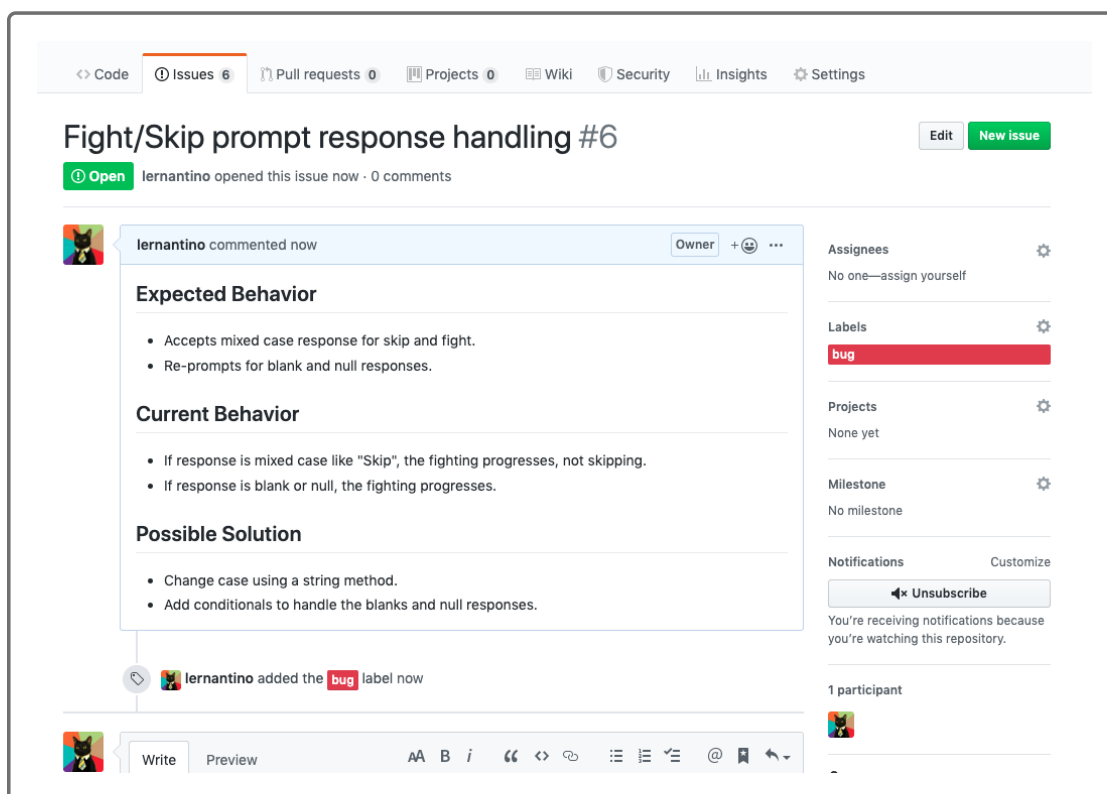


3.5.5 Fix the Prompt Bug in the fight() Function

Here's the issue we'll address next:



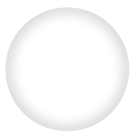
Let's create a branch called `bug/fight-skip` for this issue. Remember to checkout into the `develop` branch before creating this new branch.

While fixing the player name prompt bug, we discovered one way to loop until a valid answer is received. Another approach is to set up a generic function that uses a conditional statement to catch incorrect responses, then execute a function call that prompts the user again.

Here's an example of that type of function:

```
var test = function() {  
  var response = prompt("Question?");  
  if (response === "" || response === null) {  
    window.alert("You need to provide a valid answer! Please try again");  
    test();  
  }  
  return response;  
}
```

A key statement in this function is the recursive call, `test()`, after the `alert()` in the conditional code block. This is known as **recursive** because the function calls itself. It creates a loop that constantly calls itself as long as the conditional statement remains true.



REWIND

We used recursion in the `startGame()` and `shop()` functions when the function was called within the function expression.

Like `while` loops, recursive functions must pay special attention to the conditional statement to break the loop. Otherwise, a stack overflow error will occur, also known as an **infinite loop**.

DEEP DIVE ▲

DEEP DIVE

Recursive functions can be tricky. This looping approach is used in many computer science algorithms, as well as in mathematics, such as in the Fibonacci sequence. For more information, see the [Wikipedia article on recursive functions](https://en.wikipedia.org/wiki/Recursion_(computer_science)) ([https://en.wikipedia.org/wiki/Recursion_\(computer_science\)](https://en.wikipedia.org/wiki/Recursion_(computer_science))).

