



Girls Who Code en casa

Depuración de código faltante, parte 2
Errores lógicos

Descripción de la actividad

En la parte 1, aprendimos a identificar y depurar errores de sintaxis en el cuestionario de personalidad con errores usando la ventana de la consola. En la parte 2, examinaremos un nuevo tipo de error: el error lógico; y aprenderemos un conjunto de estrategias de depuración para reparar nuestro código de personalidad. Los programadores usan la lógica para indicarle al programa qué acciones realizar. Sin embargo, a veces nuestra lógica no se alinea con las instrucciones que la computadora necesita para ejecutar un programa correctamente. Trabajaremos con JavaScript como lo hicimos en la parte 1, pero los métodos de depuración se aplican a todos los lenguajes. No es necesario saber de JavaScript para completar esta actividad.

Lo que **usted piensa** que programó:

SI afuera hace frío,
ENTONCES use una chaqueta.

Lo que la **computadora lee** como su programa:

SI afuera hace frío,
ENTONCES baile al ritmo de la pizza.

Ya debería haber completado la [parte 1](#) de Depuración de código faltante antes de embarcarse en esta actividad.

Objetivos del aprendizaje

Al finalizar esta actividad, serás capaz de:

- ❑ describir la diferencia entre errores lógicos y errores de sintaxis;
- ❑ describir una variedad de estrategias para depurar el código;
- ❑ aplicar estrategias de depuración para arreglar el error lógico de un sitio web dañado.

Materiales

- Editor [Repl.it](https://repl.it)
- [Cuestionario de personalidad, proyecto modelo](#)
- [Cuestionario de personalidad, proyecto con errores](#) (solo con errores lógicos)
- [Guía de referencia de código faltante](#)

Conocimientos previos

Antes de comenzar este proyecto, le recomendamos que:

- esté familiarizada con conceptos informáticos clave, como [variables](#), [funciones](#) y [sentencias condicionales](#) de cualquier lenguaje de programación;
- tenga un nivel de experiencia de principiante con un lenguaje basado en texto, como JavaScript, Python, Swift, etc.

Si quiere obtener más información sobre JavaScript, consulte la actividad [Excursión virtual](#) de Girls Who Code en casa.

Si quiere practicar la depuración en Scratch, consulte la actividad [Depuración audaz, pero no perfecta con Scratch](#) de Girls Who Code en casa.

“Mujeres en tecnología” artículo destacado: Simone Giertz



Fuente: [Mashable](#)

Entre los inventos de Simone Giertz, se incluyen robots que lavan el cabello, cortan verduras y se pintan los labios por la mañana. Sin embargo, sus robots rara vez (o nunca) tienen éxito y, en su mayoría, son inútiles, pero Simone acepta este fracaso. Ella dice que “la verdadera belleza de hacer cosas inútiles [es] reconocer que uno no siempre sabe cuál es la mejor respuesta”.

Conocida por sus seguidores como “la señora del funcionamiento defectuoso”, la inventora y entusiasta de la robótica Simone Giertz pasa la mayor parte de su tiempo trasteando con Arduino en su canal de YouTube, presentando *Tested* con Adam Savage y hablando sobre sus excéntricos proyectos en vivo.

Mire [este video](#) para obtener más información sobre la máquina de aplausos, uno de los extravagantes inventos de Simone, y su inspiración detrás del producto y el diseño.

Tras ser diagnosticada con un tumor cerebral no canceroso en 2018, Simone se tomó un tiempo para alejarse de sus inventos con el fin de recibir tratamiento y reflexionar. Ahora que está recuperada, Simone sigue inventando, pero ha cambiado su enfoque y hoy se centra en productos más grandes y funcionales. ¿Su última gran jugada? Convertir un Tesla en una camioneta. Conozca a [Truckla](#).

Reflexión

Recuerde que ser un experto informático es más que sencillamente ser bueno codificando. Hable sobre cómo Simone y su trabajo se relacionan con las fortalezas en las que se centran los expertos informáticos, al igual que usted en su programa Girls Who Code.



PROPÓSITO

Simone mantiene una actitud positiva, incluso cuando sus robots o sus proyectos no funcionan de la manera esperada. ¿Cómo usted puede ayudarse a sí misma a mantener el optimismo ante un contratiempo?

Comparta sus respuestas con un familiar o amigo. Anime a los demás para que lean más sobre Simone y se unan a la charla.

Paso 1: Conocer los errores lógicos (de 6 a 8 minutos)

¿Qué es la lógica? (1 minuto)

Contrariamente a lo que se piensa, las computadoras no son inteligentes. Son máquinas que, al menos por ahora, no saben tomar decisiones ni realizar una acción por sí mismas. Las computadoras necesitan que los humanos les digan lo que tienen que hacer, es decir, que las programen con una serie de instrucciones. Estas instrucciones se denominan **algoritmos**. Pero, ¿cómo sabe el programa qué instrucción seguir o cuándo debe hacerlo? Aquí entra a jugar la **lógica**. Usamos la lógica para programar el orden o la **secuencia** que se debe seguir para que se completen las tareas.

