡Hola!" cuando se hace clic en el elemento con id="boton"

¿Por qué?

- `addEventListener("click", ...)` espera un click
- NO se ejecuta automáticamente
- Solo cuando el usuario hace clic, se muestra el alert
- `alert()` muestra una ventana emergente

¡Excelente comprensión si acertaste! 🎉

This en Eventos

This en Contexto de Eventos

"this" se refiere al elemento que recibió el evento

Analogía de la cocina:

- Cada ingrediente está en su propia estación
- Cuando usas "este" ingrediente, sabes dónde está
- `this` es como decir "este elemento específico"

```
button.addEventListener("click", function() {  
    // "this" se refiere al botón que fue clickeado  
    this.innerText = "¡Me clickeaste!";  
});
```




Propiedad innerText

innerText cambia el texto dentro de un elemento

```
// HTML
<p id="miParrafo">Este es un párrafo de ejemplo.</p>
<button id="botonCambiar">Cambiar texto</button>

// JavaScript
var botonCambiar = document.getElementById("botonCambiar");

botonCambiar.addEventListener("click", function() {
    // "this" se refiere al botón
    this.innerText = "¡Texto cambiado!";
});
```

Resultado: Al hacer clic, el texto del botón cambia de "Cambiar texto" a "¡Texto cambiado!"

Ejemplo: Eliminar Ninjas

index.html:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Eliminar Imagen</title>
  <link rel="stylesheet" href="style.css">
  <script src="script.js" defer></script>
</head>
<body>
  <div id="dojo">
    
    
    
    
  </div>
</body>
</html>
```




Ejemplo: CSS (style.css)

```
* {  
  margin: 0;  
  padding: 0;  
}  
body {  
  background-color: black;  
}  
#dojo {  
  width: 600px;  
  height: 200px;  
  display: flex;  
  align-items: center;  
  justify-content: space-evenly;  
}  
img {  
  width: 120px;  
  height: 120px;  
  cursor: pointer;  
}
```



Ejemplo: JavaScript (script.js)

```
// Obtenemos TODAS las imágenes con la clase 'imagenEliminar'  
var imagenes = document.querySelectorAll(".imagenEliminar");  
  
// Agregamos un evento 'click' a CADA imagen  
imagenes.forEach(function(imagen) {  
    imagen.addEventListener("click", function() {  
        // "this" se refiere a la imagen específica que fue clickeada  
        this.remove();  
    });  
});
```

Resultado: Al hacer clic en un ninja, ¡desaparece!  

querySelector vs getElementById

Diferentes formas de seleccionar elementos

```
// getElementById - busca por ID (solo 1 elemento)
var boton = document.getElementById("miBoton");

// querySelector - busca por selector CSS (primer elemento que coincida)
var boton = document.querySelector("#miBoton");

// querySelectorAll - busca TODOS los elementos que coincidan
var botones = document.querySelectorAll(".miClase");
```

Cuándo usar cada uno:

- `getElementById` → Cuando tienes un ID único
- `querySelector` → Cuando quieres usar selectores CSS (clases, atributos, etc.)
- `querySelectorAll` → Cuando quieres seleccionar MÚLTIPLES elementos

forEach para Eventos Múltiples

Agregar eventos a varios elementos a la vez

```
// Seleccionar todos los elementos con clase "carta"
var cartas = document.querySelectorAll(".carta");

// forEach recorre cada elemento
cartas.forEach(function(carta) {
    // Agregar evento click a cada carta
    carta.addEventListener("click", function() {
        this.style.backgroundColor = "gold";
    });
});
```

- `querySelectorAll` devuelve una lista de elementos
- `forEach` ejecuta una función para cada elemento
- Cada elemento recibe su propio evento click

? Pregunta Rápida

¿Qué hace este código?

```
var imagenes = document.querySelectorAll(".foto");

imagenes.forEach(function(imagen) {
  imagen.addEventListener("click", function() {
    this.remove();
  });
});
```

Opciones:

- a) Elimina todas las imágenes inmediatamente
- b) Elimina la primera imagen al hacer clic en ella
- c) Elimina la imagen específica en la que se hace clic
- d) Error

Piensa unos segundos... 🤔



Respuesta

```
var imagenes = document.querySelectorAll(".foto");

imagenes.forEach(function(imagen) {
  imagen.addEventListener("click", function() {
    this.remove();
  });
});
```

Respuesta correcta: c) Elimina la imagen específica en la que se hace clic

- `querySelectorAll` selecciona TODAS las imágenes con clase "foto"
- `forEach` agrega un evento a CADA imagen individualmente
- `this` se refiere a la imagen específica clickeada
- `remove()` elimina ESA imagen en particular

¡Perfecto análisis si acertaste! 🎉



Ejercicio: Contador de Clicks

¡Hora de practicar!

Crea un archivo `contador.html` con:

1. Un botón con el texto "0"
2. Cada vez que se haga clic, el número aumenta en 1
3. Usa `this.innerText` para cambiar el texto del botón
4. Pista: Necesitarás convertir el texto a número con `parseInt()`

Estructura:

