



Girls Who Code At Home

गुम कोड को डीबग करें: भाग 1
सिंटेक्स बग्स।

गतिविधि अवलोकन

डीबगिंग सबसे महत्वपूर्ण कौशलों में से एक है जिन्हें एक प्रोग्रामर विकसित कर सकता है। और इसका कीड़ों से कोई लेना-देना नहीं है। बग्स आपके प्रोग्राम की त्रुटियां होती हैं जो इसे उस तरह से काम करने से रोकती हैं जैसा आप इसे चाहते हैं। वे वर्तनी की गलती जैसी सरल भी हो सकती हैं या आपके कोड के आउट ऑफ ऑर्डर लिखने जैसी बड़ी भी हो सकती हैं। बग्स प्रत्येक प्रोग्रामिंग भाषा में सभी अनुभव स्तरों के प्रोग्रामर के लिए होते हैं। बग्स हार्डवेयर में भी होते हैं (यानी आपके सॉफ्टवेयर को चलाने वाले सर्किट में)। जब आप इसका पता नहीं लगा सकते हैं तो बग्स परेशान करने वाले हो सकते हैं और जब आखिरकार आपको समस्या का एहसास होता है तो वे ज्ञानवर्धक भी हो सकते हैं।

अगर **डीबगिंग** सॉफ्टवेयर बग्स को हटाने की प्रक्रिया है तो **प्रोग्रामिंग** उन्हें डालने की प्रक्रिया होनी चाहिए।

~ Edsger Dijkstra

इस गतिविधि के भाग 1 से आप बग के सबसे सामान्य प्रकार से परिचित होंगे: सिंटेक्स बग्स। हम टूटी हुई बग वाली व्यक्तित्व प्रश्नोत्तरी में दो सिंटेक्स त्रुटियों को हल करने के लिए एक साथ काम करेंगे। हम जावास्क्रिप्ट के साथ काम करेंगे, लेकिन पद्धति सभी भाषाओं पर लागू होती है। जावास्क्रिप्ट नहीं जानते? चिंता न करें, आपको इस गतिविधि को पूरा करने के लिए इसे जानने की आवश्यकता **नहीं** है।

सीखने के लक्ष्य

इस गतिविधि को पूरा कर लेने पर आप निम्नलिखित कर सकेंगे...

- ❑ वर्णन करें कि प्रोग्राम करना सीखने का एक महत्वपूर्ण हिस्सा डीबगिंग करना क्यों है।
- ❑ विभिन्न प्रकार के सिंटेक्स बग्स की पहचान करें जिनसे आप अपने कोड में सामना कर सकते हैं।
- ❑ एक टूटी हुई वेबसाइट में सिंटेक्स त्रुटियों को डीबग करें।

सामग्रियां

- [Repl.it](https://repl.it) एडिटर
- [बग वाला व्यक्तित्व प्रश्नोत्तरी नमूना प्रोजेक्ट](#)
- [बग वाला व्यक्तित्व प्रश्नोत्तरी - टूटा हुआ प्रोजेक्ट](#)
- [मिसिंग कोड संदर्भ मार्गदर्शिका](#)

पूर्व ज्ञान

इस प्रोजेक्ट से शुरुआत करने से पहले, हमारी सलाह है कि आप:

- किसी भी प्रोग्रामिंग भाषा में [चर राशि](#), [कार्यों](#), तथा [कंडीशनल बयान](#) सहित कोर कम्प्यूटेशनल अवधारणाओं से कुछ परिचित रहें।
- शुरुआती स्तर वालों को किसी टेक्स्ट आधारित भाषा, जैसे जावास्क्रिप्ट (JavaScript), Python, स्विफ्ट (Swift) आदि के उपयोग का अनुभव करवाएं।

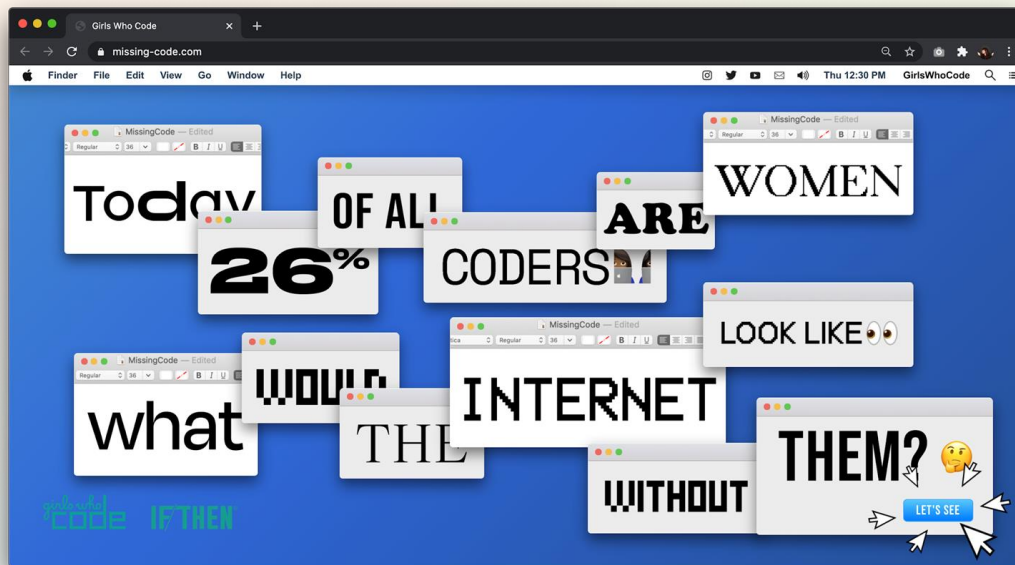
यदि आप जावास्क्रिप्ट में प्रोग्रामिंग के बारे में अधिक जानना चाहते हैं, तो होम में [Girls Who Code](#) [वर्चुअल हाइक](#) गतिविधि देखें।

यदि आप स्क्रैच में डीबगिंग का अभ्यास करना चाहते हैं, तो आप [स्क्रैच के साथ ग्रेव नॉट परफेक्ट डीबगिंग](#) घर पर Girls Who Code गतिविधि देख सकते हैं।

वुमन इन टेक स्पॉटलाइट

यह कंप्यूटर साइंस एड सप्ताह है और हमने वुमन इन टेक्नोलॉजी का जश्न मनाने के लिए हमने IF/THEN® की सहायता ली है। आज हम जिस इंटरनेट से परिचित हैं वह वुमन इन टेक के योगदानों के बिना संभव नहीं था। फिर भी, इस तथ्य के बावजूद कि 26% कोडर महिला हैं, लोग इस बात पर अड़े हैं कि हम जिन उत्पादों का रोज उपयोग करते हैं उन्हें पुरुषों ने बनाया है। गलत!