Let's use the recursive method to handle blank or null responses to the fight-or-skip prompt. We'll create a new function called `fightOrSkip()` that will be responsible for receiving a valid response from the user to determine if the player robot will continue fighting. Let's extract our fight-or-skip conditional logic from the `fight()` function, as well as the `confirmSkip` condition, and place it in the function.

Take a minute to look at the code block below, and then put it into your code above the `fight()` function.

```
var fightOrSkip = function() {  
  // ask user if they'd like to fight or skip using function  
  var promptFight = window.prompt('Would you like FIGHT or SKIP this battle? ');  
  
  // Enter the conditional recursive function call here!  
  
  // if user picks "skip" confirm and then stop the loop  
  if (promptFight === "skip" || promptFight === "SKIP") {  
    // confirm user wants to skip  
    var confirmSkip = window.confirm("Are you sure you'd like to quit?");  
  
    // if yes (true), leave fight  
    if (confirmSkip) {  
      return; // exit the function  
    }  
  }  
}
```

```
    window.alert(playerInfo.name + " has decided to skip this fight.");
    // subtract money from playerMoney for skipping
    playerInfo.playerMoney = playerInfo.money - 10;
    shop();
  }
}
```

What is the recursive function call that will make our function repeat or loop? That's right—you need to add `return fightOrSkip();`. Having the function call itself will make it loop and therefore recursive.

Replace the comment `// Enter the conditional recursive function call here!` with the code that follows:

```
// Conditional Recursive Function Call
if (promptFight === "" || promptFight === null) {
  window.alert("You need to provide a valid answer! Please try again.");
  return fightOrSkip();
}
```

HIDE PRO TIP

There is an alternative and popular shortcut for dealing with non-valid responses: using JavaScript's falsy values. **Falsy values** are values that evaluate to false in a conditional statement. In JavaScript, they include 0, `null`, `""`, `undefined`, `NaN`, and `false`. This shortcut lets you represent all of these non-valid responses in a single expression. For more information, see the [MDN web docs on falsy values in JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/Falsy) (<https://developer.mozilla.org/en-US/docs/Glossary/Falsy>).

To create this conditional statement, let's translate the code block into pseudocode to gain more clarity:

```
// if the `promptFight` is NOT a valid value, then execute
```

We can represent this with the following conditional statement:

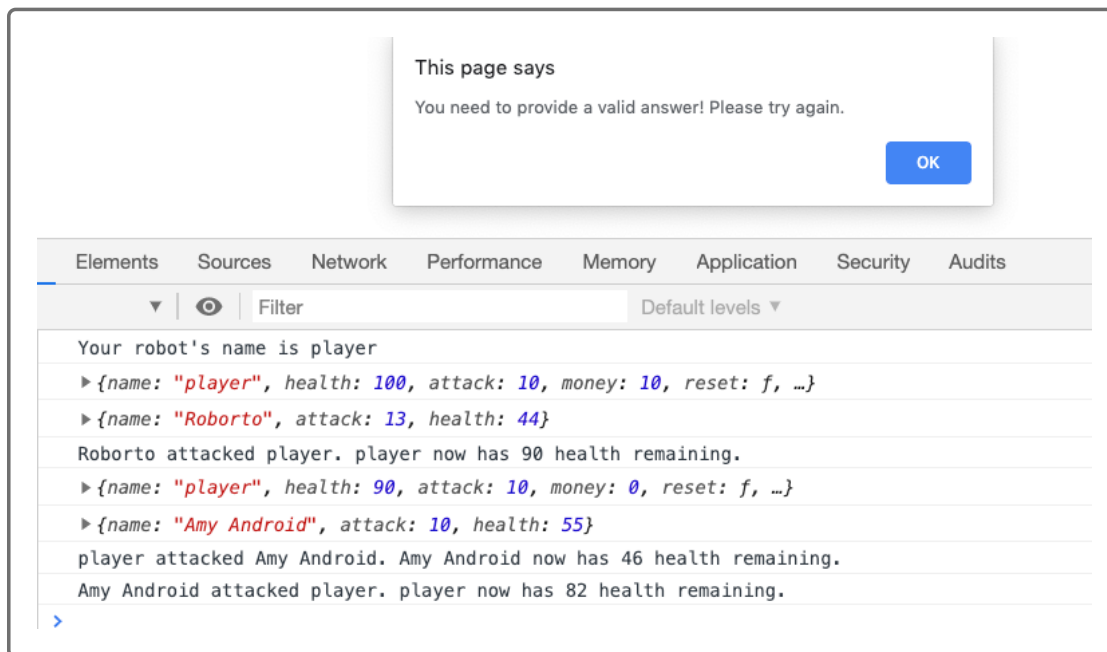
```
if (!promptFight) {  
  window.alert("You need to provide a valid answer! Please try again.");  
  return fightOrSkip();  
}
```