```
<button id="contador">0</button>
```



Tiempo: 10 minutos

Solución: Contador de Clicks (contador.html)

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Contador</title>
  <script defer>
    document.addEventListener("DOMContentLoaded", function() {
      var boton = document.getElementById("contador");
      boton.addEventListener("click", function() {
        var valorActual = parseInt(this.innerText);
        this.innerText = valorActual + 1;
      });
    });
  </script>
</head>
<body>
  <button id="contador">0</button>
</body>
</html>
```



Resumen de Conceptos Clave



Lo Más Importante

Objetos:

- Agrupan propiedades relacionadas: `{ nombre: "Ana", edad: 25 }`
- Acceso con punto: `persona.nombre`
- Métodos: funciones dentro de objetos
- `this` en objetos: referencia al objeto mismo

Funciones:

- Bloques de código reutilizables
- Se crean con `function nombre() { }`
- Se llaman con `nombre()`
- Parámetros: reciben valores
- `return` : devuelven valores



Lo Más Importante (cont.)

Factory Functions:

- Funciones que crean y devuelven objetos
- Evitan repetir código
- Patrón: `function crear(...) { return { ... }; }`

Eventos:

- Permiten interactividad
- `addEventListener("click", function() { })`
- `this` en eventos: referencia al elemento clickeado
- `innerText` : cambiar texto
- `remove()` : eliminar elementos



Conceptos Clave para Recordar

1. **Objetos:** Estructuras que agrupan datos relacionados con propiedades
2. **Métodos:** Funciones que pertenecen a un objeto
3. **this:** Palabra clave que se refiere al contexto actual
4. **Funciones:** Bloques de código reutilizables que ejecutan tareas
5. **Parámetros:** Variables que reciben valores al llamar una función
6. **return:** Devuelve un valor desde una función
7. **Factory Functions:** Funciones que crean objetos
8. **Eventos:** Acciones del usuario (click, hover, etc.)
9. **addEventListener:** Método para escuchar eventos
10. **querySelector/All:** Métodos para seleccionar elementos del DOM

Recursos Adicionales

Documentación Oficial

- [MDN - Objetos](#)
- [MDN - Funciones](#)
- [MDN - Eventos](#)
- [MDN - this](#)

Herramientas Interactivas

- [JSFiddle](#) - Probar código JavaScript online
- [CodePen](#) - Crear proyectos HTML/CSS/JS





Próximos Pasos

Para la Próxima Clase:

Conocimientos previos necesarios:

- Dominar funciones y eventos
- Entender objetos y this
- Manipulación básica del DOM

Temas que cubriremos:

-  Más eventos (hover, submit, keypress)
-  Manipulación avanzada del DOM
-  Validación de formularios
-  Crear proyectos interactivos



Práctica para Casa

Proyecto Sugerido: Lista de Tareas Interactiva

Crea una aplicación con:

1. Un input para escribir tareas
2. Un botón "Agregar" que crea nuevas tareas
3. Cada tarea tiene un botón "Eliminar"
4. Al hacer clic en una tarea, se tacha (line-through)

Conceptos que practicarás:

- Eventos click
- Crear elementos con JavaScript
- Usar this para manipular elementos
- Factory functions para crear objetos tarea

Consejos Finales

Buenas Prácticas:

- ✓ Separa tu código: HTML en un archivo, CSS en otro, JS en otro
- ✓ Usa nombres descriptivos: `botonAgregar` en vez de `btn1`
- ✓ Siempre usa defer: `<script src="script.js" defer></script>`
- ✓ Comenta tu código: Explica qué hace cada función
- ✓ Prueba en partes pequeñas: No escribas todo el código de golpe

Errores Comunes a Evitar:

- ✗ Olvidar `defer` en el script tag
- ✗ Usar `getElementById` con un id que no existe
- ✗ No usar `this` correctamente en eventos
- ✗ Confundir parámetros con argumentos

 ¡Felicidades!

Ya sabes usar funciones y eventos en JavaScript

Has dado un gran paso en tu camino como desarrollador web 

? Preguntas

¿Alguna duda sobre funciones y eventos?

 ¡Excelente Trabajo!

¡Nos vemos en la siguiente clase!

No olvides completar todos los ejercicios 

Practica creando tus propios proyectos 