¿Cómo uso la lógica? (de 3 a 4 minutos)

Aquí se mencionan tres estructuras frecuentes que los programadores usan para añadir lógica y determinar la secuencia de sus programas. Hemos escrito estos ejemplos en JavaScript, pero el uso de estas estructuras se traduce a la mayoría de los lenguajes de programación.

SENTENCIAS CONDICIONALES

Estas sentencias le indican a nuestro programa cómo tomar decisiones al evaluar si una sentencia es verdadera o falsa usando [if](#), [else if](#), [else](#) y [operadores](#) de comparaciones o lógicos.

```
if(numPlayers >= 5){
    playAmongUs();
} else {
    playScattergories();
}
```

Esta sentencia condicional toma una decisión sobre qué juego jugar según la cantidad de jugadores. Si la cantidad de jugadores es mayor o igual a cinco, jugar a *Among Us*; si no, jugar a *Scattergories*.

FUNCIONES

Cree una nueva función siempre que usted tenga partes de código que quiera reutilizar o para mantener el código legible. Algunas funciones devuelven valores específicos, mientras que otras no lo hacen.

```
if(score == 10){
    updateWinner();
    restartGame();
}

function updateWinner(){
    console.log("Winner!");
}
```

```
function restartGame{
    score = 0;
}
```

Esta sentencia "if" le dice al programa que actualice el ganador y reinicie el juego si la puntuación es 10. Las funciones hacen que el código sea más legible porque agrupamos los subpasos en funciones que tienen nombres descriptivos.

BUCLAS

Bucles para repetir una serie de instrucciones hasta que se cumpla una sentencia condicional. Los principales tipos de bucles son [for](#) y [while](#).

```
while(winner == false){
    keepPlaying();
}
```

El bucle while le indica al programa que ejecute la función keepPlaying siempre que no haya un ganador.

Nota: No trabajaremos con bucles en esta actividad, pero es importante que los conozca.

Paso 1: Conocer los errores lógicos (continuación)

¿Dónde se hallan los errores? (1 minuto)

Los errores se pueden producir en cualquier punto del programa. Puede encontrarlos escondidos en [variables](#), [funciones](#), [sentencias condicionales](#) y más.

CÓDIGO

```
if (temp <= 65){
    wearJacket();
} else if (temp >= 66){
    removeJacket();
}

function wearJacket(){
    Sacar la chaqueta del armario
    Desabrochar
    Colocar un brazo
    Colocar el otro brazo
    Abrochar la chaqueta
}

function removeJacket(){
    Desabrochar la chaqueta
    Quitar un brazo
    Quitar el otro brazo
    Colgar la chaqueta en el clóset
}
```

SECUENCIA

La secuencia de este código si la temperatura es **47 grados Fahrenheit**:

Pruebe la sentencia condicional 1: temp <= 65?

- La sentencia condicional 1 evalúa si es VERDADERA.
- Vaya a la función wearJacket.
- Ejecute todas las líneas de código en la función wearJacket.
- Salga.

La secuencia de este código si la temperatura es **84 grados Fahrenheit**:

Pruebe la sentencia condicional 1: temp <= 65?

- La sentencia condicional 1 evalúa si es FALSA.

Pruebe la sentencia condicional 2: temp >= 66?

- La sentencia condicional 2 evalúa si es VERDADERA.
- Vaya a la función removeJacket.
- Ejecute todas las líneas de código en la función removeJacket.
- Salga.

Aunque el código es el mismo en ambos casos, el orden de las instrucciones es diferente. Aquí, la secuencia de tareas depende del valor de temp.

¿Por qué los errores lógicos causan problemas? (de 2 a 3 minutos)

Estos errores son un poco más difíciles de resolver en comparación con los errores de sintaxis. Cuando depuramos errores de sintaxis, usamos la consola de errores para identificar nuestros errores y realizar cambios. Esta es una estrategia sumamente útil para los errores de sintaxis, pero usted no puede confiar en que la consola le diga cuándo usted tiene un error de lógica. En este caso, el código puede ejecutarse, pero no funcionará como usted quiere. Si su código no arroja errores mientras se compila (es decir, convierte su código en un formato que la computadora pueda ejecutar), entonces la computadora piensa que no hay errores. Los errores lógicos crean un problema con el **flujo del programa**: la secuencia o el orden en que se ejecutan sus líneas de código.