यह देखने के लिए कि अगर महिलाएं कोड नहीं करती हों तो आपके पसंदीदा मंचों का क्या होगा, www.missing-code.com पर जाएं।



झलक

उन योगदानों के बारे में अधिक जानने में कुछ समय बिताएं जो महिलाओं के सदृश लोगों ने कंप्यूटर साइंस में किए हैं। जब आप मिसिंग कोड वेबसाइट का अन्वेषण करें, तब एक तथ्य चुनें जिसके बारे में आप अधिक जानना चाहते हैं और उसकी खोजबीन करने में कुछ समय बिताएं। जब आप काम पूरा कर लें, तब नीचे दिए प्रश्न पर चिंतन करें।



प्रयोजन

यदि महिलाएं और महिलाओं के सदृश लोग कोडिंग करना बंद कर दें तो इंटरनेट बिखर नहीं जाएगा, लेकिन इसके अन्य नुकसानदेह प्रभाव होंगे। कंप्यूटर साइंस में प्रतिनिधित्व का यह अभाव लोगों और समुदायों को कैसे प्रभावित करता है?

परिवार के किसी सदस्य या मित्र के साथ अपनी प्रतिक्रियाएँ साझा करें। अन्य लोगों को महिलाओं और महिलाओं के सदृश लोगों के कंप्यूटर साइंस के क्षेत्रों में योगदानों के बारे में अधिक पढ़ने के लिए प्रोत्साहित करें।

चरण 1: बग्स का स्वागत करें (3-5 मिनट)



अपनी आँखें बंद करें और ऐसे समय के बारे में सोचें जब आपका सामना ऐसी समस्या से हुआ था जिसे हल करना आप नहीं जानते थे, चाहे वह नए वायरल टिकटोंक डांस को नेल करना हो, अपनी पसंदीदा कुकी को सही तरीके से बनाना हो, या ऐनिमल क्रॉसिंग में सही निवास का निर्माण करना हो। उस प्रक्रिया के बारे में सोचें जिससे आपने उसे हल किया था। उन भावनाओं और मानसिकता के बारे में सोचें जिन्हें आपने इस काम के दौरान महसूस किया था। अब - आँखों को बंद रखते हुए - याद करें कि समाधान को खोजना कितना अद्भुत था।

अपने आँखें खोलें। अब भी आस-पास उछलते वार्म फज़ियों के साथ, हम आपको बताना चाहते हैं कि **आपके कोड में गलतियाँ होंगी।**



इससे कोई फर्क नहीं पड़ता यदि आप बस शुरू कर रहे हैं या आप कई वर्षों से तीन भाषाओं में प्रोग्रामिंग कर रहे हैं। आपके प्रोग्राम टूटेंगे। टूटने से हमारा मतलब है कि वे सामान्य रूप से काम नहीं करेंगे, यह नहीं कि आपके कंप्यूटर का विस्फोट हो जाएगा। यह निराशापूर्ण लग सकता है, पर ऐसा नहीं है। इन गलतियों को हल करना सीखने का सबसे अच्छा तरीका है!

अपने बग्स को गले लगाएं (1 मिनट)

हम इन गलतियों को **बग्स** कहते हैं। हम इन गलतियों को खोजने और फिक्स करने की प्रक्रिया को **डीबगिंग** कहते हैं।

लेकिन बग्स आपके कोड में केवल गलती ही नहीं होते हैं। वे आपके सॉफ्टवेयर प्रोग्राम में एक तकनीकी समस्या और आप जो करना चाहते थे और जो वास्तव में हुआ इसके बीच आपकी समझ में अंतर होते हैं। यही खास वजह है कि बग्स इतने कुंठाजनक, लेकिन उपयोगी भी होते हैं। बग खोजने और फिक्स करने का मतलब है कि आपके पास अपने प्रोग्राम को और अपनी समझ को अपडेट करने का एक मौका है कि किसी प्रोग्राम को कैसे लिखते हैं।

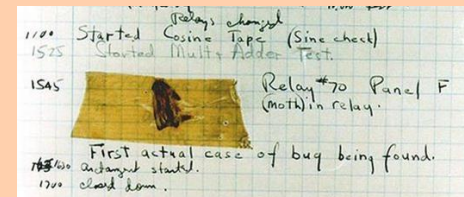
हमने पहले से ही अपनी इस समझ को अपडेट कर लिया है कि आपकी विशेषज्ञता का स्तर चाहे कोई भी हो बग फिर भी पैदा होते हैं। इसलिए, चूंकि पुराने और नए बग्स से हमारी मुलाकात होनी जारी रहेगी, उन्हें हल करने की प्रक्रिया विकसित करना जरूरी है।

चरण 2: अपने बग्स को पहचानें (7-10 मिनट)

डीबगिंग का पहला चरण यह जानना है कि आप बग का शिकार हो गए हैं। पर जब आप बग के शिकार हो जाते हैं तो वास्तव में क्या होता है? वे दिखने में कैसे हो सकते हैं? क्या मेरा प्रोजेक्ट हमेशा के लिए नष्ट हो जाएगा?! (बहुत-बहुत कम संभावना है।) चलिए इन प्रश्नों से शुरू करते हैं।

बग्स कहाँ रहते हैं? (1 मिनट)

बग्स आपके प्रोग्राम के किसी भी बिंदु पर हो सकते हैं। आप उन्हें अपनी चर राशियों, फंक्शनों, कंडीशनल्स, इत्यादि में छिपा हुआ पा सकते हैं।



स्रोत: [नेशनल जियोग्राफिक](#)

उन्हें बग इसलिए कहते हैं क्योंकि एक जमाने में जब कंप्यूटरों का निर्माण वैक्यूम नलियों से किया जाता था, तब असली कीड़े नलियों में घुस जाते थे जिसके कारण वे ठीक से काम नहीं कर पाते थे। इस गतिविधि में हम सॉफ्टवेयर बग्स के बारे में बात करेंगे, लेकिन बग्स आपके हार्डवेयर में हो सकते हैं!

