



Girls Who Code en casa

Depuración de código faltante, parte 1

Errores de sintaxis

Descripción de la actividad

La depuración es una de las habilidades más importantes que puede desarrollar un programador. Y no tiene nada que ver con los insectos. Los errores o bichos ("bug" en inglés) son errores de un programa que impiden que ese programa funcione como usted quiere. Esos errores pueden ser tan pequeños como una falta de ortografía o tan grandes como la escritura de un código desordenado. Los programadores, independientemente de los distintos niveles de experiencia, cometen errores en todos los lenguajes de programación. Los errores se producen incluso en el hardware, es decir, los circuitos que ejecutan el software. Los errores pueden ser frustrantes cuando uno no logra resolverlos y esclarecedores cuando finalmente uno se da cuenta del problema.

Si la **depuración** es el proceso que implica quitar errores de software, entonces la **programación** debe ser el proceso que implica colocar esos errores.

~ Edsger Dijkstra

En la parte 1 de esta actividad, se hace una introducción sobre el tipo de error más frecuente: los errores de sintaxis. Trabajaremos en conjunto para resolver dos errores de sintaxis en el cuestionario de personalidad con errores. Si bien trabajaremos con JavaScript, los métodos se aplican a todos los lenguajes. ¿No conoce JavaScript? No se preocupe, **no** tiene por qué saber cómo completar esta actividad.

Objetivos del aprendizaje

Al finalizar esta actividad, será capaz de:

- ❑ describir por qué la depuración es una parte importante del aprendizaje de la programación;
- ❑ identificar diferentes tipos de errores de sintaxis que es posible encontrar en el código;
- ❑ depurar los errores de sintaxis en un sitio web dañado.

Materiales

- Editor [Repl.it](https://repl.it)
- [Cuestionario de personalidad, proyecto modelo](#)
- [Cuestionario de personalidad, proyecto con errores](#)
- [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.

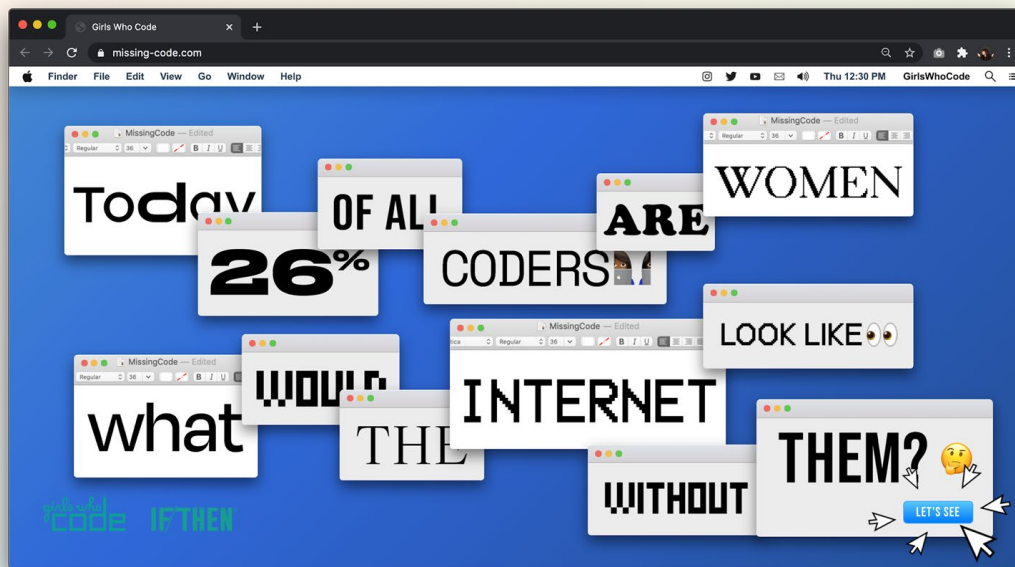
Si quiere obtener más información sobre JavaScript, consulte la actividad [Excursión virtual](#) de Girls Who Code en casa.