The conditional statement above with the "not" or "!" operator will act similarly to our original conditional `if` statement, which executes if a blank or null response is evaluated.

This new function must be called within the `fight()` function. It replaces the `promptFight` variable declaration and the `if` block that follows it, so delete both of those and replace them with `fightOrSkip()`, like this:

```
// repeat and execute as long as the enemy robot is alive  
while(enemy.health > 0 && playerInfo.health > 0) {  
  fightOrSkip(); // <-- Replace code with this function call  
  var damage = randomNumber(playerInfo.attack - 3, playerInfo.attack);
```

Now, save the file and we'll test the changes to see the results. Reload `index.html` file in the browser.

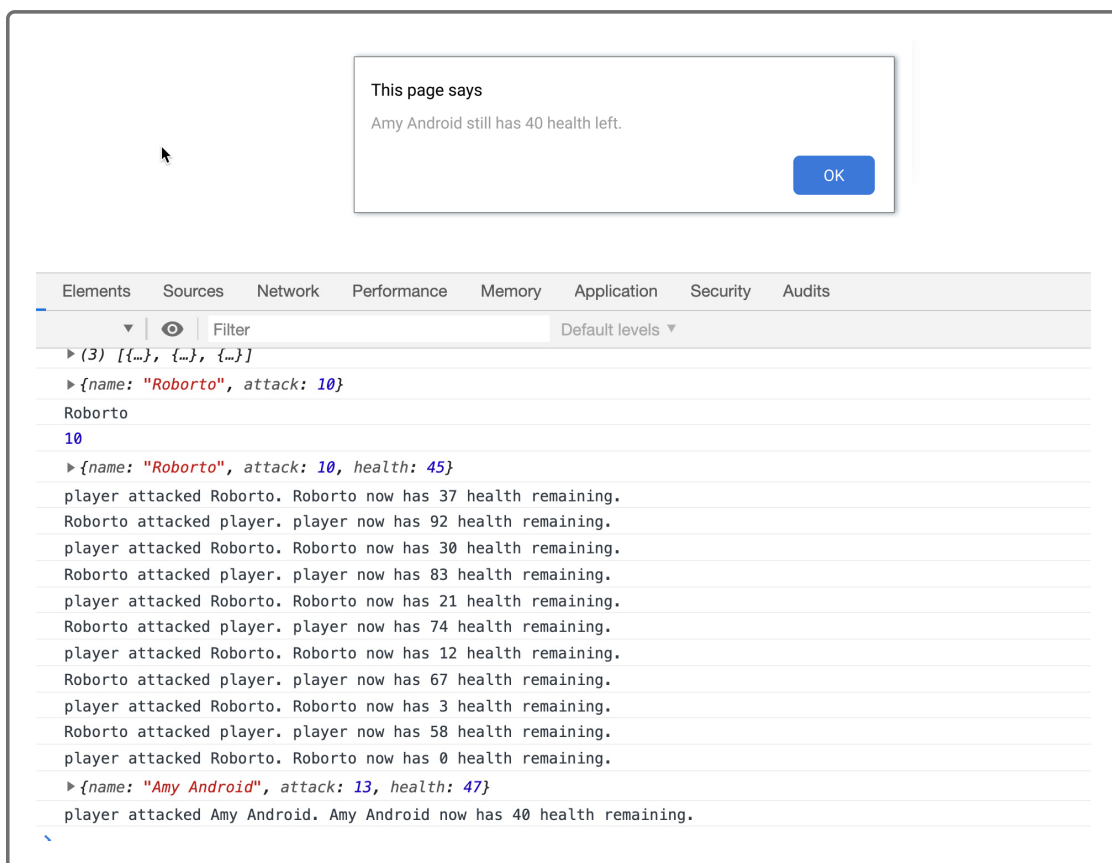
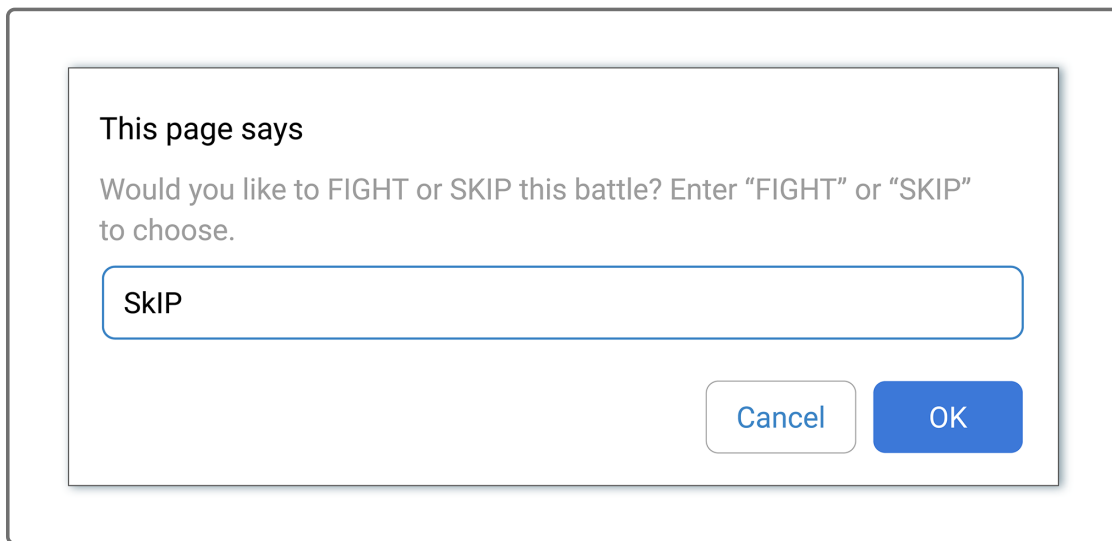