बग्स कहाँ रहते हैं? (1 मिनट)

बग्स आपके प्रोग्राम के किसी भी बिंदु पर हो सकते हैं। आप उन्हें अपनी [चर राशियों](#), [फंक्शनों](#), [कंडीशनल्स](#), इत्यादि में छिपा हुआ पा सकते हैं।

```
// चर राशि घोषित नहीं की गई है: var myNumber = 1;
myNumber = 1;

// हमारे कंडीशनल में कोई बंद होने वाला कर्ली ब्रैकेट नहीं है
if(myNumber > 10){
  addOne();
}

// फंक्शन के भीतर चर राशि के नाम की वर्तनी गलत है।
function addOne(){
  myNumbr += 1;
}
```

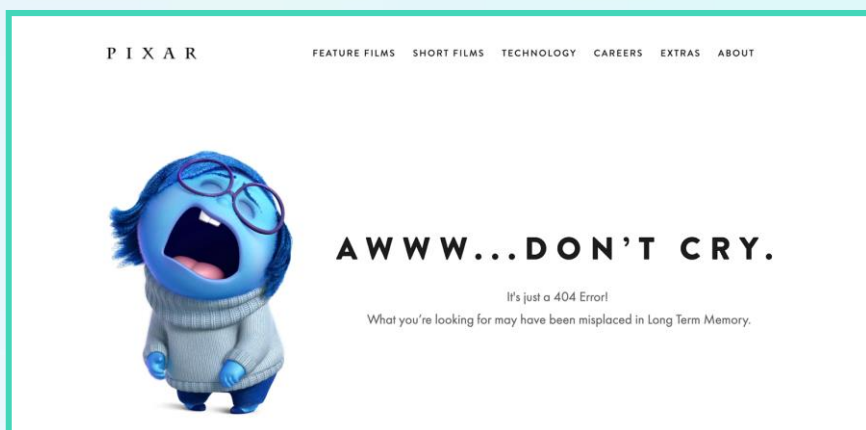
जब मेरे पास कोई बग होता है तो क्या होता है? (1 मिनट)

सभी बग्स आपके प्रोग्राम की कार्यक्षमता को प्रभावित करते हैं। कुछ बग छोटे होते हैं और कुछ बग बड़े होते हैं। बग्स बहुत अधिक मज़ेदार हो सकते हैं यदि उनके कारण चित्र फैलने लगें, स्क्रीन पर से बटन निकल कर गिरने लगें, और टेक्स्ट उड़ने लगे, लेकिन 99.9% बार ऐसा होता है।

इसकी बजाय, बग के कारण...

- किसी वेबसाइट का बटन काम करना बंद कर सकता है।
- किसी चित्र का प्रदर्शन नहीं हो पाता है।
- किसी व्यक्ति को वेबसाइट के गलत पेज पर भेज सकता है।
- किसी निश्चित संख्या पर पहुंचने के बाद खिलाड़ी का स्कोर ट्रैक होना बंद हो सकता है।
- वेबसाइट एक या कई ब्राउज़रों में गलत रूप में दिखाई दे सकती है।

और यही नहीं, किसी बड़े बग के कारण पूरी वेबसाइटें, गेम, इत्यादि काम करना बंद कर सकते हैं क्योंकि प्रोग्रामों को सामान्य तौर से कंपाइल और/या चलाया नहीं जा सकता है।



यदि आपने कभी 404 पेज देखा है, तो आपने एक बग के हरकत में देखा है।

स्रोत: [Pixar](#)

क्या बग्स कई प्रकार के होते हैं? (5-7 मिनट)

हाँ! बग्स को श्रेणीबद्ध करना उपयोगी होता है क्योंकि गलती की पहचान करना आधी लड़ाई है। इनमें **सिंटेक्स बग** और लॉजिक बग्स शामिल हैं। इस गतिविधि में हम सिंटेक्स बग्स पर ध्यान देंगे और लॉजिक बग्स को अगली गतिविधि में कवर करेंगे।

सिंटेक्स बग्स

आपने व्याकरण के संदर्भ में **सिंटेक्स** शब्द सुना होगा। इसका प्रोग्रामिंग से क्या लेना-देना है? बहुत-कुछ! प्रोग्रामिंग भाषा का सिंटेक्स किसी भी अन्य भाषा के सिंटेक्स के समान होता है। किसी भी प्रोग्राम को चलाने के लिए अक्षरों (जैसे, b और B), संख्याओं (जैसे 11 या 3.14) और प्रतीकों (जैसे, { } या #) को कुछ नियमों का पालन करना होता है। अलग-अलग भाषाओं के अलग-अलग सिंटेक्स होते हैं।

उदाहरण के लिए, जावास्क्रिप्ट एक प्रोग्रामिंग भाषा है जिसका उपयोग मुख्य तौर पर वेबसाइट्स में परस्पर गतिविधि जोड़ने के लिए किया जाता है और इस गतिविधि में हम इसी भाषा का उपयोग करेंगे। चलिए जावास्क्रिप्ट के दो सबसे साफ नज़र आने वाले नियमों को देखते हैं:

- जावास्क्रिप्ट कोड के टुकड़ों, जैसे फंक्शनों या कंडीशनल वक्तव्यों को अलग करने के लिए कर्ली ब्रैकेटों { } और कोष्ठकों () का उपयोग करती है।** पायथन जैसी अन्य भाषाएं कोड के टुकड़ों को अलग करने के लिए कर्ली ब्रैकेटों की बजाय इंडेंटेशन का उपयोग करती हैं।
- जावास्क्रिप्ट आपके कोड में वक्तव्य को समाप्त करने के लिए ; सेमीकोलनों का उपयोग करती है।** वक्तव्य एक निर्देश है जो आप अपने प्रोग्राम को दे रहे हैं जैसे `console.log(score);`। ठीक जैसे आपको वाक्य के अंत में पूर्णविराम लगाना चाहिए, वैसे ही आपको अपने वक्तव्यों को हमेशा सेमीकोलन से समाप्त करना चाहिए।

अधिकांश आम सिंटेक्स बग्स से बचने में आपकी मदद करने के लिए यहाँ कुछ दिशानिर्देश दिए गए हैं:

बहुत सारी सिंटेक्स गलतियाँ वर्तनी की गलतियाँ होती हैं।

उदाहरण के लिए, डेटा स्टोर करने के लिए हम चर राशियों का उपयोग करते हैं इसलिए जरूरत पड़ने पर हम उन्हें बदल सकते हैं या अन्य मानों के साथ उनकी तुलना कर सकते हैं। क्योंकि हम उन्हें बार-बार उपयोग करते हैं, इस बात की अधिक संभावना है कि हम किसी बिंदु पर उनकी वर्तनी गलत लिख दें।

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

चर राशियाँ, फंक्शन के नाम, और कीवर्ड केस सेंसिटिव होते हैं।

आप लोअरकेस और अपरकेस अक्षरों को आपस में नहीं बदल सकते।

`var bee` और `var Bee` एक समान नहीं हैं

`function` और `Function` एक समान नहीं हैं।

यदि आप इसे खोलते हैं, तो आपको इसे बंद करना चाहिए।

जब आप कोड के कई टुकड़ों को समूहबद्ध करते हैं तो कोई कर्ली ब्रैकेट या कोष्ठक के छूट जाने या अतिरिक्त के जुड़ जाने की गलती हो सकती है। याद रखें कि सभी कर्ली ब्रैकेटों और कोष्ठकों का एक साथी होना चाहिए।

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

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





जावास्क्रिप्ट के बारे में अधिक जानकारी के लिए, तानिया रैशिया की [यह गाइड](#) देखें।

चरण 3: बगी साइट की जाँच करें (3-5 मिनट)





किसी भी चीज को डीबग करने से पहले, हमें समस्या को समझने की जरूरत है। इस प्रक्रिया को शुरू करने का सबसे अच्छा तरीका प्रोग्राम की जाँच करना है। सबसे पहले आप फंक्शनिंग साइट पर प्रश्नोत्तरी लेंगे। फिर, आप यह जाँचने के लिए बगी संस्करण को देखेंगे कि उसका व्यवहार काम कर रहे संस्करण की तुलना में कैसा है।

What kind of bug are you?





What is your favorite dessert?


Pick a location for your next vacation.

Which Color of the Year do you prefer?





   

You are a...







What kind of bug are you?





What is your favorite dessert?


Pick a location for your next vacation.

Which Color of the Year do you prefer?

You are a grasshopper!



फिक्स्ड बगी पर्सनैलिटी प्रश्नोत्तरी लें (3-5 मिनट)

[प्रश्नोत्तरी के फिक्स्ड संस्करण](#) के लिंक का अनुसरण करें। अब यह देखने के लिए प्रश्नोत्तरी पूरी करें कि आप कौन से बग हैं! जब आप काम पूरा कर लें, तो निम्नलिखित प्रश्नों के बारे में सोचें। यदि उपयोगी लगता है तो अपने विचारों को लिख लें।

- ☐ अपना परिणाम प्राप्त करने के लिए आपने कौन से कदम उठाए?
- ☐ आपके विचार से प्रोग्राम ने कैसे पता लगाया कि आप कौन से बग हैं?



पेज 2 पर संदर्भ मार्गदर्शिका में अपने विचारों की जाँच करें।

टूटी हुई प्रश्नोत्तरी लें (1-2 मिनट)

चलिए देखते हैं कि टूटी हुई प्रश्नोत्तरी कैसी लगती है। [बगी संस्करण के लिए इस लिंक](#) का अनुसरण करें और उसकी जाँच करें। यह पता लगाने के लिए कि प्रोग्राम कैसे व्यवहार करता है, प्रश्नोत्तरी को कुछ बार दोहराएं। अलग-अलग तरीकों से प्रश्नों का उत्तर देकर देखने की कोशिश करें कि क्या कुछ बदलता है या नहीं। यदि सबकुछ ठीक से टूटा है, तो प्रश्नोत्तरी कभी भी परिणाम नहीं देगी।

चरण 4: Repl.it के साथ शुरू करें (10 - 15 मिनट)

इस गतिविधि के लिए हम Repl.it वेब एडिटर का उपयोग करेंगे। [Repl.it](#) एक मुफ्त, सहयोगात्मक, ब्राउज़र पर आधारित एडिटर है जो अनेक प्रोग्रामिंग भाषाओं का समर्थन करता है। यह शक्तिशाली टूल आपको कोड करने और उसी समय अपने कई मित्रों के साथ बात करने की अनुमति तक दे सकता है!

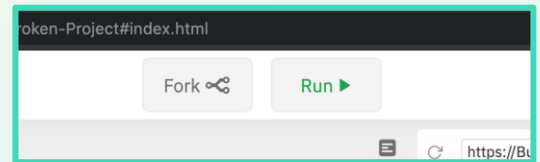
अपना खाता बनाएं (5-7 मिनट)

- ❑ **Repl.it में साइन अप या लॉगिन करें।** अपने काम को सहेजने के लिए आपको एक खाता बनाने की जरूरत पड़ेगी। खाता बनाने के लिए साइन अप फॉर्म पर दिए निर्देशों का पालन करें। यदि आप 13 वर्ष से कम उम्र की हैं, तो आपको साइन अप करने के लिए अपने माता/पिता के ईमेल पते की आवश्यकता होगी।
- ❑ **खाता बनाने के लिए निर्देशों का पालन करें।** अधिक तेज लॉगिन के लिए आप अपने Google, GitHub, या Facebook खाते के साथ साइन अप करने का चुनाव कर सकते हैं।