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

Es la Semana de la Educación en Ciencias de la Computación, por eso nos hemos asociado con IF/THEN® para rendirles homenaje a las mujeres involucradas en la tecnología. Internet, tal y como la conocemos, no sería posible sin el aporte de las mujeres en tecnología. Sin embargo, a pesar de que el 26 % de los programadores son mujeres, existe una percepción obstinada de que los productos que usamos a diario están hechos por hombres. ¡Equivocado!

Consulte www.missing-code.com para ver qué pasaría con sus plataformas favoritas si existiera un mundo donde las mujeres no codificaran.



Reflexión

Tómese un momento para conocer mejor los aportes a la informática de las personas que se identifican como mujeres. Mientras explora el sitio web de Missing Code, elija un hecho sobre el que quiera aprender más y dedique algunos minutos a investigarlo. Cuando termine, reflexione acerca de la pregunta a continuación.



PROPÓSITO

Internet no sufrirá daños si las mujeres y las personas que se identifican como mujeres dejan de codificar, pero habrá otros efectos negativos. ¿Cómo afecta esta falta de representación en las ciencias de la computación a las personas y las comunidades?

Comparta sus respuestas con un familiar o amigo. Anime a otras personas a leer más sobre los aportes de las mujeres y de las personas que se identifican como mujeres en los campos de las ciencias de la computación.



Paso 1: Darles la bienvenida a los errores (de 3 a 5 minutos)

Cierre los ojos y piense en alguna ocasión en la que haya tenido un problema que no haya sabido resolver, ya sea lograr un nuevo baile viral de TikTok, hacer bien su receta de galletas favorita o construir la morada perfecta en Animal Crossing. Piense en el proceso que ha seguido para descubrir ese problema. Piense en las emociones que sintió y en la actitud que adoptó en el camino. Ahora, con los ojos aún cerrados, recuerde lo maravilloso que fue encontrar una solución.

Abra los ojos. Con todas esas emociones todavía en el aire, aprovechamos esta oportunidad para decirle que **se producirán errores en su código**.



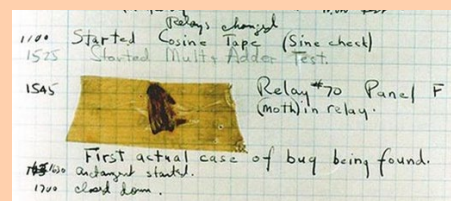
No importa si está empezando o si lleva años programando en tres lenguajes. Los programas se rompen. Cuando hablamos de “romper”, nos referimos a que los programas no funcionen como es debido. No se trata de que su computadora vaya a explotar. Esto puede parecer un fastidio, pero no lo es. Resolver estos errores es la mejor manera de aprender.

Aceptar los errores propios (1 minuto)

En inglés, estos errores se denominan **bugs**. El proceso de localizar y arreglar estos errores se denomina **depuración**.

Los errores o “bugs” no son simples errores en el código. Constituyen un problema técnico en su programa de software Y una brecha entre lo que usted quería que sucediera y lo que realmente sucedió. Precisamente por eso los errores son tan frustrantes, pero también útiles. Encontrar y arreglar un error significa que usted tiene la oportunidad de actualizar su programa y mejorar su comprensión sobre cómo escribir un programa.

Ya sabemos que los errores ocurren más allá del nivel de experiencia. Así que, dado que nos encontraremos con viejos y nuevos errores durante bastante tiempo, es importante desarrollar un proceso para resolverlos.



Fuente: [National Geographic](#)

Se llaman “bugs” (bichos en español) porque en la época en que las computadoras estaban hechas de tubos de vacío, los insectos de verdad podían meterse dentro de los tubos y causar un mal funcionamiento. En esta actividad, abordaremos los errores en el software, pero también puede haber errores en el hardware.

Paso 2: Reconocer los propios errores (de 7 a 10 minutos)

El primer paso en la depuración es saber cuándo hay un error. Pero, ¿qué ocurre realmente cuando hay uno? ¿Qué forma toma? ¿Mi proyecto quedará destruido para siempre?! (Muy, pero muy improbable). Comencemos con estas preguntas.

