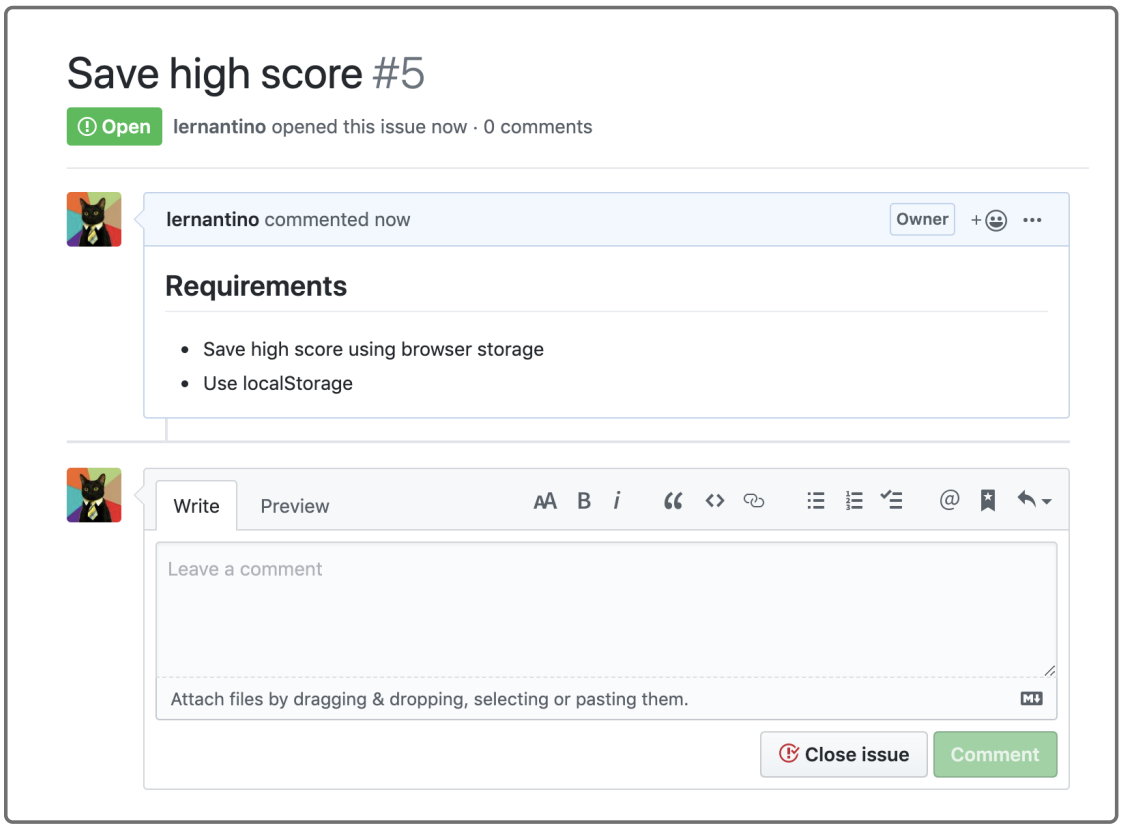
3.5.8 Save and Load High Score from localStorage

With just a few minutes left in our game jam, let's quickly take a look at our last remaining issue and create a feature branch named `feature/highscore`:



Earlier in this lesson, we did some googling to find that `localStorage` allows us to store our high score in the browser. Let's review the **MDN web docs on localStorage (https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Local_storage)** to see how to use this property.

From the documentation, we see that this is a property of the `window` object and that there is also a `sessionStorage` property as well. A big distinction between the two is that where `sessionStorage` will persist the data during the session of the page or while the browser is open, `localStorage` will persist the data in the browser until the data is manually deleted. This means that if you close the browser, the data in `sessionStorage` will disappear but the data in `localStorage` will persist.

The similarities of `sessionStorage` and `localStorage` include the interface methods and the use of key-value pairs to store the object data. Note that the keys and values will always be converted into strings as a condition for browser storage. But what is a key-value pair?

To understand key-value pairs, let's use a familiar object to illustrate them:

```
var playerInfo = {
  name: "Robocop",
  health: 100
}
```

In the `playerInfo` object, the `name` and `health` properties are the **keys**, which are used to access their respective **values**: `"Robocop"` and `100`. It's important to note that `localStorage` only stores strings, so it will force the number "100" to become a string. Therefore, when we retrieve this value, the data type will be a string.