Great job, except this bug is a two parter. We also need to make the fight-or-skip prompt case-insensitive.

Expand Logic to Accept Mixed-Case Strings

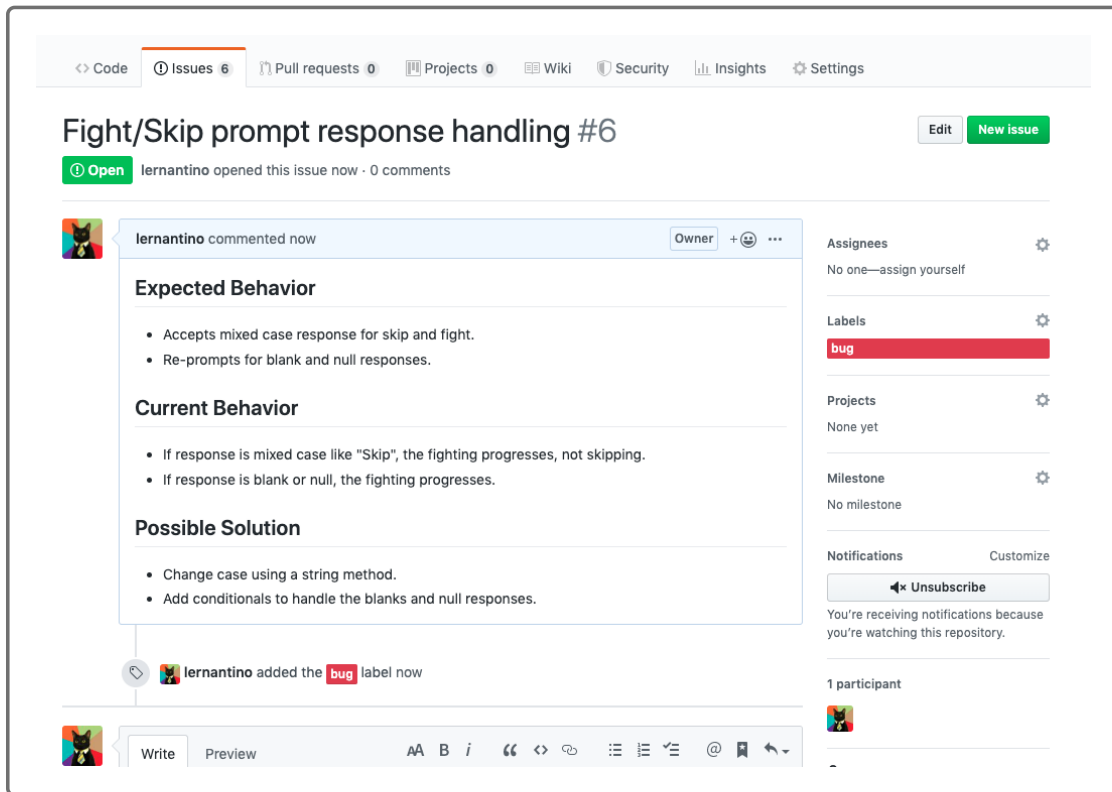
It's important to note here that JavaScript is case-sensitive. This means that the string "Fight" is not the same as "fight", "FIGHT", or "FiGht". But imagine if our user could enter "fight" or "skip" as a response in any variation of mixed upper and lowercase letters.

