



# Girls Who Code At Home

गुम कोड को डीबग करें भाग 2  
संदर्भ मार्गदर्शिका

## गुम कोड को डीबग करें भाग 2 संदर्भ मार्गदर्शिका



इस दस्तावेज़ में आपको गतिविधि के कुछ प्रश्नों के सभी उत्तर मिलेंगे। गतिविधि में आगे बढ़ते रहें, और जब आपको यह आइकॉन दिखे तो रुकें और यहां अपने विचारों की जांच करें।

### चरण 1: लॉजिक बग से मिलें

#### लॉजिक त्रुटियों से समस्याएँ क्यों होती हैं?

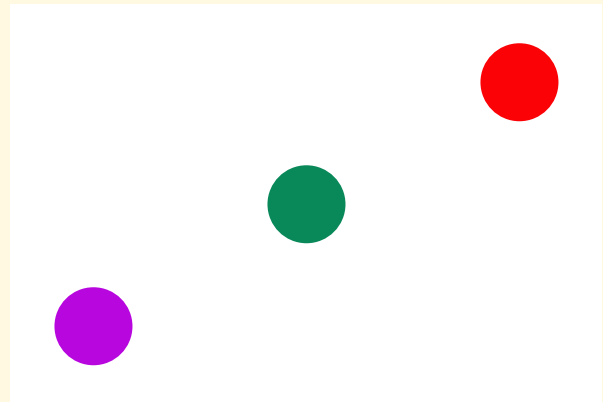
रंग भूरा होगा क्योंकि प्रोग्राम पेंटर को हर उपयोग के बाद ब्रश साफ करने के लिए नहीं कहता है। यह केवल उन्हें पेंटिंग खत्म करने के बाद ब्रश को साफ करने के लिए कहता है।

अंतिम सर्कल के वाइब्रेंट पर्पल होने के लिए, आप प्रत्येक रंग के बाद **ब्रश को धोएं** प्रत्येक रंग के बाद निर्देश:

#### कोड

ब्रश को लाल पेंट में डुबाएं  
सर्कल को पेंट करें  
ब्रश को धोएं  
अपने ब्रश को हरे पेंट में डुबाएं  
सर्कल को पेंट करें  
ब्रश को धोएं  
अपने ब्रश को पर्पल पेंट में डुबाएं  
सर्कल को पेंट करें  
ब्रश को धोएं  
कैनवस को सूखने दें

#### परिणाम



### चरण 4: कोड की समीक्षा करें

चरण की शुरुआत में [ब्रोकेन कोड](#) का विस्तृत स्पष्टीकरण नीचे दिया गया है।

#### चर राशि को घोषित करें

सबसे पहले, हमने प्रत्येक बग के लिए स्कोर को संग्रहीत करने के लिए चर राशि घोषित किया (जैसे, `butterflyScore`)। फिर, हमने अपने प्रश्नों में प्रत्येक उत्तर बटन के लिए HTML तत्व को स्टोर करने के लिए चर राशियों की एक सिरीज की घोषणा की (जैसे, `q2a3`)। किसी व्यक्ति ने कितने प्रश्नों पर क्लिक किया है, यह जानने के लिए हमने चर राशि घोषित कि है (`questionCount`) और परिणाम संग्रहीत करने और HTML तत्वों को पुनरारंभ करने के लिए।

## बटन के क्लिक सुनें

फिर, हमने अपने सभी बटनों में कार्यक्रम श्रोता जोड़े। [कार्यक्रम श्रोता](#) वेबपेज पर किसी इनपुट के लिए सुनने, जैसे, माउस क्लिक, कीप्रेस, या स्क्रोलिंग। यदि ब्राउज़र वह इनपुट “सुनता है” या उसका पता लगता है जिसकी उसे प्रतीक्षा है, तो प्रोग्राम कार्रवाई करता है। हम इस कार्रवाई को एक फंक्शन में परिभाषित करते हैं जो कार्यक्रम श्रोता के भीतर जाता है। उदाहरण के लिए, यदि कोई व्यक्ति प्रश्न 1 के लिए **Mochi** का चुनाव करता है, तो वह **q1a1** बटन पर क्लिक करता है। जब वह बटन पर क्लिक करता है, तो ब्राउज़र **butterfly** फंक्शन में कोड को निष्पादित करता है।

जावास्क्रिप्ट	विवरण
<code>q1a1.addEventListener("click", butterfly);</code>	<ul style="list-style-type: none"><li>→ <b>q1a1</b>: वह चर राशि जो mochi बटन से इनपुट को स्टोर करती है (पहले प्रश्न का पहला उत्तर)।</li><li>→ <b>.</b>: पीरियड या डॉट हमारे प्रोग्राम से उसके बाद की पद्धति में चर राशि को अटैच करने के लिए कहता है। यहाँ वह पद्धति <code>addEventListener</code> है।</li><li>→ <b>addEventListener</b>: कार्यक्रम श्रोता पद्धति (या फंक्शन) को कॉल करने के लिए कीवर्ड।</li><li>→ <b>"click"</b>: वह कार्यक्रम जिसके लिए हम सुन रहे हैं।</li><li>→ <b>butterfly</b>: यदि फंक्शन होता है, तो फंक्शन निष्पादित करेगा</li><li>→ <b>;</b>: सभी जावास्क्रिप्ट वक्तव्य अर्धविराम (सेमीकोलन) से समाप्त होते हैं।</li></ul>