¿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.

¿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.

```
// La variable no está declarada: var myNumber = 1;
myNumber = 1;

// No hay ninguna llave de cierre en la sentencia condicional
if(myNumber > 10){
  addOne();

// El nombre de la variable está mal escrito dentro de la función.
function addOne(){
  myNumbr += 1;
}
```

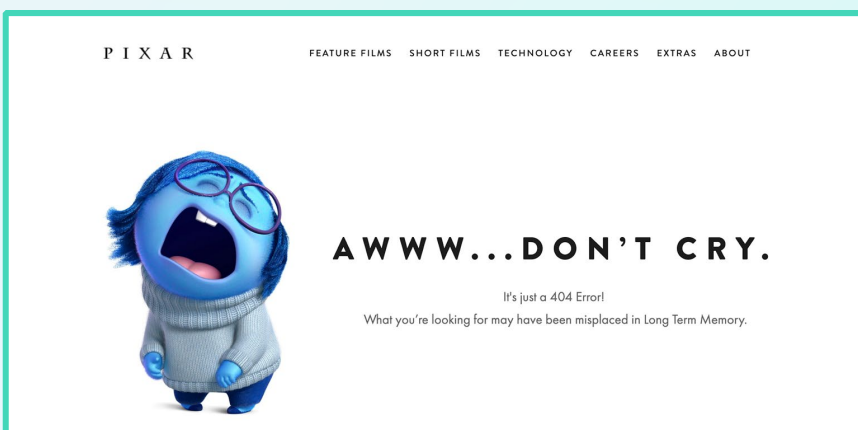
¿Qué sucede si hay un error? (1 minuto)

Todos los errores afectan la funcionalidad de un programa. Algunos errores son pequeños y algunos son grandes. Los errores serían mucho más divertidos si hicieran que las imágenes se estiraran, los botones se cayeran de la pantalla y el texto volara, pero el 99.9 % de las veces no es así.

En cambio, un error podría hacer lo siguiente:

- hacer que un botón deje de funcionar en un sitio web;
- evitar que se muestre una imagen;
- enviar a una persona a la página equivocada de un sitio web;
- dejar de registrar la puntuación de un jugador cuando alcanza un valor determinado;
- hacer que un sitio web se muestre de forma incorrecta en uno o más navegadores.

Y, por supuesto, un error grande puede hacer que sitios web enteros, juegos y demás dejen de funcionar porque el programa no puede compilar o ejecutarse como debería.



Si alguna vez vio la página 404, quiere decir que ha visto un error en acción.

Fuente: [Pixar](#)

¿Hay distintos tipos de errores? (de 5 a 7 minutos)

¡Sip! Clasificar los errores es útil, puesto que con la identificación ya se tiene la mitad de la batalla ganada. Hay **errores de sintaxis** y errores lógicos. En esta actividad, nos centraremos en los errores de sintaxis, y abordaremos los errores lógicos en la siguiente actividad.

Errores de sintaxis

Quizá haya escuchado la palabra **sintaxis** en referencia a la gramática. ¿Qué tiene que ver eso con la programación? ¡Mucho! La sintaxis de un lenguaje de programación es como la sintaxis de cualquier otro lenguaje. Hay reglas que las letras (por ejemplo, b y B), los números (por ejemplo, 11 o 3.14) y los símbolos (por ejemplo, { } o #) deben seguir para procesar y ejecutar un programa. Los distintos lenguajes tienen una sintaxis diferente.

Por ejemplo, JavaScript es un lenguaje de programación que se usa principalmente para agregar interactividad a los sitios web, y es el lenguaje que usaremos en esta actividad. Veamos dos de las reglas sintácticas más reconocidas en JavaScript:

1. **JavaScript usa llaves { } y paréntesis () para separar las partes de un código, como funciones o sentencias condicionales.** Otros lenguajes, como Python, usan la sangría en lugar de las llaves para separar las partes de un código.
2. **JavaScript usa punto y coma ; para finalizar una sentencia en los códigos.** Una sentencia es una instrucción que se le da al programa, como `console.log(score);`. Del mismo modo que usted debe usar un punto al final de una frase, debe terminar sus sentencias con un punto y coma.