टूटे हुए बगी व्यक्ति की प्रश्नोत्तरी को फोर्क करें (2-3 मिनट)

- ❑ **टूटे हुए बगी व्यक्ति की प्रश्नोत्तरी को खोलें।**
- ❑ **रन बटन के बगल में स्थित फोर्क बटन पर क्लिक करके टूटे हुए बगी व्यक्ति की प्रश्नोत्तरी की एक कॉपी बनाएं।** फोर्क बनाने से संपूर्ण प्रोजेक्ट का डुप्लीकेट बन जाता है और वह आपके Repl खाते में एक नए प्रोजेक्ट की तरह जुड़ जाता है।

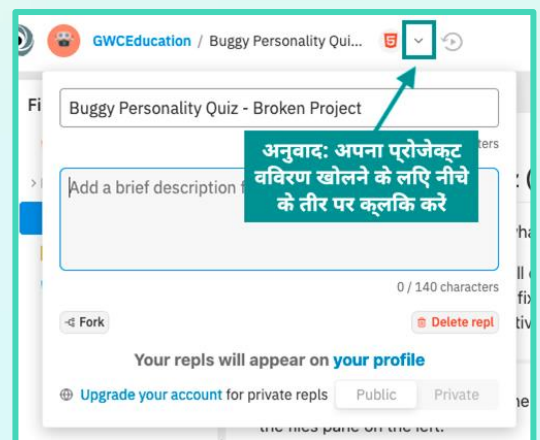
आप अन्य लोगों के स्रोत कोड को फोर्क कर सकते हैं या अपने खुद के प्रोजेक्ट फोर्क कर सकते हैं यदि, मान लें कि, आप बग फिक्स बना रहे हैं, लेकिन अपने कोड का पुराना संस्करण रखना चाहते हैं जिस पर आप डीबगिंग करते समय अधिक गलतियाँ होने की स्थिति में वापस जा सकते हैं।



कोई प्रोजेक्ट विवरण जोड़ें (1-2 मिनट)

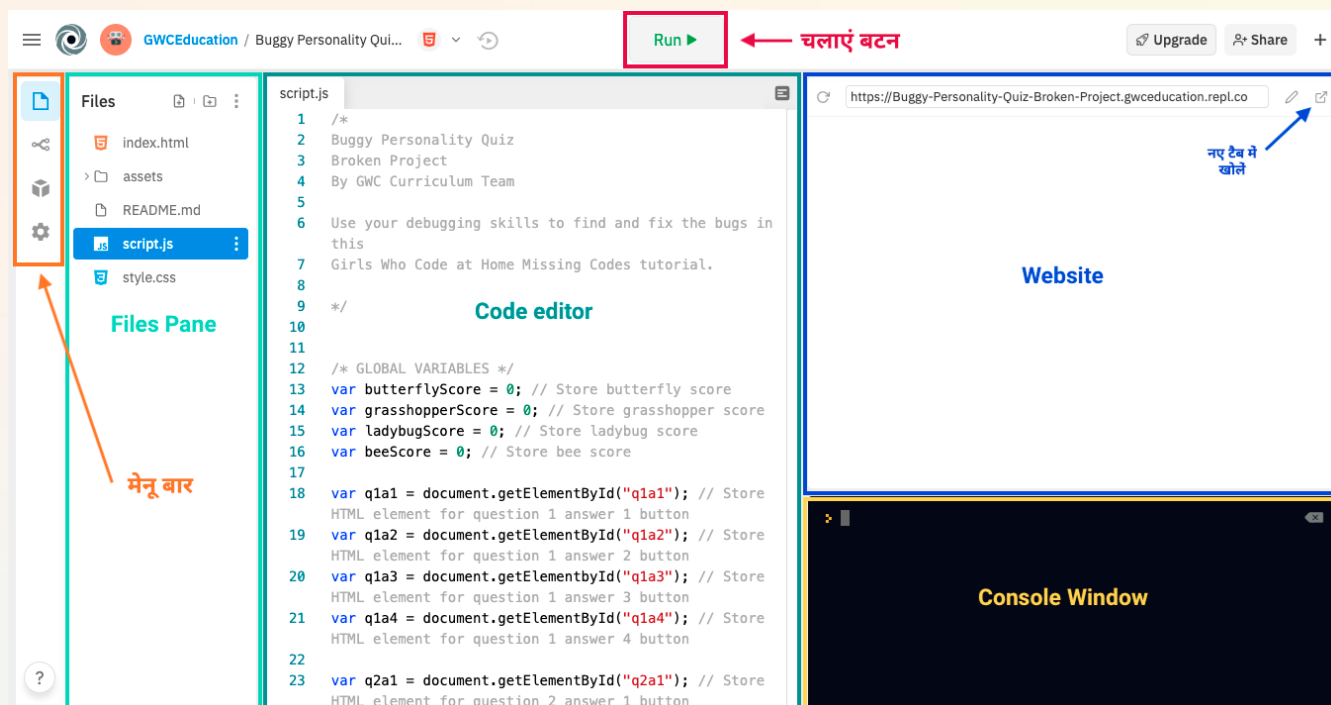
अब जब आपके पास बगी प्रश्नोत्तरी की अपनी कॉपी है, चलिए अपने Repl में संक्षिप्त प्रोजेक्ट विवरण जोड़ते हैं।

- ❑ **प्रोजेक्ट का विवरण खोलें।** अपने स्क्रीन के ऊपरी बायें तरफ अपने प्रोजेक्ट का नाम खोजें। नाम के दायीं तरफ स्थित छोटे डाउन तीर पर क्लिक करें। इससे एक नई विंडो खुलनी चाहिए जहाँ आपके प्रोजेक्ट का नाम और संक्षिप्त विवरण के लिए क्षेत्र होगा।
- ❑ **एक संक्षिप्त विवरण जोड़ें।** आप अपने विवरण में निम्नलिखित जानकारी शामिल करना चाहेंगे:
 - ❑ **ओवरव्यू:** इसे कैसे काम करना चाहिए?
 - ❑ **निर्देश:** क्या आपके प्रोजेक्ट को चलाने के लिए किन्हीं विशिष्ट निर्देशों की जरूरत है?
 - ❑ **विशेषताएं:** क्या आपको दूसरों से या अतिरिक्त संसाधनों से मदद मिली? सुनिश्चित करें कि आप इन लोगों और संसाधनों को आवाज़ दें!



