

T - CT - Pensamiento Creativo con IA | JavaScript con Inteligencia Artificial

Para este ejercicio elegí que ChatGPT me explicara el tema de “objetos” en el lenguaje de programación de JavaScript, relacionándolo con la serie de Netflix “StrangerThings”.

Fase 1. Primer prompt para que la IA (ChatGPT) explique el tema “Objetos en JavaScript”.

Imagina que eres un maestro de desarrollo web que está a punto de dar una clase sobre objetos en el lenguaje de programación JavaScript, explicarás el tema relacionándolo con la serie Stranger things. Tu audiencia será un grupo de jóvenes novatos en el mundo de la programación, por lo que sí puedes manejar un lenguaje técnico, pero claro y detallado para la comprensión del tema. Quiero que expliques la información paso a paso, con ejemplos y definición de términos. El texto debe estar bien estructurado.

Respuesta de ChatGPT:

★ Clase: Objetos en JavaScript explicados con Stranger Things

Resumen final (tipo “Previously on Stranger Things”)

- Un **objeto** es un conjunto de **propiedades** y **métodos**.
- Las **propiedades** describen características.
- Los **métodos** representan acciones.
- Se accede a las propiedades con **punto** o **corchetes**.
- `this` se refiere al objeto actual.
- Los objetos pueden **crecer**, **anidarse** y **modificarse**.


Fase 2. Segundo prompt

Sigamos hablando de objetos, pero quiero que seas más dinámico con la temática stranger things. Además, quiero que me muestres más ejemplos de cómo declarar un objeto, incluso utilizando métodos no convencionales. Quiero que muestres la información lo más sintetizado posible, con al menos tres ejemplos.

Respuesta de la IA:

Objetos en JavaScript — “Modo Stranger Things Dinámico”

En el mundo de Hawkins, cada personaje, criatura o artefacto raro puede representarse como un **objeto**.

Aquí tienes **tres formas distintas** (y poco convencionales) de crearlos, con ejemplos inspirados en la serie 



1. Crear objetos con `Object.create()`



2. Objetos creados con funciones constructoras



3. Objetos usando clases (`class`)



Bonus: Objeto declarado “a la velocidad del Demogorgon”



Resumen “Previously on Hawkins...”

- **`Object.create()`:** perfecto para heredar comportamientos “oscuros”.
- **Funciones constructoras:** generan personajes como fábrica.
- **Clases:** héroes modernos con poderes y métodos limpios.
- **Objetos literales:** rápidos y directos.

Tercer prompt:

Quiero que expliques el método `Object.create()` utilizando un solo ejemplo, creando un mini-juego donde los personajes de Stranger Things son objetos. Quiero que mantengas el dinamismo con la temática de stranger things, pero que agregues notas claras en el código para entender la secuencia. Además, quiero que me proporciones el código html y js para poder ejecutar el código.

Fase 3. Cuarto prompt.




Quiero que revises el código, quiero que evalúes la legibilidad posibles bugs, buenas prácticas y quiero al menos dos optimizaciones para mejorar la eficiencia o legibilidad.

¡Perfecto! Vamos a hacer una **revisión completa de tu código**, evaluando legibilidad, posibles bugs, buenas prácticas y proponiendo optimizaciones. Me basaré en el **código del mini-juego con `Object.create()`** que compartimos antes.

Respuesta de la IA.

1 Revisión del código


Legibilidad


-  El código está **bien estructurado**, con secciones comentadas: prototipo base, creación de personajes, mini-juego.
-  Nombres de variables y funciones son **claros** (`personajeBase`, `eleven`, `demogorgon`, `atacar`, `recibirDaño`).
-  Comentarios explicativos son útiles, sobre todo para novatos.

Pequeña mejora de legibilidad:




- Podríamos **agrupar acciones del juego** en una función `simularBatalla()` para encapsular la lógica, en vez de dejar los turnos directamente en el flujo global. Esto ayuda a mantener el código limpio y más fácil de expandir.
-

Posibles bugs

-  **Vida negativa:** actualmente vida puede bajar por debajo de 0. Esto no rompe el código, pero visualmente puede ser confuso.

-  **Valores fijos de daño:** los ataques tienen valores estáticos (40, 25, 80). Esto no es un bug, pero limita la reutilización y la aleatoriedad del juego.

Buenas prácticas

-  Uso correcto de `Object.create()` para herencia prototípica.
-  Separación de código en HTML y JS.
-  Buenas prácticas faltantes:
 1. Evitar “hardcodear” valores dentro de la simulación.
 2. Usar **const** y **let** de manera consistente (ya se está haciendo).
 3. Posible encapsulamiento de funciones de juego dentro de un objeto Juego o función para evitar contaminación global.

2 Optimización 1: Encapsular la simulación de batalla

En lugar de escribir cada turno en el flujo global, podemos crear una función `simularBatalla()` que reciba los personajes y los ataques. Esto mejora **legibilidad y reusabilidad**.

-  **Ventaja: el juego se vuelve más dinámico y realista.**

3 Resumen de mejoras sugeridas

Categoría	Acción sugerida
Legibilidad	Encapsular la batalla en una función (<code>simularBatalla</code>)
Robustez	Evitar vida negativa en <code>recibirDaño()</code>
Dinamismo	Permitir daño aleatorio o parametrizado
Escalabilidad	Crear un objeto Juego o funciones reutilizables para simular más batallas
Buenas prácticas JS	Mantener <code>const/let</code> , evitar valores “hardcode”, mantener funciones cortas

Quinto prompt.

Me gusta, pero me gustaría que agregaras más estilo y diseño a la página.

Respuesta de la IA.

