🧠 Approaching Debugging

Author(s): Hayden Smith

(Download as PDF)

 My Code Isn't Working!

Has this happened to you? Well this lesson is about how to manage a situation where your code isn't working.

First we need to unpack the stages of resolution.

📅 Stages Of Resolution

The stages of resolution can be broken up into:

1. 💬 **What** is going wrong
2. 📸 **Where** is it going wrong
3. 🏫 **Why** is it going wrong

Easter egg

📅 Stages Of Resolution

The stages of resolution can be broken up into:

1. 💬 **What** - EASY 🐱
2. 📸 **Where** - EASY 🐱
3. 🏫 **Why** - HARD 💀

Easter egg

🧠 Running The Code

```
 1 function frequency(list) {
 2   const obj = {};
 3   for (const item of list) {
 4     if (!(item in obj)) {
 5       obj[item] = 1;
 6     }
 7     obj[item]++;
 8   }
 9   return obj;
10 }
11 frequency(['a','a','b','b','c'])
```

debug_1.1.js

Always ask yourself - what is the expected output?

🧠 Running The Code

```
1 function frequency(list) {
2   const obj = {};
3   for (const item of list) {
4     if (!(item in obj)) {
5       obj[item] = 1;
6     }
7     obj[item]++;
8   }
9   return obj;
10 }
11 frequency(['a','a','b','b','c'])
```

debug_1.1.js

Always ask yourself - what is the expected output?

```
1 {
2   'a': 2,
3   'b': 2,
4   'c': 1
5 }
```

🧠 Running The Code

```
 1 function frequency(list) {
 2   const obj = {};
 3   for (const item of list) {
 4     if (!(item in obj)) {
 5       obj[item] = 1;
 6     }
 7     obj[item]++;
 8   }
 9   return obj;
10 }
11 frequency(['a','a','b','b','c'])
```

debug_1.1.js

But this is the actual output...

```
1 {
2   'a': 3,
3   'b': 3,
4   'c': 2
5 }
```

💬 What

```
{
  'a': 2,
  'b': 2,
  'c': 1
}
```

```
{
  'a': 3,
  'b': 3,
  'c': 2
}
```

The expected output appears to be 1 higher than we want it to be

📸 Where

The reality here is that you will just end up putting a bunch of "print" (`console.log`) statements everywhere.

📸 Where

The reality here is that you will just end up putting a bunch of "print" (`console.log`) statements everywhere.

We need to INCREMENTALLY establish truths. The aim is to slowly, but surely, keep confirming assumptions until something breaks our assumption

📸 Where

📸 Where

```
1  function frequency(list) {
2    const obj = {};
3    console.log(obj);
4    for (const item of list) {
5      if (!(item in obj)) {
6        obj[item] = 1;
7      }
8      obj[item]++;
9    }
10   console.log(obj);
11   return obj;
12 }
13 frequency(['a','a','b','b','c'])
```

debug_1.2.js

We expect the first one to be {} and the second one to be the wrong object.

📸 Where

```
1 function frequency(list) {
2   const obj = {};
3   console.log(obj);
4   for (const item of list) {
5     if (!(item in obj)) {
6       obj[item] = 1;
7     }
8     obj[item]++;
9   }
10  console.log(obj);
11  return obj;
12 }
13 frequency(['a','a','b','b','c'])
```

debug_1.2.js

We expect the first one to be {} and the second one to be the wrong object.

```
1 {}
2 { 'a': 3, 'b': 3, 'c': 2}
```

📸 Where

```
1  function frequency(list) {
2    const obj = {};
3    console.log(obj);
4    for (const item of list) {
5      if (!(item in obj)) {
6        obj[item] = 1;
7      }
8      obj[item]++;
9    }
10   console.log(obj);
11   return obj;
12 }
13 frequency(['a','a','b','b','c'])
```

debug_1.2.js

We expect the first one to be {} and the second one to be the wrong object.

```
1 {}
2 { 'a': 3, 'b': 3, 'c': 2}
```

Let's keep going

📸 Where

Start logging inside the loop

📸 Where

Start logging inside the loop

Main Code

```
 1 function frequency(list) {
 2   const obj = {};
 3   console.log(obj);
 4   for (const item of list) {
 5     console.log('item', item);
 6     if (!(item in obj)) {
 7       obj[item] = 1;
 8     }
 9     obj[item]++;
10   }
11   console.log(obj);
12   return obj;
13 }
14 frequency(['a','a','b','b','c'])
```

debug_1.3.js

Output

```
1 {}
2 item a
3 item a
4 item b
5 item b
6 item c
7 { 'a': 3, 'b': 3, 'c': 2}
```

📸 Where

Start logging inside the loop

Main Code

```
 1 function frequency(list) {
 2   const obj = {};
 3   console.log(obj);
 4   for (const item of list) {
 5     console.log('item', item);
 6     if (!(item in obj)) {
 7       obj[item] = 1;
 8     }
 9     obj[item]++;
10   }
11   console.log(obj);
12   return obj;
13 }
14 frequency(['a','a','b','b','c'])
```

debug_1.3.js

Output

```
1 {}
2 item a
3 item a
4 item b
5 item b
6 item c
7 { 'a': 3, 'b': 3, 'c': 2}
```