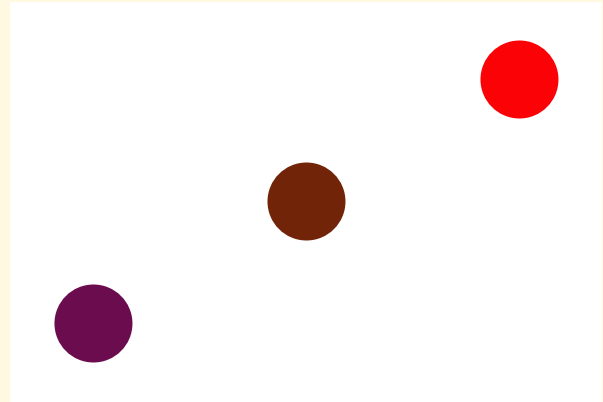
Paso 1: Conocer los errores lógicos (continuación)

Supongamos que quiere pintar círculos de colores rojo, verde y púrpura. Consulte esta instrucción de ejemplo sobre cómo crear su pintura:

CÓDIGO

```
Mojar el pincel en pintura roja
Pintar un círculo en la parte
superior derecha
Mojar el pincel en pintura verde
Pintar un círculo en el centro
Mojar el pincel en pintura púrpura
Pintar un círculo en la parte
inferior izquierda
Enjuagar el pincel
Dejar secar el lienzo
```

RESULTADOS



En nuestro programa actual, el último círculo probablemente sea de un color marrón violáceo. ¿Por qué cree que esto es así? ¿Cómo reescribiría este “programa” para que el último círculo sea púrpura?



Verifique sus ideas en la Guía de referencia (página 2).

Paso 2: Crear estrategias de depuración (de 5 a 10 minutos)

En la parte 1, aprendimos algunas técnicas de depuración, como el uso de la consola, para leer los mensajes de error, observar con detenimiento y probar un cambio realizado. A medida que se avanza hacia un código más complejo, es útil tener una caja de herramientas llena de estrategias de depuración y un sistema para aplicarlas.

La depuración es el ciclo que implica la prueba del código, la definición del problema, la formulación de hipótesis de soluciones y la realización de cambios y de nuevas pruebas de código hasta que el problema se resuelva. Todos los errores son diferentes, pero hay estrategias en común que pueden ser de ayuda para encontrar una solución. En la siguiente página, se incluye una lista de estrategias que usted puede usar cuando un error afecta su código.



Paso 2: Crear estrategias de depuración (continuación)

1. Adoptar la actitud correcta para depurar

Lo más probable es que solo encuentre errores después de mucho tiempo y de haber invertido mucha energía en su código. Quizá lo mejor sería despejar la mente e ir a acariciar a su perro, tomar un refrigerio, dar un paseo o incluso dejar allí el error un rato antes de sumergirse de lleno en él. La depuración debe realizarse lenta y metódicamente. Si usted actúa con rapidez, es probable que se pierda algo o se generen nuevos errores.

2. Volver a las cosas esenciales

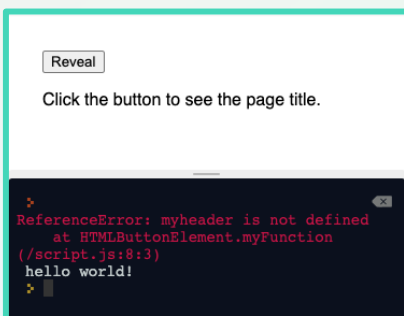
¿Usted está conectada a Internet? ¿Su editor de código o IDE (es decir, el entorno de desarrollo integrado) funciona correctamente? ¿El navegador debe reiniciarse para una actualización? Quizá suene tonto... hasta que todo se arregla.

3. Hacer una copia

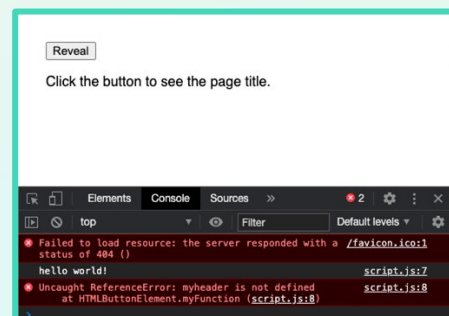
Tratar de arreglar los problemas puede traer aparejados más problemas. Haga siempre una copia si necesita realizar algún cambio importante. Más vale prevenir que curar.

4. Comenzar por lo más sencillo

Verifique los aspectos más sencillos en primer lugar. Se trata de un proceso de eliminación, por lo que hay que quitarse de encima lo más fácil. Busque errores básicos y compruebe si hay errores de sintaxis. Si su IDE tiene una consola de depuración, compruebe si hay mensajes de error. Si está programando para la web, también puede usar [herramientas para desarrolladores](#) en el navegador. A continuación, se muestra un ejemplo de cómo aparece el mismo error en la consola de Repl.it a la izquierda; y si se usan las herramientas para desarrolladores en Chrome, a la derecha.



La consola de Repl.it es la caja negra que se encuentra debajo de la ventana de visualización. Muestra un mensaje de error en rojo con la ubicación del error, y luego imprime **hello world!** en la pantalla.