## स्कोर को ट्रैक और अपडेट करें

अब हमें उन फंक्शनों की रचना करने की जरूरत है जिन्हें हमने अपने कार्यक्रम श्रोताओं के भीतर सूचीबद्ध किया था। इनमें से चार प्रत्येक बग के स्कोर पर नज़र रखते हैं: **bee**, **butterfly**, **grasshopper**, और **ladybug**। यदि इनमें से किसी फंक्शन को बुलाया जाता है (यानी जब कोई व्यक्ति उस फंक्शन वाले बटन को क्लिक करता है जिसे उससे अटैच किया गया है), हम बग के स्कोर को 1 से बढ़ाते हैं और चर राशि को 1 से बढ़ाते हैं **questionCount**

फिर, क्विज़ के पूरे होने पर हमारे पास मूल्यांकन करने के लिए एक कंडीशनल वक्तव्य होता है। यदि **questionCount** 3 के बराबर है (यहाँ हम [कंपेयरीजन ऑपरेटर](#) `==` का उपयोग करते हैं, असाइनमेंट ऑपरेटर का नहीं `=`), तो हम नीचे **updateResult** फंक्शन को निष्पादित करते हैं। यह फंक्शन प्रत्येक स्कोर को जाँचकर देखता है कि क्या वह 2 से बड़ा है या उसके बराबर। यदि उनमें से एक उस शर्त को पूरा करता है, तो यह फंक्शन आपके परिणाम के नीचे मौजूद टेक्स्ट को अपडेट करता है। यदि उनमें से कोई भी ऐसा नहीं करता है, तो हम व्यक्ति को फिर से प्रयास करने के लिए कहते हैं।

## क्विज़ को रीस्टार्ट करें

अंतिम फंक्शन निष्पादित होता है यदि **Restart** को क्लिक किया जाता है। यहाँ, रीस्टार्ट का मतलब सभी स्कोर और **questionCount** को 0 पर रीसेट करना है। अन्यथा, संख्याएं बढ़ती जाती हैं।

## चरण 5: समस्या का वर्णन करें

नीचे एक उदाहरण का वर्णन है कि हम अपने कार्यक्रम में क्या करना चाहते हैं।

- जब कोई व्यक्ति किसी उत्तर बटन को क्लिक करता है, तो प्रोग्राम कार्यक्रम श्रोता में उससे संलग्न विशिष्ट बग फंक्शन (i.e. `bee`, `butterfly`, `grasshopper`, या `ladybug`) को कॉल करता है।
- फंक्शन उस बग के वर्तमान स्कोर में 1 जोड़ता है और वर्तमान `questionCount` मूल्य में 1 जोड़ता है।
- जाँच करें कि क्या `questionCount` मूल्य, जब भी `questionCount` का बढ़ता है, तब 3 के बराबर है या नहीं।  
**ध्यान दें:** चूंकि क्विज़ में तीन प्रश्न हैं, हम जानते हैं कि व्यक्ति ने काम पूरा कर लिया है यदि यह मूल्य 3 के बराबर है। यदि हमारे पास चार प्रश्न हुए तो हम उसे बदलकर 4 कर देंगे।
- यदि `questionCount` 3 के बराबर नहीं है, तो कुछ मत करें और उस व्यक्ति को क्लिज़ जारी रखने दें।
- यदि `questionCount` 3 के बराबर है, तो प्रत्येक बग के स्कोर के मूल्य की जाँच करने के लिए `updateResult` फंक्शन को कॉल करें।
- प्रत्येक बग के स्कोर का मूल्यांकन करें और `updateResult` फंक्शन में एक परिणाम रिटर्न करें:
  - ◆ यदि `beeScore` 2 से अधिक या उसके बराबर है, तो नीचे दिए टेक्स्ट को इस प्रकार अपडेट करें, "You are a bee!"
  - ◆ अन्यथा यदि `butterflyScore` 2 से अधिक या उसके बराबर है, तो नीचे दिए टेक्स्ट को इस प्रकार अपडेट करें, "You are a butterfly!"
  - ◆ अन्यथा यदि `grasshopperScore` 2 से अधिक या उसके बराबर है, तो नीचे दिए टेक्स्ट को इस प्रकार अपडेट करें, "You are a grasshopper!"
  - ◆ अन्यथा यदि `ladybugScore` 2 से अधिक या उसके बराबर है, तो नीचे दिए टेक्स्ट को इस प्रकार अपडेट करें, "You are a ladybug!"
  - ◆ अन्यथा नीचे दिए गए टेक्स्ट को इस प्रकार अपडेट करें, "Hmm...not sure. Try again later"
- जब `Restart` बटन को क्लिक किया जाता है, तो `restartQuiz` फंक्शन को कॉल करके सभी चर राशियों को 0 पर सेट करने के लिए कहें।