एडिटर व्यू का अन्वेषण करें (3-5 मिनट)

चलिए Repl.it के लिए एडिटर व्यू को देखते हैं। अब जबकि हमने एक नया प्रोजेक्ट बना लिया है, हमें यह समझने की जरूरत है कि कहाँ कोड करना है, फाइलों और संपत्तियों के बीच कैसे नेविगेट करना है और आपके कोड को कैसे सहेजना और चलाना है!



- **फाइल्स पेन:** यह विंडो आपकी सभी प्रोजेक्ट फाइलों को प्रदर्शित करती है। इस गतिविधि के लिए, हम केवल एक फाइल, script.js में काम करेंगे, लेकिन आप index.html, style.css, और README.md फाइलें भी देखेंगे। इनमें से प्रत्येक फाइल क्या करती है इसके बारे में अधिक जानकारी के लिए README फाइल पर क्लिक करें।
- **मेनू बार:** यह बार आपको फाइल पेन में व्यू को बदलने की अनुमति देगा। कुछ विकल्पों में संस्करण नियंत्रण बदलना, पैकेज जोड़ना, और सेटिंग्स अपडेट करना शामिल है। सेटिंग्स में आप लेआउट, थीम, फॉन्ट का आकार, और अन्य टेक्स्ट सेटिंग्स बदल सकते हैं।
- **कोड एडिटर:** यह वह जगह है जहाँ आप अपना कोड लिखते हैं!
- **रन बटन:** अपने कोड में परिवर्तन करने के बाद, एडिटर के शीर्ष पर चलाएं बटन पर क्लिक करें। आपको दायीं ओर स्थित आउटपुट/कंसोल विंडो में अपना परिणाम दिखना चाहिए।
- **वेबसाइट:** यह विंडो आपकी वेबसाइट को प्रदर्शित करती है। यदि आप इसे एक पूरे वेबपेज के रूप में देखना चाहते हैं, तो शीर्ष दायें कोने में नई टैब बटन पर क्लिक करें।
- **कंसोल विंडो:** यह आपके कोड में किसी भी आउटपुट को प्रदर्शित करती है। सभी आउटपुट दिखाई देंगे, इसलिए यदि आप रन बटन पर कई बार क्लिक करते हैं, तो यहाँ प्रत्येक परिणाम दिखाया जाएगा।

आपने देखा होगा कि Repl.it में कोई **सहेजें** बटन नहीं है। जब तक आपके पास इंटरनेट कनेक्शन होगा, आपके कोड के सभी परिवर्तन स्वचालित रूप से सहेजे जाएंगे। जैसे ही आप रन बटन पर क्लिक करते हैं आपका Repl सहेज लिया जाता है, इसलिए सुनिश्चित करें कि अपने एडिटर को बंद करने से पहले आप हमेशा अपने कोड को रन कराएं!

चरण 5: पहले बग को फिक्स करें (5-7 मिनट)

चरण 3 में, हमने देखा कि हमारी टूटी हुई प्रश्नोत्तरी का व्यवहार कैसा था और वर्णन किया कि वह कैसे किसी व्यक्ति के बग स्कोर को ट्रैक कर सकती है। अब समय है हमारी **script.js** में वास्तविक कोड की समीक्षा करने का ताकि हम देख सकें कि समस्याएं कहाँ हैं।

ऊपरी नेविगेशन बार के केंद्र में *चलाएं* बटन को दबाएं। स्क्रीन के नीचे दायीं ओर स्थित कंसोल को देखें।

SyntaxError: /script.js:111:1 पर इनपुट की अनपेक्षित समाप्ति

हमारे पास अपना पहला त्रुटि संदेश है! त्रुटि संदेश सर्वोत्तम होते हैं। वे हमें एक सुराग देते हैं कि समस्या क्या है और उसे खोजने में हमारी मदद करते हैं। एक पल के लिए रुकें और इस संदेश को डीकोड करने का प्रयास करें:

- **SyntaxError** का मतलब है कि आपके सिंटेक्स (यानी, वर्तनी, मात्रा, फॉर्मेटिंग, आदि) की किसी समस्या के कारण बग पैदा हुआ है।
- **इनपुट की अनपेक्षित समाप्ति** का मतलब है कि कंप्यूटर को पता नहीं था कि कहाँ समाप्त करना है। यदि आप अपने कोड में कहीं पर कर्ली ब्रैकेट को बंद करना भूल जाते हैं तो आम तौर पर ऐसा त्रुटि संदेश प्राप्त होता है।
- **at /script.js** का मतलब है कि आपकी त्रुटि **script.js** फाइल में है।
- **111:1** बताता है कि आपकी त्रुटि पंक्ति 111 के पहले स्पेस में है।

अब यह काफी सारी जानकारी है जिसके साथ हम काम कर सकते हैं। चलिए पंक्ति 109 पर जाकर शुरू करते हैं। वह अजीब सी है। लगता है वहाँ पर एक कर्ली ब्रैकेट पहले से है। कर्ली ब्रैकेट पर क्लिक करके देखें कि क्या होता है।

```
101 // प्रश्नोत्तरी दोबारा शुरू करें
102 function restartQuiz() {
103     result.innerHTML = "You are a...";
104     questionCount = 0;
105     beeScore = 0;
106     butterflyScore = 0;
107     grasshopperScore = 0;
108     ladybugScore = 0;
109 }
```

← बंद होने वाले कर्ली ब्रैकेट पर क्लिक करें

यहाँ एक जोरदार सुझाव है: अधिकांश IDEs में जब आप अपने कर्सर को एक कर्ली ब्रैकेट या कोष्ठक के बगल में जाते हैं, तो वह मौजूदा कोड में पार्टनर कर्ली ब्रैकेट या कोष्ठक को हाइलाइट करता है। इस तकनीक का उपयोग करके, हम जानते हैं कि:

1. 109 पर मौजूद कर्ली ब्रैकेट का पार्टनर पहले से ही पंक्ति 102 पर है
2. restartQuiz फंक्शन में कोई भी अन्य खुले कर्ली ब्रैकेट नहीं हैं।

इसका मतलब है कि हमें जासूसी करनी होगी!