Las herramientas para desarrolladores aparecen en la pantalla derecha o inferior. La pestaña **Console** (Consola) imprime **hello world!** en la pantalla; luego muestra un mensaje de error en rojo a la izquierda y la ubicación en blanco a la derecha.

5. Buscar patrones

La programación, al igual que otros lenguajes humanos, se basa en patrones como una manera de organizar y dar sentido al código. Podemos usar patrones que hemos visto en códigos anteriores y aplicarlos a un desafío actual. Por ejemplo, supongamos que usted está programando un botón para que disminuya la puntuación cada vez que se haga clic en él. Ya ha codificado dos botones para aumentar la puntuación, pero no está segura de cómo programar el segundo. Puede observar similitudes o patrones en el comportamiento para ayudarle a encontrar una solución.

Haga clic aquí para aumentar en 1.

```
if(clicked == true){  
  score = score + 1;  
}
```

Haga clic aquí para aumentar en 2.

```
if(clicked == true){  
  score = score + 2;  
}
```

Haga clic aquí para disminuir en 1.



6. Describir el problema

A veces, cuando analizamos un problema por nuestra cuenta durante demasiado tiempo, empezamos a pasar por alto lo más sencillo. Sin embargo, si usted tiene que describir el problema a otra persona, deberá ofrecer más detalles. Imagine que le está contando o escribiendo a un amigo sobre su problema y describa:

- ☐ Lo que ya ha hecho.
- ☐ Qué funciona mal.
- ☐ Lo que esperaba o quiere que ocurra.
- ☐ Qué adiciones o cambios realizó en el código desde la última vez que lo probó.

Revise también sus notas de planificación, el pseudocódigo y los diagramas. Lo más probable es que descubra el error por sí sola.

7. Hacerlo observable

Es muy útil observar cómo cambian los valores de las variables en el transcurso del tiempo o cuándo se producen diferentes eventos, como el clic de un botón o la entrada de texto. A menudo, usará estas variables para tomar decisiones sobre el flujo del programa. Por ejemplo:

```
if(zoomCalls >= 5){
    takeWalk();
}
```

Pero, ¿cómo se puede saber cuál es el valor de una variable? ¿Cómo sé la cantidad de `zoomCalls`? Puede usar la función `console.log(variableName)` para imprimir el valor de una variable en la ventana de la consola del IDE o verla con las [herramientas para desarrolladores](#).

CÓDIGO

```
console.log("zoomCalls = " + zoomCalls);
if(zoomCalls >= 5){
    takeWalk();
}
```

CONSOLA

```
zoomCalls = 3
zoomCalls = 4
```

Nota: Si quiere saber qué valor está imprimiendo, puede añadir contexto a sus valores con texto. Ponga el texto que quiera imprimir entre comillas dobles `" "` o simples `' '` seguidas de un signo más `+` y el nombre de la variable. Esto es especialmente útil si quiere imprimir más de un valor a la vez.

Paso 2: Crear estrategias de depuración (continuación)

8. Probar una cosa a la vez

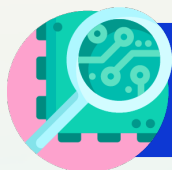
Esta es una situación que definitivamente no queremos: al cambiar algunas “pequeñas cosas” en su código, usted generará un nuevo problema y podrá averiguar si es el problema anterior o los nuevos cambios los que lo causan. Su objetivo es aislar el problema. Cambie una cosa a la vez y luego pruebe.

9. Buscar las interwebs

Hay muchísimos sitios web dedicados enteramente a responder preguntas sobre errores de código. Es muy probable que alguien más se haya encontrado con su problema. Una vez que haya descrito el problema en detalle, use su motor de búsqueda favorito para buscar respuestas. Si no aparece nada, intente replantear o reformular la pregunta: ser capaz de buscar una solución de forma eficiente es una habilidad en sí misma. También puede copiar cualquier mensaje de error y comenzar su búsqueda allí. Si está programando para la web, [W3Schools](#) y [MDN](#) son excelentes recursos. [StackOverflow](#) es un buen sitio para que los desarrolladores que trabajan en todos los idiomas hagan preguntas y aprendan de las soluciones de otras personas. Independientemente de cuál sea el lenguaje de programación, es probable que haya foros y muchos sitios web o publicaciones que se dediquen a brindar información.

10. Preguntar a un amigo

Si después de intentar todo lo anterior sigue sin poder resolver el problema, llame a un amigo, un mentor o un maestro.



Si desea obtener más recursos de depuración, consulte [esta serie de videos](#) de Clay Shirky en NYU y esta fantástica [Guía de campo de depuración](#) de la comunidad p5.js.

Paso 3: Probar el cuestionario (de 3 a 4 minutos)

Ahora que tenemos un práctico kit de herramientas de estrategias de depuración, es el momento de resolver nuestro último error: el error lógico. En primer lugar, tendremos que averiguar qué y dónde está el error antes de pensar en solucionarlo.