A continuación, se incluyen algunas pautas que pueden servirle para evitar los errores de sintaxis más frecuentes:

Muchos errores de sintaxis son errores de ortografía.

Por ejemplo, usamos variables para almacenar datos y modificarlos o compararlos con otros valores cuando es necesario. Como los usará repetidamente, hay más probabilidades de que los escriba mal en algún momento.

```
var mothScore = 0;  
if(motScore > 2)  
mothScre = 10;
```

Las variables, los nombres de las funciones y las palabras clave distinguen entre mayúsculas y minúsculas.

No se pueden intercambiar letras minúsculas y mayúsculas.

```
var bee no es lo mismo que var Bee.
```

```
function no es lo mismo que Function.
```

Si lo abre, debe cerrarlo.

Es fácil perder o agregar una llave o paréntesis adicional cuando se agrupan partes de código. Recuerde que todas las llaves y los paréntesis deben tener un compañero.

```
function restartQuiz(){  
  if(score > 10){  
    score = 0;  
  }  
}  
  
function restartQuiz(){  
  if(score > 10){  
    score = 0;  
  }
```


Si desea obtener más información sobre la sintaxis de JavaScript, consulte [esta guía](#) de Tania Rascia.

Paso 3: Examinar el sitio con errores (bugs) (de 3 a 5 minutos)

Antes de depurar algo, tenemos que entender el problema. La mejor manera de empezar este proceso es probar el programa. Primero, hará el cuestionario en el sitio de funcionamiento. Luego, comprobará la versión con errores para ver cómo se comporta en comparación con la versión que sí funciona.

What kind of bug are you?

What is your favorite dessert?




Mochi Cookies Cake Fruit

Pick a location for your next vacation.



Woods City Beach Mountains

Which Color of the Year do you prefer?



PAINTBOX 10-2020 T100 Greenery PAINTBOX 10-2020 T101 Sherry Vines PAINTBOX 10-2020 T102 Cosmic Blue PAINTBOX 10-2020 T103 Living Coral


2017 2018 2020 2019

You are a...

Restart


What kind of bug are you?

What is your favorite dessert?




Mochi Cookies Cake Fruit

Pick a location for your next vacation.



Woods City Beach Mountains

Which Color of the Year do you prefer?



PAINTBOX 10-2020 T100 Greenery PAINTBOX 10-2020 T101 Sherry Vines PAINTBOX 10-2020 T102 Cosmic Blue PAINTBOX 10-2020 T103 Living Coral

2017 2018 2020 2019

You are a grasshopper!

Restart

Hacer el cuestionario de personalidad arreglado (de 3 a 5 minutos)

Siga el enlace a la [versión arreglada del cuestionario](#). Ahora, haga el cuestionario para ver qué bicho es usted. Cuando termine, piense en las siguientes preguntas. Anote sus ideas si le resulta útil.

- ☐ ¿Qué medidas ha tomado para obtener ese resultado?
- ☐ ¿Cómo cree que el programa averiguó qué bicho es usted?



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

Hacer el cuestionario con errores (de 1 a 2 minutos)

Veamos cómo se compara el cuestionario con errores. Siga [este enlace a la versión con errores](#) y pruébela. Haga el cuestionario varias veces para ver cómo se comporta el programa. Intente responder las preguntas con diferentes configuraciones para ver si se produce alguna modificación. Si todo está correctamente “roto”, el cuestionario nunca devolverá el resultado.

Paso 4: Comenzar con Repl.it (de 5 a 15 minutos)

Para esta actividad, usaremos el editor web Repl.it. [Repl.it](https://repl.it) es un editor gratuito, colaborativo y basado en el navegador que admite múltiples lenguajes de programación. Esta potente herramienta le permite incluso codificar y hablar con un grupo de amigos al mismo tiempo.