Let's demo the bug first, then look at the GitHub issue:



In the images above we can note that although a mixed case input for "SKIP" was entered, the player robot fought anyway instead of proceeding to the next round.

Take a look at this [GitHub issue](#) and think about how to solve this bug:



One way to approach this is to change all the characters in the response to lowercase and then have our condition check the lowercase version of the command.

PAUSE

Let's harness our Google powers to find the method that changes characters in a string to lowercase.

In the search results, choose the [MDN web docs on finding a method to convert into lowercase characters](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/toLowerCase) (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/toLowerCase). There, you'll see that there's a method called `toLowerCase()`. Excellent detective work! Notice there is a method for `toUpperCase()` as well. Can you guess what that does?

[Hide Answer](#)

We can use this method to convert the response from our `prompt()`. How will this change our conditional statement? Try to modify it yourself before proceeding.

The conditional statement in the `fightOrSkip()` function that tests whether the response is "skip" or "SKIP" should now be revised to the following:

```
promptFight = promptFight.toLowerCase();  
  
if (promptFight === "skip") {  
    ... // rest of the conditional code block
```

Let's go one step further and have the `fightOrSkip()` function return a value of `true` for a confirmed skip, and `false` otherwise. This refactor will clean up the `while` loop and make it easier to read by creating a function with a singular responsibility to handle the response for the fight-or-skip prompt.

Now we can place a single conditional in the `while` loop in place of the nest of conditional statements we previously had.

Take a moment to place the return statements in their proper locations to return true for a skip and false otherwise. If the user confirms that skipping is their intention, we can return a true value as follows:

```
// if yes (true), leave fight  
if (confirmSkip) {  
    window.alert(playerInfo.name + " has decided to skip this fight.  
    // subtract money from playerMoney for skipping, but don't let t  
    playerInfo.money = Math.max(0, playerInfo.money - 10);  
  
    // return true if user wants to leave  
    return true;  
}
```

The `return false` statement should be located as the last line of the `fightOrSkip()` function.

Now let's replace our `fightOrSkip()` function call in the `fight()` function with the code below to determine whether the player robot will go shopping and stop fighting or continue fighting.