Recuerde: La mayoría de las veces, el programa no se equivoca porque está siguiendo las instrucciones que usted le dio. Esto significa que los errores lógicos son, en realidad, el resultado de nuestras propias concepciones erróneas sobre cómo el programa *debería* funcionar en contraposición con cómo funciona *en realidad*. Los errores lógicos son errores en la ejecución de su programa. Antes de ahondar en el tema, probemos el cuestionario para obtener pistas.

Hacer el cuestionario (de 2 a 3 minutos)

Cuando probamos la versión con errores al principio de esta actividad, el cuestionario no devolvió nunca el resultado. Ahora que hemos arreglado nuestros errores de sintaxis, vamos a probarlo de nuevo para asegurarnos de que no haya cambiado nada más.

- ☐ Haga el cuestionario 3 o 4 veces.
- ☐ Responda las preguntas en distintas configuraciones.
- ☐ Use el botón de reinicio después de terminar para reiniciar el cuestionario.

Paso 3: Probar el cuestionario (continuación)

Ahí lo tenemos: ¡aún no arroja resultados! Y no hay más mensajes de error. En este punto, lo mejor es volver, literalmente y en sentido figurado. Si necesita una pausa breve y un refrigerio, este es el momento. En la próxima sección, comenzaremos buscando las pistas mediante la revisión del funcionamiento del código.

You are a...

Restart

Captura de pantalla del texto del resultado en la parte inferior que dice "Usted es..." y un botón de reinicio (botón Restart) debajo

Paso 4: Revisar el código (de 3 a 7 minutos)

Probablemente haya leído el código y los comentarios mientras arreglaba los primeros errores. Pero si no lo hizo, vuelva a hacerlo ahora. No hay problema si no entiende todo lo que está pasando. Nuestro objetivo es manejar el flujo del programa, por lo que solo necesitamos un alto nivel de comprensión en este punto.



Verifique sus ideas en la Guía de referencia (página 2).

Paso 5: Describir el problema (de 5 a 7 minutos)

Empecemos por describir lo que queremos que ocurra. Tome un papel y un bolígrafo o abra un editor de texto y escriba lo que debe hacer el programa cuando una persona hace el cuestionario. El primer paso se incluye a continuación.

- Cuando una persona hace clic en un botón de respuesta, el programa llama a la función de bicho especificada (p. ej., **abeja**, **mariposa**, **saltamontes** o **mariquita**) asociada en el agente de escucha de eventos.



Verifique sus ideas en la Guía de referencia (página 4).

Después de escribir esto, vemos lo importante que es conocer el valor de nuestras variables de puntuación a lo largo del programa. Por ejemplo, si **grasshopperScore** es mayor a 4 y no se muestra el resultado, esa es una gran pista.

Paso 6: Hacerlo observable (de 10 a 15 minutos)

¿Cómo accedemos al valor de nuestras variables? Con `console.log()`, ¡por supuesto! Queremos conocer los valores de cada variable de puntuación de bichos y `questionCount` **cada vez que se actualicen**. Esta última parte es importante porque nos ayuda a saber dónde poner la función `console.log()`. Queremos conocer estos valores después de que una persona haya hecho clic en el botón, por lo que pondremos `console.log()` dentro de cada función de puntuación de los bichos.

- ❑ Use `console.log()` para imprimir la puntuación de la variable y el valor `questionCount` dentro de las funciones a continuación. Añada contexto a sus mensajes usando comillas simples o dobles y un signo más para añadir un texto explicativo que le permita saber qué valor va con qué puntuación (consulte el paso 3 para ver un ejemplo si lo necesita).
 - ❑ función `abeja`
 - ❑ función `mariposa`
 - ❑ función `saltamontes`
 - ❑ función `mariquita`

Consejo: Puede usar `"\t"` para crear un espacio entre los dos valores.



Verifique sus ideas en la Guía de referencia (página 4).

Una vez que haya añadido `console.log()` a sus funciones, será el momento de probar el código.

- ❑ Haga clic en el botón de ejecutar.
- ❑ Haga clic en Mochi para la pregunta uno y, luego, verifique la ventana de la consola.

¡Finalmente, obtuvimos resultados de nuestro programa! En la consola se mostrará el siguiente texto:

```
questionCount = 1          butterflyScore = 1
```

- ❑ Ahora, haga clic en *Ciudad* para la pregunta dos y verifique la consola. Debería aparecer el siguiente texto:

```
questionCount = 1          butterflyScore = 1
questionCount = 2          beeScore = 1
```

¡Luce bien hasta el momento! Nuestros valores se incrementan con cada clic, por lo que sabemos que las funciones de bichos están funcionando.

Paso 6: Hacerlo observable (continuación)

- ❑ Para la tercera pregunta, haga clic en 2018 y verifique la consola. Debería aparecer el siguiente texto:

```
questionCount = 1      butterflyScore = 1
questionCount = 2      beeScore = 1
questionCount = 3      butterflyScore = 2
```

En este punto, deberíamos tener un resultado. El valor `questionCount` es igual a 3 y tenemos una puntuación mayor o igual a 2 (`butterflyScore`), pero el texto resultante no ha cambiado.