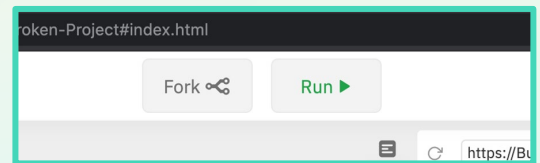
Crear una cuenta (de 5 a 7 minutos)

- ❑ **Suscríbase o inicie sesión en Repl.it.** Para guardar su trabajo, deberá crear una cuenta. Siga las instrucciones que aparecerán en el formulario de inscripción para crear una cuenta. Si usted es menor de 13 años de edad, necesitará la dirección de correo electrónico de uno de sus padres para suscribirse.
- ❑ **Siga las instrucciones para crear una cuenta.** Puede optar por suscribirse con su cuenta de Google, GitHub o Facebook para acceder más rápidamente.

Bifurcar el cuestionario de personalidad con errores (de 2 a 3 minutos)

- ❑ **Abra el [cuestionario de personalidad con errores](#).**
- ❑ **Cree una copia del cuestionario de personalidad con errores haciendo clic en el botón de bifurcación junto al botón de ejecutar.** Una bifurcación duplica todo el proyecto y lo añade a su cuenta de Repl como un nuevo proyecto.

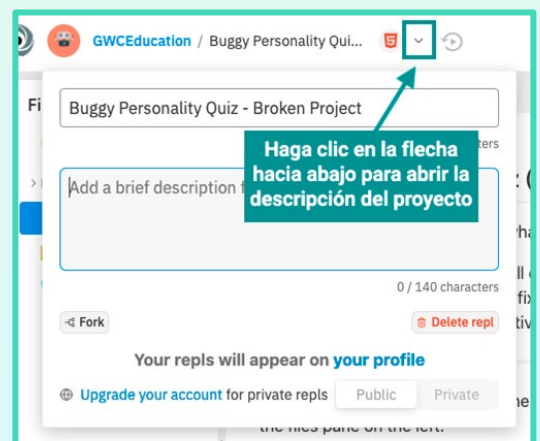
Usted puede bifurcar el código fuente de otras personas o puede bifurcar sus propios proyectos si, por ejemplo, está arreglando errores, pero quiere tener una versión anterior de su código a la que pueda volver en caso de que cree más errores mientras lleva a cabo el proceso de depuración.



Agregar una descripción del proyecto (de 1 a 2 minutos)

Ahora que tiene su copia del cuestionario con errores, vamos a añadir una breve descripción del proyecto a Repl.