-  **Mejoras aplicadas en esta versión**

1. **Estilo visual** tipo Stranger Things (fondo oscuro, rojo intenso, tipografía).

2. **Barras de vida animadas** y cambio de color crítico.
3. **Log visual** en la página para seguir la batalla, en vez de la consola.
4. **Botón interactivo** para iniciar la batalla y resetear vidas.
5. Código sigue siendo limpio, modular y escalable para agregar más personajes o ataques especiales.

Fase 4. Sexto prompt.

Finalmente, quiero que generes un diagrama que explique de forma clara con elementos visuales por qué se puede utilizar el método `Object.create()` para la creación de objetos en JavaScript. Quiero que el diagrama lo mantengas con la temática del código y que utilices un lenguaje apropiado para explicar el tema a un público no técnico.

Respuesta de la IA.

Concepto del Diagrama

Idea central: `Object.create()` permite **crear nuevos personajes** que heredan poderes y habilidades de un "molde" o **prototipo**, tal como Eleven y Demogorgon heredan del `personajeBase`.

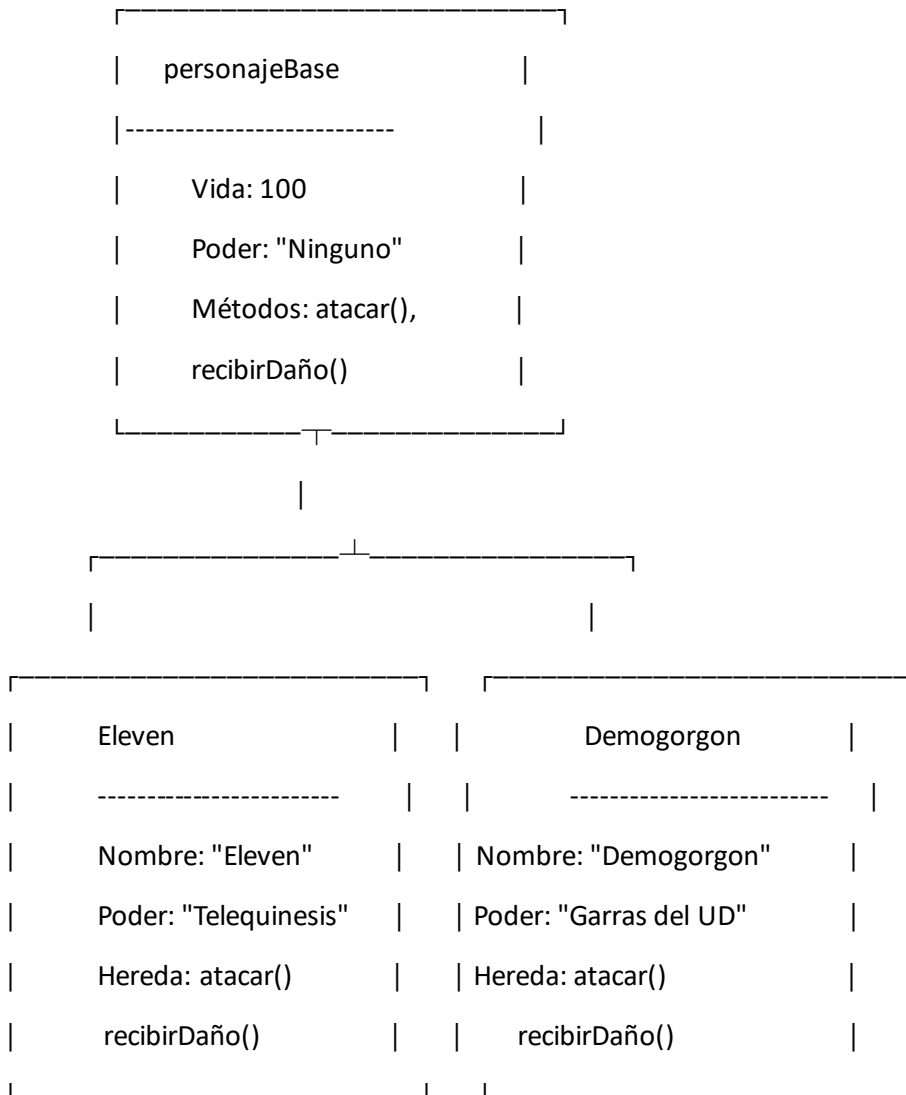
El diagrama tendrá tres niveles:

1. **Molde base (`personajeBase`)** → el prototipo común.
 2. **Personajes que heredan (Eleven, Demogorgon)** → usan los métodos y propiedades del prototipo.
 3. **Propiedades propias** → cada personaje puede tener atributos únicos (poderes, nombre).
-

Resultado final:

Explicación sencilla para no técnicos

- **Molde base (`personajeBase`)** → Es como la **receta de las criaturas de Hawkins**, con vida y habilidades comunes.
- **`Object.create()`** → Es como **clonar la receta**, pero luego puedes cambiar el nombre o los poderes de cada personaje.
- **Herencia de métodos** → No necesitamos reescribir `atacar()` ni `recibirDaño()` para cada personaje; todos los clones la usan automáticamente.
- **Propiedades únicas** → Cada personaje puede tener **atributos especiales**, como Eleven con telequinesis o Demogorgon con garras.



◆ Analogía Stranger Things

- **Molde Base** → Laboratorio secreto de Hawkins donde crean las criaturas.
- **Personajes creados** → Eleven, Demogorgon... todos salen del laboratorio, **heredando habilidades básicas**, pero cada uno tiene su **poder especial único**.
- **Ventaja de `Object.create()`** → Evita copiar todo a mano y mantiene el código limpio y organizado, como si fueran "clones controlados" desde un prototipo común.