Pensemos en estas pistas y hagámonos algunas preguntas:

- ❑ ¿Qué controla el texto resultante?
- ❑ ¿Cuándo llamamos (o usamos) a la función `updateResult`?
- ❑ ¿La sentencia condicional se ejecuta cuando la necesitamos?



Verifique sus ideas en la Guía de referencia (página 5).

Paso 7: Buscar patrones (de 1 a 2 minutos)

Estamos empezando a delimitar el problema a la sentencia condicional de la línea 82:

```
81 // Realizar un seguimiento del final del cuestionario
82 if (questionCount == 3){
83     updateResult();
84 }
```

En este momento, esta sentencia condicional permite probar el valor `questionCount` después de todas las funciones de seguimiento de la puntuación de los errores. Es posible que haya notado un patrón que actualiza la puntuación y que las variables `questionCount` ocurren dentro de la función del bicho. Quizá necesitemos probar el valor de `questionCount` dentro de cada función.

Paso 8: Probar una cosa a la vez (de 5 a 10 minutos)

Probemos esta hipótesis. En lugar de cambiar todas las funciones, intentemos cambiar primero la función abeja. Si eso funciona, entonces podemos cambiar las funciones restantes.

Paso 8: Probar una cosa a la vez (continuación)

- ❑ **Copie toda la sentencia condicional `questionCount` completa.** Asegúrese de incluir todas las llaves.

```
if (questionCount == 3){  
  updateResult();  
}
```

- ❑ **Pegue esto dentro de la función `abeja`.** Debería quedar debajo de la función `console.log()` y encima de la llave de cierre.
- ❑ **Ejecute su código.**
- ❑ **Pruebe su hipótesis seleccionando todas las respuestas de las abejas.** Haga clic en el botón de las respuestas de la abeja para cada pregunta:
 - ❑ P1: Fruta
 - ❑ P2: Ciudad
 - ❑ P3: 2019
- ❑ **Verifique el texto del resultado.** ¿Aparecen los resultados o sigue habiendo un error? Debería notar que obtuvimos el resultado de la abeja, lo que nos hace pensar que tal vez hayamos identificado el problema. Probemos si nuestra hipótesis funciona para los demás resultados.

You are a bee!

Restart

Captura de pantalla del texto del resultado en la parte inferior que dice "¡Usted es una abeja!" y un botón de reinicio (Restart) debajo

Tenemos un resultado que coincide con nuestras elecciones al probar nuestra hipótesis sobre las respuestas de las abejas. Parece que hemos identificado el problema que causa el error.

- ❑ **Añada la sentencia condicional en el mismo lugar de las funciones restantes.** Recuerde *probar una hipótesis a la vez*. Siga los mismos pasos para probar las siguientes funciones:
 - ❑ función `mariposa`
 - ❑ función `saltamontes`
 - ❑ función `mariquita`

Pruebe el programa nuevamente. Responda las preguntas en varias configuraciones y observe qué resultado se obtiene. En la página siguiente, hay una clave de puntuación que será de ayuda para realizar la prueba.



Verifique sus ideas en la Guía de referencia (página 6).

Paso 8: Probar una cosa a la vez (continuación)

Resultado de bicho	abeja	mariposa	saltamontes	mariquita
P1: Postre	Fruta q1a4	Mochi q1a1	Galletas q1a2	Pastel q1a3
P2: Vacaciones	Ciudad q2a2	Playa q2a3	Bosque q2a1	Montañas q2a4
P3: Color	2019 q3a4	2018 q3a2	2020 q3a3	2017 q3a2



¡Felicitaciones!

¡Ha depurado el cuestionario de personalidad con errores correctamente! Ahora que tenemos un cuestionario en pleno funcionamiento, puede pasar a las extensiones para aprender más.

Consejo: ¿Quiere evitar por completo los errores lógicos en el futuro? Dedique tiempo a planificar, diagramar y escribir su código en un lenguaje legible para los humanos (es decir, use un pseudocódigo).

Paso 9: Extensiones (de 5 a 75 minutos)

Extensión 1: Redactar su lista de verificación personal para depuraciones (de 5 a 10 minutos)

Personalice las estrategias anteriores para encontrar un método que le funcione. Cuando empieza a programar y se encuentra con diferentes errores, no es fácil recordar por dónde empezar. Tome un papel y un bolígrafo o abra su editor de textos favorito y cree una lista de estrategias que quiera recordar. Póngala en un lugar de fácil acceso para que sepa dónde recurrir ante el próximo error.





Extensión 2: Deshabilitar los botones (de 15 a 20 minutos)

Ahora, el usuario podría hacer clic en más de una opción de respuesta para una determinada pregunta y arruinar los resultados del cuestionario. Piense en cómo podría deshabilitar los botones de selección de respuesta una vez que se haga clic en uno de ellos.