## चरण 6: इसे प्रेक्षण-योग्य बनाएं

प्रत्येक स्कोरिंग फंक्शन में `console.log()` जोड़कर कोड करें:

```
58 // बी स्कोर को ट्रैक करें
59 फंक्शन bee() {
60     beeScore += 1;
61     questionCount += 1;
62     console.log("questionCount = " + questionCount + "\t" + "beeScore = " +
63     beeScore);
64 }
```

```

65 // बटरफ्लाई स्कोर को ट्रैक करें
66 फंक्शन butterfly() {
67     butterflyScore += 1;
68     questionCount += 1;
69     console.log("questionCount = " + questionCount + "\t" + "butterflyScore = "
70         + butterflyScore);
71 }
72 // ग्रासहॉपर स्कोर को ट्रैक करें
73 फंक्शन grasshopper() {
74     grasshopperScore += 1;
75     questionCount += 1;
76     console.log("questionCount = " + questionCount + "\t" + "grasshopperScore = "
77         + grasshopperScore);
78 }
79 // लेडीबग स्कोर को ट्रैक करें
80 function ladybug() {
81     ladybugScore += 1;
82     questionCount += 1;
83     console.log("questionCount = " + questionCount + "\t" + "ladybugScore = "
84         + ladybugScore);
85 }

```

हमारे सुराग के उत्तर:

- ❑ परिणाम टेक्स्ट को क्या नियंत्रित करता है?
  - ❑ `updateResult` फंक्शन।
- ❑ हम `updateResult` फंक्शन को कब कॉल (या उपयोग) करते हैं?
  - ❑ कंडीशनल वक्तव्य में तब मूल्यांकन करता है जब `questionCount` 3 के बराबर होता है।
- ❑ क्या कंडीशनल वक्तव्य तब चलता है जब हमें उसके चलने की जरूरत होती है?
  - ❑ नहीं!

## चरण 8: एक बार में एक चीज की जाँच करें

अवधारणा #1: bee फंक्शन में questionCount कंडीशनल की जाँच करना

```
58 // बी स्कोर को ट्रैक करें
59 फंक्शन bee() {
60     beeScore += 1;
61     questionCount += 1;
62     console.log("questionCount = " + questionCount + "\t" + "beeScore = " +
        beeScore);
63     // कंडीशनल जोड़ें
64     if (questionCount == 3){
65         updateResult();
66     }
67 }
```

अवधारणा #2: शेष फंक्शनों में questionCount कंडीशनल की जाँच करना

```
69 // बटरफ्लाई स्कोर को ट्रैक करें
70 फंक्शन butterfly() {
71     butterflyScore += 1;
72     questionCount += 1;
73     console.log("questionCount = " + questionCount + "\t" + "butterflyScore = "
        + butterflyScore);
74     // कंडीशनल जोड़ें
75     if (questionCount == 3){
76         updateResult();
77     }
78 }
79
80 // ग्रासहॉपर स्कोर को ट्रैक करें
81 फंक्शन grasshopper() {
82     grasshopperScore += 1;
83     questionCount += 1;
84     console.log("questionCount = " + questionCount + "\t" + "grasshopperScore = "
        + grasshopperScore);
85     // कंडीशनल जोड़ें
86     if (questionCount == 3){
87         updateResult();
88     }
89 }
```

```

91 // लेडीबग स्कोर को ट्रैक करें
92 function ladybug() {
93     ladybugScore += 1;
94     questionCount += 1;
95     console.log("questionCount = " + questionCount + "\t" + "ladybugScore = "
96         + ladybugScore);
97     // कंडीशनल जोड़ें
98     if (questionCount == 3){
99         updateResult();
100     }

```



### बग फिक्स हो गया!

इस त्रुटि में, प्रोग्राम ने कभी भी उस कंडीशनल का मूल्यांकन नहीं किया जिसने `updateResult` फंक्शन को कॉल किया था। हालांकि सभी बग स्कोर फंक्शनों के बाद `questionCount` मूल्य की जाँच करना तर्कसंगत लग सकता है, हमें वास्तव में हर बार जब कोई बग स्कोर फंक्शन चलता है तब इसकी जाँच करने की जरूरत है। हमने प्रत्येक बग स्कोर फंक्शन के भीतर कंडीशनल को रखकर समस्या को हल किया ताकि प्रोग्राम के मूल्य की जाँच कर सके `questionCount` व्यक्ति के द्वारा किसी बटन को क्लिक करने के बाद.