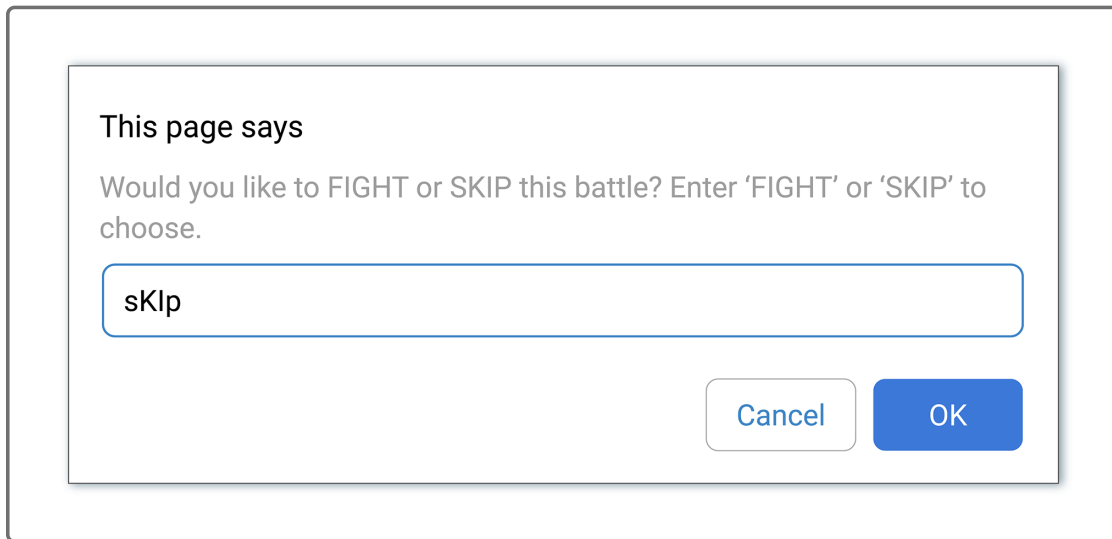
This condition should be placed in the `while` loop, replacing the logic we removed for the fight-or-skip conditional statements. The `while` loop in the `fight()` function should now look like this:

```
while (playerInfo.health > 0 && enemy.health > 0) {  
  // ask user if they'd like to fight or skip using fightOrSkip func  
  if (fightOrSkip()) {  
    // if true, leave fight by breaking loop  
    break;  
  }  
  ... // fight logic
```

Now if the `fightOrSkip()` function returns true, meaning if the player robot wants to skip fighting, then we'll break the `while` loop and return to the `for` loop to face a different robot in a new round. If the `fightOrSkip()` function returns false, the fight continues in the `while` loop.

Test the empty string and mixed case inputs as well as the cancel option in the fight-or-skip prompt to see the changes in the game.

Here's an example of the mixed case input response:



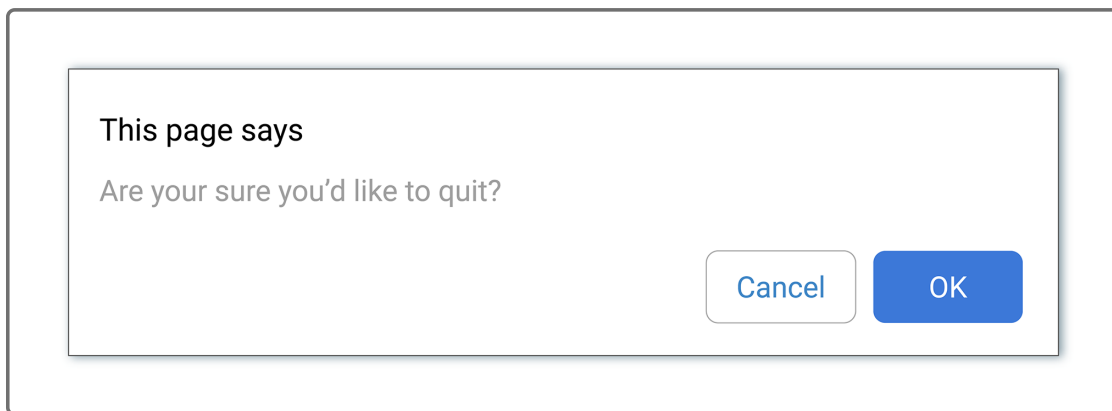
This page says

Would you like to FIGHT or SKIP this battle? Enter 'FIGHT' or 'SKIP' to choose.

sKlp

Cancel OK

A screenshot of a game prompt dialog box. The dialog box has a title bar that says "This page says". Below the title bar, the text reads "Would you like to FIGHT or SKIP this battle? Enter 'FIGHT' or 'SKIP' to choose." There is a text input field containing the text "sKlp". At the bottom right of the dialog box, there are two buttons: "Cancel" and "OK".



This page says

Are your sure you'd like to quit?

Cancel OK

A screenshot of a game confirmation dialog box. The dialog box has a title bar that says "This page says". Below the title bar, the text reads "Are your sure you'd like to quit?". At the bottom right of the dialog box, there are two buttons: "Cancel" and "OK".

In the images above, we can note that the mixed case input still progressed to the confirm quit message which is correct.

Looks good. Let's preserve our work in GitHub and merge it into the `develop` branch. Great job finishing another bug fix! We are making great progress! Let's close this issue and squash the next bug.