Tiene un [ejemplo de esta extensión](#) aquí.

What kind of bug are you?

What is your favorite dessert?







Candy

Cookie

Cake

Fruit

Pick a location for your next vacation.



Woods

City

Beach

Mountains

Para comenzar, podría hacer lo siguiente:

- ❑ Crear funciones que deshabiliten todos los botones de respuesta para una pregunta determinada.
- ❑ A continuación, puede añadir otro agente de escucha de eventos para cada botón de opción de respuesta que llama a su función de inhabilitación cada vez que se hace clic en uno de los botones de opción de respuesta.
- ❑ Por último, si añadió una extensión para reiniciar el juego, también deberá asegurarse de volver a habilitar todos los botones como parte de la función de reinicio.

Recursos de extensión

- [Propiedad de desactivación del botón HTML DOM \(modelo de objetos del documento\)](#)
- [Agentes de escucha de eventos](#)
- [Funciones de JavaScript](#)
- [HTML DOM de JavaScript](#)

Extensión 3: Personalizar el cuestionario de personalidad (de 25 a 45 minutos)

Cree su propio cuestionario de personalidad. Es bastante raro que los programadores creen algo completamente de cero. De hecho, es mejor escribir un código que se pueda reutilizar. Intente reutilizar este código para personalizar su cuestionario. Tendrá que actualizar el archivo **index.html** en esta extensión, por eso es útil si ya tiene conocimientos de HTML. Si no los tiene, no hay problema; consulte los Recursos de extensión para obtener ayuda. A continuación, se indican algunos pasos para empezar.

Planificación

Complete esta guía de planificación mientras responde las preguntas de esta sección:

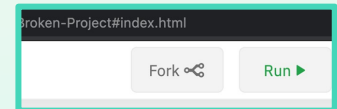
PREGUNTA 1				
Respuestas	1.	2.	3.	4.
Puntuación del resultado	+1	+1	+1	+1
PREGUNTA 2				
Respuestas	1.	2.	3.	4.
Puntuación del resultado	+1	+1	+1	+1
PREGUNTA 3				
Respuestas	1.	2.	3.	4.
Puntuación del resultado	+1	+1	+1	+1

Para comenzar, podría hacer lo siguiente:

- ❑ Elija cuatro posibles resultados finales. Para este cuestionario, elija solo cuatro resultados posibles que su usuario tendrá bien al final. Luego, tendrá que asignar cada respuesta a cada pregunta a uno de los posibles resultados.
 - ❑ Por ejemplo: abeja, mariposa, saltamontes y mariquita.
- ❑ Elija tres preguntas para hacer. Elabore un mapa de las preguntas que hará en su cuestionario. Cada pregunta debe estar relacionada con las demás y debe haber cuatro respuestas posibles.
 - ❑ Por ejemplo: ¿Cuál es su postre favorito?
- ❑ Elija cuatro respuestas posibles para cada pregunta. Cada respuesta debe estar directamente relacionada con uno de los resultados finales del cuestionario.
 - ❑ Por ejemplo: la respuesta 1 a la pregunta 1 es mochi. Mochi se asigna a la mariposa, por lo que al elegir Mochi, **butterflyScore** aumentará en 1.
- ❑ Seleccione la foto que quiere que represente cada respuesta. Guárdela en su computadora. Si tiene un nombre largo, puede ponerle un nombre más corto y descriptivo, como **q1a3-cake.jpg**.
 - ❑ Recuerde que las imágenes de stock gratuitas o de [Creative Commons](#) son excelentes para usar en sus trabajos. Eche un vistazo a las opciones disponibles en sitios como [Unsplash](#), [Pixabay](#), y [Burst](#).

Actualizar el texto y las imágenes en el archivo `index.html`

- ❑ Bifurque el proyecto de Repl.it en el que está trabajando. Cámbiele el nombre y asígnele una descripción.
- ❑ Cargue las imágenes a la carpeta de **recursos digitales** en el panel de archivos a la izquierda.
- ❑ Actualice el título del cuestionario cambiando el texto dentro de las etiquetas `<h1>`.
- ❑ Actualice el texto y las imágenes de la pregunta 1 y sus respuestas usando la guía de planificación anterior.
 - ❑ Reemplace la pregunta actual dentro del primer conjunto de etiquetas `<h2>` con la primera pregunta.
 - ❑ Actualice la imagen de la primera respuesta. Justo debajo de `<div class="answer-choice">`, hay una etiqueta `` que muestra la imagen que usted quiere usar para la primera respuesta a la pregunta 1. Sustituya el enlace existente por un enlace a la imagen que quiere usar. El enlace será **assets/imageName.jpg**.
 - ❑ Actualice el texto del botón. Debajo de la etiqueta ``, debería ver una etiqueta `<button>` con el texto de respuesta. Por ejemplo, `<button id="q1a1">Mochi</button>`. Elimine la respuesta existente y añada la suya.
 - ❑ Repita este proceso para el resto de las respuestas a la pregunta 1.
- ❑ Actualice el texto y las imágenes de las preguntas y las respuestas restantes en el archivo **index.html** usando la guía de planificación anterior.