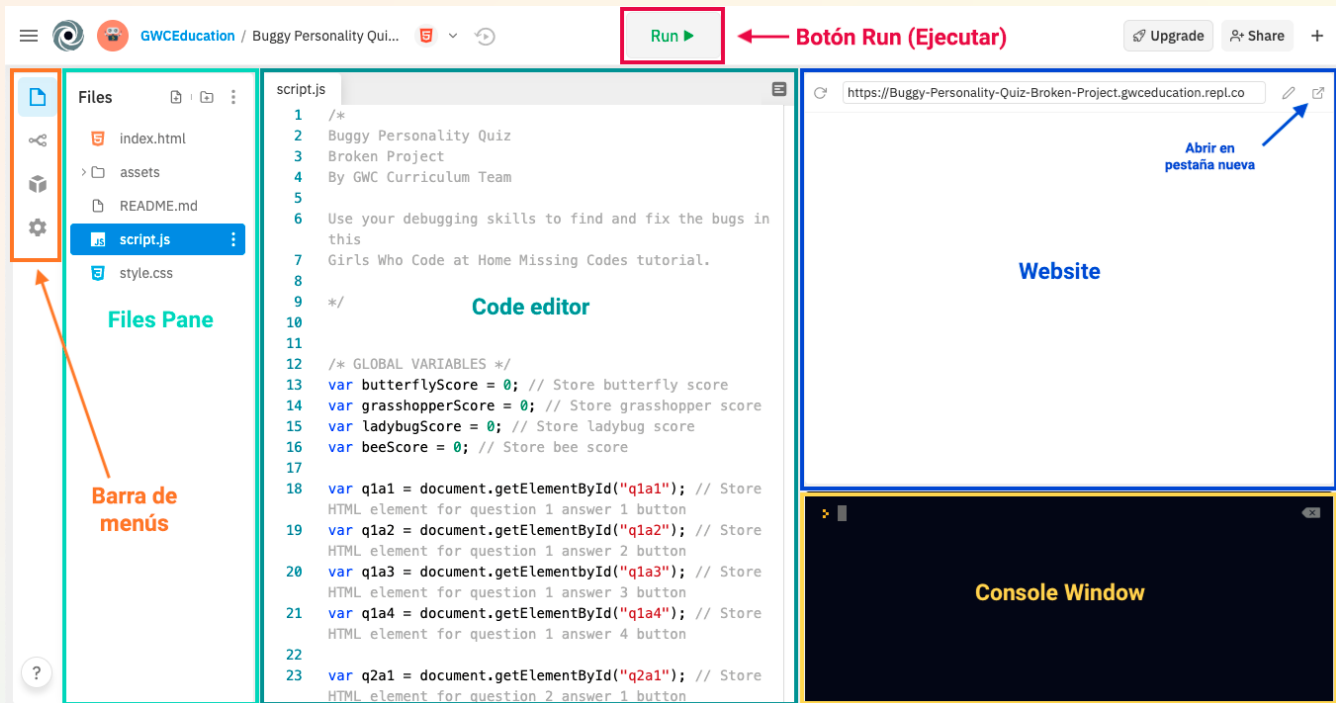
- ❑ **Abra la descripción del proyecto.** Localice el nombre de su proyecto en la parte superior izquierda de la pantalla. Haga clic en la pequeña flecha hacia abajo a la derecha del nombre. Debería abrirse una nueva ventana con el nombre de su proyecto y un área para añadir una breve descripción.
- ❑ **Añada una breve descripción.** En la descripción, usted podría incluir algo como lo siguiente:
 - ❑ **Descripción general:** ¿Cómo se supone que funciona?
 - ❑ **Instrucciones:** ¿Hay instrucciones específicas para ejecutar su proyecto?
 - ❑ **Atributos:** ¿Obtuvo ayuda de otras personas o recursos adicionales? Asegúrese de dar a conocer a estas personas y estos recursos.



Obtenga más información sobre Repl.it en la Guía de referencia (páginas 3 y 4).

Explorar la vista del editor (de 3 a 5 minutos)

Veamos la vista del editor de Repl.it. Ahora que hemos creado un nuevo proyecto, tenemos que entender dónde codificar; cómo navegar entre los archivos y recursos digitales; y cómo guardar y ejecutar el código.



- **Panel de archivos.** Esta ventana muestra todos los archivos del proyecto. Para esta actividad, solo trabajaremos en un archivo, script.js, pero usted también verá los archivos index.html, style.css y README.md. Haga clic en el archivo README para obtener más información sobre lo que hace cada uno de estos archivos.
- **Barra de menú.** Esta barra le permitirá cambiar la vista en el panel de archivos. Entre algunas de las opciones, se incluyen cambiar el control de versiones, añadir paquetes y actualizar la configuración. En los ajustes, puede personalizar el diseño, el tema, el tamaño de la letra y otros ajustes del texto.
- **Editor de código.** Aquí es donde escribirá su código.
- **Botón de ejecutar.** Después de realizar los cambios en el código, haga clic en el botón de ejecutar en la parte superior del editor. Debería ver el resultado en la ventana de salida/console de la derecha.
- **Sitio web.** Esta ventana muestra su sitio web. Si quiere verlo como una página web completa, haga clic en el botón de nueva pestaña en la esquina superior derecha.
- **Ventana de la consola.** Muestra el resultado del código. Todos los resultados estarán visibles. Por eso, si hace clic en el botón de ejecutar varias veces, cada resultado se mostrará aquí.