Still seems OK

Where

Do even more in the loop

📸 Where

Do even more in the loop

Main Code

```
1  function frequency(list) {
2    const obj = {};
3    console.log(obj);
4    for (const item of list) {
5      console.log('item', item);
6      if (!(item in obj)) {
7        obj[item] = 1;
8      }
9      obj[item]++;
10     console.log(`obj[${item}]`, obj[item])
11   }
12   console.log(obj);
13   return obj;
14 }
15 frequency(['a','a','b','b','c'])
```

debug_1.4.js

Output

```
1  {}
2  item a
3  obj['a'] 2
4  item a
5  obj['a'] 3
6  item b
7  obj['b'] 2
8  item b
9  obj['b'] 3
10 item c
11 obj['c'] 2
12 { 'a': 3, 'b': 3, 'c': 2}
```

📸 Where

Do even more in the loop

Main Code

```
 1 function frequency(list) {
 2   const obj = {};
 3   console.log(obj);
 4   for (const item of list) {
 5     console.log('item', item);
 6     if (!(item in obj)) {
 7       obj[item] = 1;
 8     }
 9     obj[item]++;
10     console.log(`obj[${item}]`, obj[item])
11   }
12   console.log(obj);
13   return obj;
14 }
15 frequency(['a','a','b','b','c'])
```

debug_1.4.js

Output

```
 1 {}
 2 item a
 3 obj['a'] 2
 4 item a
 5 obj['a'] 3
 6 item b
 7 obj['b'] 2
 8 item b
 9 obj['b'] 3
10 item c
11 obj['c'] 2
12 { 'a': 3, 'b': 3, 'c': 2}
```

Oh no! They jump up too early

📸 Where

We can confirm this FURTHER by removing coding

📸 Where

We can confirm this FURTHER by removing coding

Main Code

```
1  function frequency(list) {
2    const obj = {};
3    console.log(obj);
4    for (const item of list) {
5      if (!(item in obj)) {
6        obj[item] = 1;
7      }
8      //obj[item]++;
9    }
10   console.log(obj);
11   return obj;
12 }
13 frequency(['a','a','b','b','c'])
```

debug_1.5.js

Output

```
1 {}
2 { 'a': 1, 'b': 1, 'c': 1}
```

📸 Where

We can confirm this FURTHER by removing coding

Main Code

```
1  function frequency(list) {
2    const obj = {};
3    console.log(obj);
4    for (const item of list) {
5      if (!(item in obj)) {
6        obj[item] = 1;
7      }
8      //obj[item]++;
9    }
10   console.log(obj);
11   return obj;
12 }
13 frequency(['a','a','b','b','c'])
```

debug_1.5.js

Output

```
1 {}
2 { 'a': 1, 'b': 1, 'c': 1}
```

And there is our problem...

📸 Where

```
1  function frequency(list) {
2    const obj = {};
3    for (const item of list) {
4      if (!(item in obj)) {
5        obj[item] = 1;
6      }
7      obj[item]++;
8    }
9    return obj;
10 }
11 frequency(['a','a','b','b','c'])
```

debug_1.6.js

Why

Well this one is easy, we set the initial variable to `1` instead of `0`.

🤓 One More Example

```
1  function sortByAge(users) {
2    return users.sort((a, b) => a.age < a.age);
3  }
4
5  function findMedianAge(users) {
6    if (users.length === 0) {
7      return null;
8    }
9
10   let middleIndex = Math.floor(users.length / 2);
11   if (users.length % 2 === 0) {
12     return (users[middleIndex].age + users[middleIndex - 1].age) / 2;
13   } else {
14     return users[middleIndex + 1].age;
15   }
16 }
17
18 let userList = [
19   { name: "Alice", age: 25 },
20   { name: "Bob", age: 19 },
21   { name: "Charlie", age: 32 },
22   { name: "David", age: 17 }
23   { name: "Hayden", age: 99 }
24 ];
25
26 let sortedUsers = sortByAge(userList);
27 let medianAge = findMedianAge(sortedUsers);
28 console.log("Median age:", medianAge);
```

debug_2.1.js

📝 Summary

- Print "strings" with your console logs to better track
- Use console logs to debug, but start by establishing truths, not searching for errors
- "What" and "Where" are most of the battle, and you can all do it

```
1  function sortByAge(users) {
2    return users.sort((a, b) => a.age < a.age);
3  }
4
5  function findMedianAge(users) {
6    if (users.length === 0) {
7      return null;
8    }
9
10   let middleIndex = Math.floor(users.length / 2);
11   if (users.length % 2 === 0) {
12     return (users[middleIndex].age + users[middleIndex - 1].age) / 2;
13   } else {
14     return users[middleIndex + 1].age;
15   }
16 }
17
18 let userList = [
19   { name: "Alice", age: 25 },
20   { name: "Bob", age: 19 },
21   { name: "Charlie", age: 32 },
22   { name: "David", age: 17 }
23   { name: "Hayden", age: 99 }
24 ];
25
26 let sortedUsers = sortByAge(userList);
27 let medianAge = findMedianAge(sortedUsers);
28 console.log("Median age:", medianAge);
```

debug_2.1.js

🦻 Feedback

Or go to the form here.