Actualizar la lógica en el archivo script.js

Hay una lógica integrada en el cuestionario que determina el resultado de una persona según cómo responda a cada pregunta. Como ha notado al revisar el código, cada respuesta se asigna a un resultado específico usando un agente de escucha de eventos. Usted tendrá que actualizar esta lógica basándose en sus propias preguntas, respuestas y resultados. Para hacer una revisión completa de la lógica y de cómo se construye, consulte el paso 4 de la Guía de referencia en la página 4.

- ❑ En la parte de arriba, debajo del comentario `/* Global Variables */`, actualice los nombres de las variables de puntuación con los cuatro posibles resultados del cuestionario.
- ❑ Actualice los nombres de cada variable **everywhere** en el programa. *Si el programa no funciona, este es el primer lugar donde debería revisar.*
- ❑ Actualice los nombres de las funciones en cada agente de escucha de eventos. El nombre debería ser el resultado al que se asigna la respuesta. Por ejemplo, si la respuesta a `q1a1` se asigna al resultado del helado, entonces usted podría nombrar la función `iceCream`.
- ❑ Sustituya los nombres de las funciones actuales por los que acaba de crear en los agentes de escucha de eventos. *Asegúrese de que coincidan con las variables de puntuación correctas.*

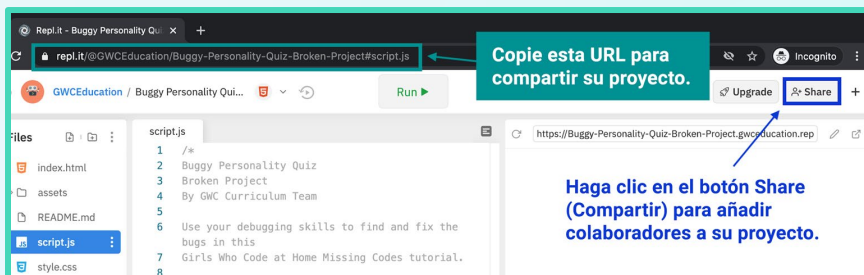
Recursos de extensión

- [Introducción a HTML](#)
- [Comparta sus habilidades de Girls Who Code en casa](#) en una introducción práctica a HTML
- [Etiquetas de encabezamiento \(p. ej., <h1>\)](#)
- [Etiqueta](#)
- [Etiqueta <button>](#)
- [Agentes de escucha de eventos](#)
- [Funciones de JavaScript](#)
- [HTML DOM de JavaScript](#)

Paso 10: Comparta su proyecto de Girls Who Code en casa (5 minutos)

Acceso de solo lectura (de 1 a 2 minutos)

Compartir su trabajo en Repl.it es fácil. Solo tiene que copiar y pegar la dirección URL de su proyecto Repl en la barra de direcciones web de la parte superior. Esta acción permitirá que otros ejecuten su proyecto, vean su código y bifurquen el proyecto y lo remezclen por su cuenta. Nos encantaría ver su trabajo de depuración, y sabemos que a las demás personas también les encantaría. Comparta con nosotros su código arreglado.



Comparta este enlace en sus cuentas de redes sociales y recuerde usar las etiquetas `@girlswwhocode` `#codefromhome` para que podamos mencionarla en nuestra cuenta.

Paso 10: Compartir su proyecto de Girls Who Code (continuación)

Agregar colaboradores (2 minutos)

Si quiere trabajar con un grupo de amigos en un proyecto, puede invitarlos fácilmente a colaborar mediante el botón de compartir (botón **Share**), que se encuentra en la parte superior derecha de la ventana. Debería aparecer una nueva ventana con dos opciones para invitar a otros a colaborar en su proyecto.

- ❑ **Invitar por correo electrónico o nombre de usuario de Repl.it.** Esta opción permite compartir su proyecto con personas específicas escribiendo la dirección de correo electrónico o el nombre de usuario de Repl.it si estas personas ya tienen cuenta en Repl.it. Se recomienda esta opción para que usted se asegure de que compartirá su proyecto con las personas adecuadas.
- ❑ **Comparta el enlace de invitación.** En la parte inferior de la ventana, hay un enlace de invitación único. Puede copiar y pegar este enlace para sus amigos y, de esa manera, accederán a su proyecto.

Nota acerca de colaboradores: Recuerde que, si agrega colaboradores, le dará a otros acceso de edición a su proyecto. De esta manera, los colaboradores podrán cambiar su código, el nombre y la descripción. **NO comparta el enlace de invitación en las redes sociales.** Seleccione con cuidado con quién comparte los derechos de edición.



Siga en contacto para recibir más actividades de Girls Who Code en casa.