Habrá notado que no hay un botón de **guardar** en Repl.it. Mientras usted tenga una conexión a Internet, todos los cambios en el código se guardarán automáticamente. Una vez que haga clic en el botón de ejecutar, su Repl se guardará; por eso, **asegúrese de ejecutar siempre el código antes de cerrar el editor.**

Paso 5: Arreglar el primer error (de 5 a 7 minutos)

En el paso 3, observamos cómo se comportaba nuestro cuestionario con errores y hemos descrito cómo se supone que usted debe registrar la puntuación de los errores de una persona. Ahora es el momento de revisar el código real del archivo **script.js** para ver dónde están los problemas.

Presione el botón de ejecutar (botón *Run*) en el centro de la barra de navegación superior. Observe la consola en el lado inferior derecho de la pantalla.

SyntaxError: Unexpected end of input at /script.js:111:1

¡Tenemos nuestro primer mensaje de error! Los mensajes de error son lo mejor. Nos dan una pista de cuál es el problema y nos ayudan a localizarlo. Deténgase un momento e intente descifrar este mensaje:

- **SyntaxError** significa que un problema con la sintaxis (es decir, la ortografía, la puntuación, el formato, etc.) está causando el error.
- **Unexpected end of input** significa que la computadora no sabía dónde terminar. Este es un mensaje de error frecuente que se obtiene si usted se olvida cerrar una llave `{ }` en algún lugar del código.
- **at /script.js** significa que el error está en el archivo **script.js**.
- **111:1** indica que el error está en la línea 111 en el primer espacio.

Ahora, contamos con mucha información con la que podemos trabajar. Comencemos con la línea 109. Es raro. Parece que ya tenemos una llave ahí. Intente hacer clic en la llave y ver qué pasa.

```
101 // Reiniciar cuestionario
102 function restartQuiz() {
103     result.innerHTML = "Usted es...";
104     questionCount = 0;
105     beeScore = 0;
106     butterflyScore = 0;
107     grasshopperScore = 0;
108     ladybugScore = 0;
109 }
```

Haga clic en la llave de cierre.

Consejo útil: En la mayoría de los IDE, cuando se mueve el cursor junto a una llave o paréntesis, se resalta la llave o el paréntesis asociado con el código existente. Al usar esta técnica, sabemos lo siguiente:

1. La llave de cierre de la línea 109 ya tiene su compañera en la línea 102.
2. No hay otras llaves abiertas en la función `restartQuiz`.

Esto significa que necesitamos nuestros sombreros de detective.

Paso 5: Arreglar el primer error (continuación)

Siga los pasos a continuación y compruebe el código en la Guía de referencia:

- ❑ **Busque cerca.** Si el error no aparece directamente en el lugar donde el mensaje decía que lo haría, intente buscar cerca. No hay código después de la función `reStartQuiz`, así que examinemos la función `updateResult` que está justo antes en la línea 87:

```
110 // Actualizar resultado del cuestionario
111 function updateResult() {
112     if (beeScore >= 2) {
113         result.innerHTML = "¡Usted es una abeja!";
114     } else if (butterflyScore >= 2) {
115         result.innerHTML = "¡Usted es una mariposa!";
116     } else if (grasshopperScore >= 2) {
117         result.innerHTML = "¡Usted es un saltamontes!";
118     } else if (ladybugScore >= 2) {
119         result.innerHTML = "¡Usted es una mariquita!";
120     } else {
121         result.innerHTML = "Mmm...tengo dudas. Inténtelo de nuevo
122         más tarde".;
123     }
```

- ❑ **Observe con detenimiento.** ¿Qué es lo primero que nota? MUCHAS llaves. Esta es una ubicación privilegiada para los errores. La función `updateResult` contiene una larga sentencia condicional que evalúa la puntuación de una persona y devuelve qué error es.
- ❑ **Piense en lo que usted necesita.** En este punto, es útil retroceder y pensar en lo que usted sabe que necesita. Una llave abierta al principio de la función y una llave de cierre al final. Si no hubiera nada dentro, se vería así:

```
function updateResult(){
}
```

- ❑ **Identifique las herramientas con las que cuenta.** ¿Qué herramientas tiene o conoce para resolver esto? Anteriormente, aprendimos que podemos usar el cursor para mostrar la compañera de una llave. Intentémoslo.
- ❑ **Pruébelo.** Compruebe cada corchete en la función `updateResult` para ver cuál se asocia con la primera llave abierta.
- ❑ **Realice los cambios necesarios.** Solo necesita realizar un cambio.
- ❑ **Ejecute el programa.** Pruebe el programa para ver si se ejecuta sin errores.



