3.5.4 Handle Blank, Null, and Mixed-Case User Input

We'll start at the bottom of the list and create a development branch called `bug/player-name`. Remember to branch from the `develop` branch and checkout into the new branch.

IMPORTANT

> In this lesson, we won't give detailed instructions about how to create GitHub feature branches. Refer to previous lessons if you need a refresher.

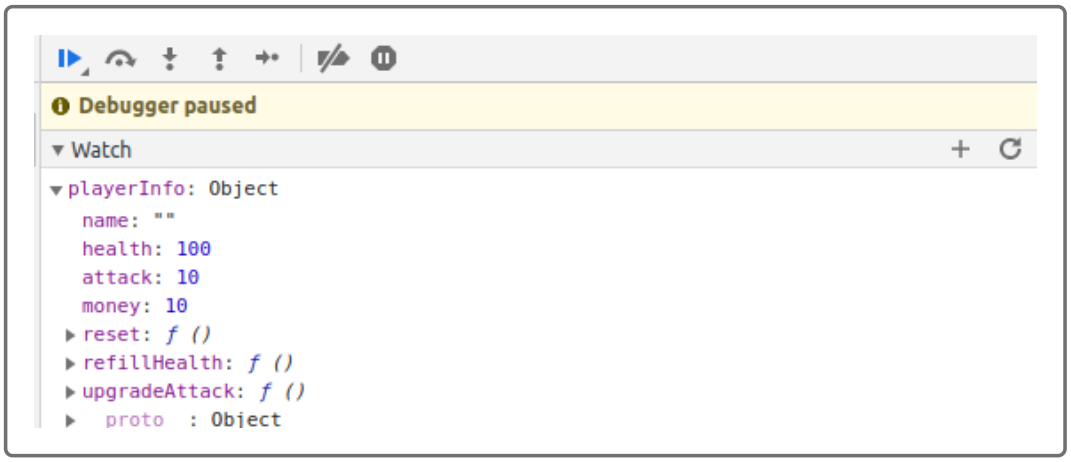
Let's reference the GitHub issue shown earlier for the prompt for the player name. We'll trigger the DevTools debugger by adding the `debugger` keyword to the `for` loop in `startGame()`. Add it after the "Welcome to Battlebots!" alert:

```
    window.alert('Welcome to Battlebots! Round ' + (i + 1));
    debugger;
```

This is a good place to start debugging, becuase we know it's just before the gameplay logic starts.

Reload `index.html` in the browser. When the player name prompt appears, leave the input field blank and press Enter. When you see the "Welcome to Battlebots! Round 1" alert, press Enter again. This will trigger the DevTools debugger.

On the right-hand side of the sources interface, click the "+" in the Watch panel. This will allow us to examine the state of a specific variable. Type "playerInfo" in the resulting field and press Enter. "playerInfo" should appear in the Watch panel. Expand the arrow to the left of "playerInfo" variable. `playerInfo.name` will be either an empty string (`""`), or `null`:



Now we can see for ourselves that `PlayerInfo.name` can be set to a problemataic value. Let's write some code to avoid this situation.

First, let's wrap up our debugging session. Click the "resume execution" button above the Watch panel in DevTools and then "skip" your way through the rest of this game. Finally remove the `debugger` keyword you added to `startGame()` and save the file.

PAUSE

Why is it important to reproduce the errors?

> Seeing the error gives us a clue as to how it happened and how it can be fixed, or squashed. Yes, **bug squashing** is real thing in developer lingo!
>
> Hide Answer

To keep our code organized, we'll start by creating a function called `getPlayerName()` to handle the player name input. This function should do the following:

- Not accept invalid data, such as blank or null
- Prompt the user until valid data is received

The repeated prompting is a clue that the function should contain a loop. So, we'll set up a `getPlayerName()` function with a loop placeholder as a comment. Put this function directly above the `playerInfo` object initialization, because we'll be calling this function within that object; if it isn't encountered before the object is acted on, the object won't be able to find the function and the program will halt:

```
// function to set name
var getPlayerName = function() {
  var name = "";

// **************************************
// ADD LOOP HERE WITH PROMPT AND CONDITION
// **************************************

  console.log(""Your robot's name is "" + name);
  return name;
};
```

Then add a `while` loop with a condition that checks if the `name` is "", which is blank, or null. It's important to initialize `var name = ""` before the

`while` loop to guarantee entering the loop at least once to prompt the user for the player robot name. Replace the loop placeholder with the following:

```
while (name === "" || name === null) {
  name = prompt("What is your robot's name?");
}
```

Because the `getPlayerName()` function will now return a valid player name, we can place the function call in the `playerInfo` object in the `name` property:

```
/* GAME INFORMATION / VARIABLES */
var playerInfo = {
  name: getPlayerName(),
  ... // other playerInfo properties and methods
```

Now let's test our code. First save the changes, then refresh the browser. When you get to the player name prompt, leave the input blank and press Enter. Notice that the prompt simply reappears. The user is now required to enter something into the input field in order to continue. Success!

Excellent work! You're done with this bug fix, so add, commit, and push your work to GitHub. Then checkout into the `develop` branch, and merge your work. The last step is to close the GitHub issue and continue to the next one.