नीचे दिए चरणों का अनुसरण करें, फिर संदर्भ मार्गदर्शिका में अपने कोड की जाँच करें:

- ❑ **आस-पास देखें।** यदि त्रुटि सीधे वहाँ नहीं दिखती है जहाँ संदेश ने कहा था, तो आस-पास देखने की कोशिश करें। `restartQuiz` फंक्शन के बाद कोई कोड नहीं है, इसलिए चलिए पंक्ति 87 पर शुरू होने से ठीक पहले `updateResult` फंक्शन की जाँच करते हैं:

```

110 // प्रश्नोत्तरी का परिणाम अपडेट करें
111 फंक्शन updateResult() {
112   यदि (beeScore >= 2){
113     result.innerHTML = "आप एक बी हैं!";
114   } अन्यथा यदि (butterflyScore >= 2){
115     result.innerHTML = "आप एक बटरफ्लाई हैं!";
116   } अन्यथा यदि (grasshopperScore >= 2){
117     result.innerHTML = "आप एक ग्रासहॉपर हैं!";
118   } अन्यथा यदि (ladybugScore >= 2){
119     result.innerHTML = "आप एक लेडीबग हैं!";
120   } अन्यथा {
121     result.innerHTML = "हूँ...पक्का नहीं कह सकते। बाद में दोबारा प्रयास करें!";
122   }
123 }
```

- ❑ **टिप्पणियाँ तैयार करें।** आपको सबसे पहले क्या दिखाई देता है? ढेर सारे कर्ली ब्रैकेट! यह प्राइम बग लोकेशन है। `updateResult` फंक्शन में एक लंबा कंडीशनल वक्तव्य है जो व्यक्ति के स्कोर का मूल्यांकन करता है और बताता है कि वह कौन सा बग है।
- ❑ **सोचें कि आपको क्या चाहिए।** इस बिंदु पर, पीछे हटकर यह सोचना उपयोगी होता है कि क्या आपको पता है कि आपको क्या चाहिए। फंक्शन की शुरुआत में एक खुला कर्ली ब्रैकेट और अंत में एक बंद कर्ली ब्रैकेट। यदि अंदर कुछ नहीं था, तो वह ऐसा दिखेगा:

```

function updateResult(){
}
```

- ❑ **अपने टूल्स की पहचान करें।** आपके पास कौन से टूल हैं या आपको इसका हल करने के लिए पता है? इससे पहले हमने सीखा कि कर्ली ब्रैकेट के पार्टनर को प्रदर्शित करने के लिए हम अपने कर्सर का उपयोग कर सकते हैं। चलिए इसे आजमाते हैं।
- ❑ **इसकी जाँच करें।** `updateResult` फंक्शन में हर कर्ली ब्रैकेट की जाँच करके देखें कि कौन सा वाला पहले खुले कर्ली ब्रैकेट का पार्टनर है।
- ❑ **अपने परिवर्तन करें।** आपको केवल एक ही परिवर्तन करना है।
- ❑ **प्रोग्राम चलाएं।** अपने प्रोग्राम को चलाकर जाँचें कि क्या वह बिना त्रुटि के चलता है।



संदर्भ मार्गदर्शिका के पेज 4-5 पर अपने कोड की जाँच करें।

चरण 6: दूसरे बग को हल करें (3-5 मिनट)

हमने अपने पहले बग को सफलतापूर्वक हल किया - चलिए अब उस दूसरे त्रुटि संदेश को देखते हैं:

TypeError: document.getElementById, /script.js:20:21 पर एक फंक्शन नहीं है

हमारे पास अपना पहला त्रुटि संदेश है! त्रुटि संदेश सर्वोत्तम होते हैं। वे हमें एक सुराग देते हैं कि समस्या क्या है और उसे खोजने में हमारी मदद करते हैं। एक पल के लिए रुकें और इस संदेश को डीकोड करने का प्रयास करें:

- **TypeError** का मतलब यह है कि प्रोग्राम संचालन नहीं कर सकता है क्योंकि एक मान उस प्रकार का मान नहीं है जिसकी अपेक्षा कंप्यूटर को थी। इस मामले में, मान एक फंक्शन का नाम है।
- **document.getElementById** उस फंक्शन का नाम है जिसके कारण बग पैदा हो रहा है।
- **एक फंक्शन नहीं है** का मतलब है कि कंप्यूटर इसे एक फंक्शन के रूप में नहीं पहचानता है।
- **at /script.js** का मतलब है कि आपकी त्रुटि **script.js** फाइल में है।
- **20:21** बताता है कि आपकी त्रुटि पंक्ति 20 के 21वें स्पेस में है।

अंतिम चरण में हमने जो कुछ सीखा उसके आधार पर, हम जानते हैं कि त्रुटि **script.js** फाइल में पंक्ति 20 पर है।

```
var q1a3 = document.getElementById("q1a3"); // रश्म 1 उत्तर 3 बटन के लिए HTML तत्व को स्टोर करें
```

त्रुटि संदेश कहता है कि **document.getElementById** एक फंक्शन नहीं है। लेकिन हम जानते हैं कि यह एक फंक्शन है क्योंकि हम इसका उपयोग कई अन्य जगहों में करते हैं। तो कहाँ गड़बड़ है? पता लगाने के लिए इन चरणों का अनुसरण करें:

- ❑ **पैटर्नों की तलाश करें।** क्या आपको उन अन्य स्थानों में जहाँ इसका उपयोग किया जाता है कोई अंतर दिखाई देते हैं?

```
18 var q1a1 = document.getElementById("q1a1"); // प्रश्न 1 उत्तर 1 बटन के लिए HTML तत्व को
    स्टोर करें
19 var q1a2 = document.getElementById("q1a2"); // प्रश्न 1 उत्तर 2 बटन के लिए HTML तत्व को
    स्टोर करें
20 var q1a3 = document.getElementById("q1a3"); // प्रश्न 1 उत्तर 3 बटन के लिए HTML तत्व को
    स्टोर करें
21 var q1a4 = document.getElementById("q1a4"); // प्रश्न 1 उत्तर 4 बटन के लिए HTML तत्व को
    स्टोर करें
```

- ❑ **कोई भी परिवर्तन करने की कोशिश करें।** यदि आपको लगता है कि आप जानते हैं कि समस्या कहाँ है, तो उसे फिक्स करें।
- ❑ **प्रोग्राम चलाएं।** अपने प्रोग्राम को चलाकर जाँचें कि क्या वह बिना त्रुटि के चलता है।



संदर्भ मार्गदर्शिका के पेज 6 पर अपने कोड की जाँच करें।



बधाई!

आपने सभी सिंटेक्स त्रुटियों को सफलतापूर्वक डीबग कर लिया है। जब आप प्रोग्राम को चलाएंगे, तो आपको कोई त्रुटि संदेश नहीं दिखना चाहिए। तथापि, यदि आपने प्रश्नोत्तरी पूरी करने की कोशिश की थी, तो आपने देखा होगा कि वह अब भी काम नहीं कर रही है। इसका कारण यह है कि एक बग बच गया है: लॉजिक बग। ये बग होते हैं क्योंकि प्रोग्राम के प्रवाह या आपकी कोड की पंक्तियों के निष्पादन के क्रम में एक समस्या है। अगली गतिविधि में, हम डीबगिंग रणनीतियों के एक सेट का अन्वेषण करेंगे जिनका उपयोग आप इस बग और भविष्य में आपके सामने आने वाले किसी भी बग को फिक्स करने के लिए कर सकते हैं?

चरण 9: अपने Girls Who Code at Home परियोजना को साझा करें! (5 मिनट)

केवल एक्सेस देखें (1-2 मिनट)

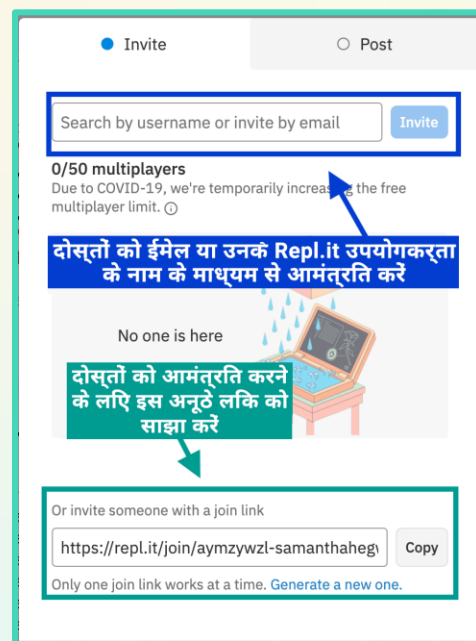
Repl.it पर आपके काम को साझा करना आसान है! बस अपने Repl प्रोजेक्ट के URL पते को शीर्ष पर वेब एड्रेस बार में कॉपी और पेस्ट करें। इससे अन्य लोग आपके प्रोजेक्ट को चलाने, आपका कोड देखने, और आपके प्रोजेक्ट को फोर्क करने तथा अपने खुद के प्रोजेक्ट के साथ रीमिक्स करने में सक्षम होंगे। हम आपके डीबगिंग काम को देखना पसंद करेंगे और हम जानते हैं कि दूसरे भी ऐसा करेंगे। अपने फिक्स्ड कोड को हमारे साथ साझा करें! **इस लिंक को अपने सोशल मीडिया खातों पर साझा करें तथा @girlswhocode #codefromhome को टैग करना मत भूलें और हो सकता है कि हम आपको अपने खाते में शामिल कर लें!**

सहयोगी जोड़ना (2 मिनट)

यदि आप किसी प्रोजेक्ट पर मित्रों के समूह के साथ काम करना चाहते हैं, तो आप विंडो के ऊपरी-दाएं भाग में स्थित **साझा करें** बटन का उपयोग करके उन्हें सहयोग करने के लिए आसानी से आमंत्रित कर सकते हैं। इससे एक नई विंडो खुलेगी जिसमें आपके प्रोजेक्ट पर सहयोग करने के लिए अन्य लोगों को आमंत्रित करने के दो विकल्प होंगे।

- ❑ **ईमेल या Repl.it यूजरनेम के द्वारा आमंत्रित करें।** यह विकल्प आपको विशिष्ट लोगों के साथ अपना प्रोजेक्ट साझा करने के लिए उनका ईमेल पता या यदि उनके पास पहले से ही Repl.it के साथ खाता है तो Repl.it यूजरनेम टाइप करके आमंत्रित करने देता है। हम यह सुनिश्चित करने के लिए कि आप अपने प्रोजेक्ट को सही लोगों के साथ साझा कर रहे हैं, इस विकल्प की अनुशंसा करते हैं!
- ❑ **आमंत्रण लिंक साझा करें।** विंडो के तल पर एक अद्वितीय आमंत्रण लिंक है। आप इस लिंक को मित्रों को कॉपी और पेस्ट कर सकते हैं ताकि वे आपके प्रोजेक्ट को एक्सेस कर सकें।

सहयोगियों के बारे में टिप्पणी: याद रखें कि सहयोगी जोड़ने से अन्य लोगों को आपके प्रोजेक्ट को संपादित करने की सुविधा मिलती है। इससे वे आपके कोड, नाम, और विवरण को बदलने में सक्षम होते हैं। **अपने आमंत्रण लिंक को सोशल मीडिया पर साझा मत करें!** इन संपादन अधिकारों को आप किनके साथ साझा करेंगे इस बारे में चयनशील रहें।



मिसिंग कोड को डीबग करें भाग 2 में अधिक जानकारी की प्रतीक्षा करें!