Compruebe el código en la Guía de referencia (páginas 4 y 5).

Paso 6: Resolver el segundo error (de 3 a 5 minutos)

Resolvimos el primer error de forma triunfal; ahora vamos a ver el segundo mensaje de error:

`TypeError: document.getElementById is not a function at /script.js:20:21`

¡Tenemos nuestro primer mensaje de error! Los mensajes de error son lo mejor. Nos dan una pista de cuál es el problema y nos ayudan a localizarlo. Deténgase un momento e intente descifrar este mensaje:

- **TypeError** significa que el programa no puede ejecutar la operación porque un valor no es el tipo de valor que la computadora esperaba. En este caso, el valor es el nombre de una función.
- `document.getElementById` es el nombre de la función que está causando el error.
- **is not a function** significa que la computadora no reconoce esto como una función.
- **at /script.js** significa que el error está en el archivo **script.js**.
- **20:21** indica que el error está en la línea 20 en el espacio 21.

Si nos basamos en lo aprendido en el último paso, sabemos que el error está en el archivo **script.js** en la línea 20.

```
var q1a3 = document.getElementById("q1a3"); // Almacenar el elemento HTML para el botón de
pregunta 1, respuesta 3
```

El mensaje de error dice que `document.getElementById` no es una función. Pero sabemos que se trata de una función, porque la usamos en muchos otros lugares. Entonces, ¿qué es lo que está mal? Siga estos pasos para averiguarlo:

- ❑ **Busque patrones.** ¿Observa alguna diferencia en los otros lugares donde se usa?

```
18 var q1a1 = document.getElementById("q1a1"); // Almacenar el elemento
    HTML para el botón de pregunta 1, respuesta 1
19 var q1a2 = document.getElementById("q1a2"); // Almacenar el elemento
    HTML para el botón de pregunta 1, respuesta 2
20 var q1a3 = document.getElementById("q1a3"); // Almacenar el elemento
    HTML para el botón de pregunta 1, respuesta 3
21 var q1a4 = document.getElementById("q1a4"); // Almacenar el elemento
    HTML para el botón de pregunta 1, respuesta 4
```

- ❑ **Intente realizar cambios.** Si cree que sabe dónde está el problema, arréglolo.
- ❑ **Ejecute el programa.** Pruebe el programa para ver si se ejecuta sin errores.



Compruebe el código en la Guía de referencia (página 6).



¡Felicitaciones!

Ha depurado correctamente todos los errores de sintaxis. Cuando ejecute el programa, ya no debería ver un mensaje de error. Sin embargo, si ha intentado hacer el cuestionario, se habrá dado cuenta de que todavía no funciona. Esto se debe a que queda un error: un error lógico. Estos errores se producen porque hay un problema con el flujo del programa o el orden en que se ejecutan sus líneas de código. En la siguiente actividad, exploraremos un conjunto de estrategias de depuración que usted puede usar para arreglar este error y cualquier otro que pueda encontrar en el futuro.

Paso 9: 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 @girlswhocode #codefromhome para que podamos mencionarla en nuestra cuenta.**

Copie esta URL para compartir su proyecto.

Haga clic en el botón Share (Compartir) para añadir colaboradores a su proyecto.

```
1 /*
2 Buggy Personality Quiz
3 Broken Project
4 By GWC Curriculum Team
5
6 Use your debugging skills to find and fix the
7 bugs in this
8 Girls Who Code at Home Missing Codes tutorial.
```


Paso 9: 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 información sobre Depuración de código faltante, parte 2.