Storage Methods

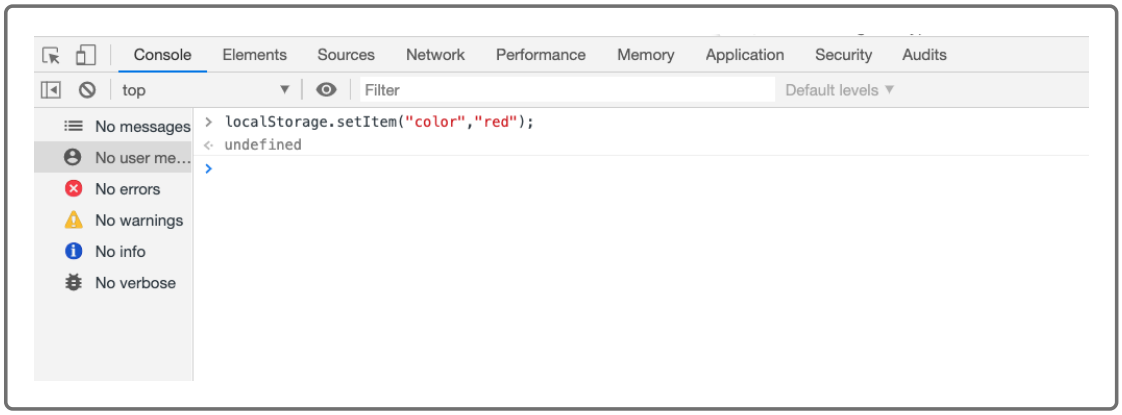
Now that we know we're storing the data as objects, the next step is to use the storage methods to access and enter the object into the browser's storage. This is known as the Web Storage API.

Consider the following code snippet:

```
localStorage.setItem("color", "red");
```

The `setItem` method always receives two arguments. The first argument is the key and the second argument is the value.

Type this example in the console of the browser as shown here:

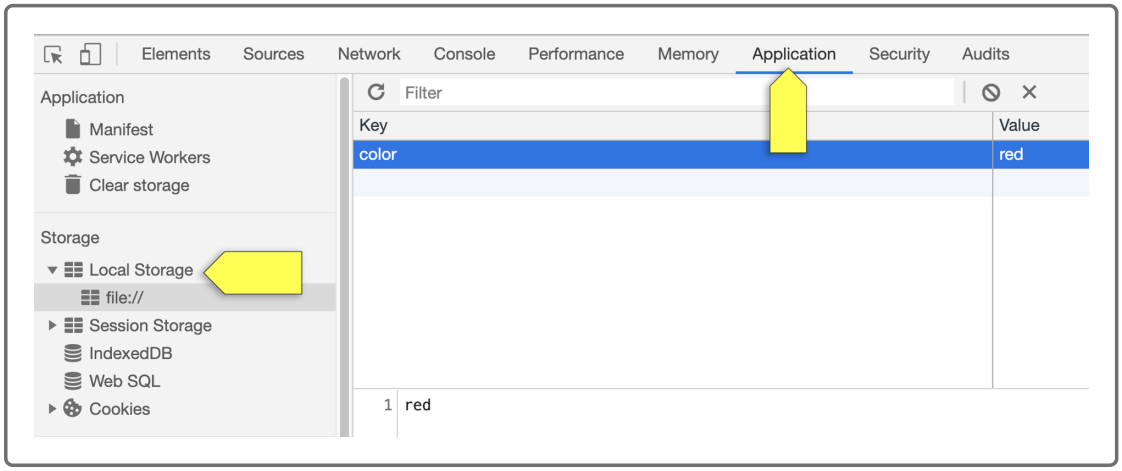


NERD NOTE

We can type simple JavaScript statements into the console because it's part of the browser.

The undefined message shown here indicates that nothing is returned from this statement. This is correct because we're setting or storing our object in `localStorage`. We aren't retrieving or returning anything in this statement.

To see if our statement actually worked, we can use Chrome DevTools. Open the Application tab and click on the Local Storage option:

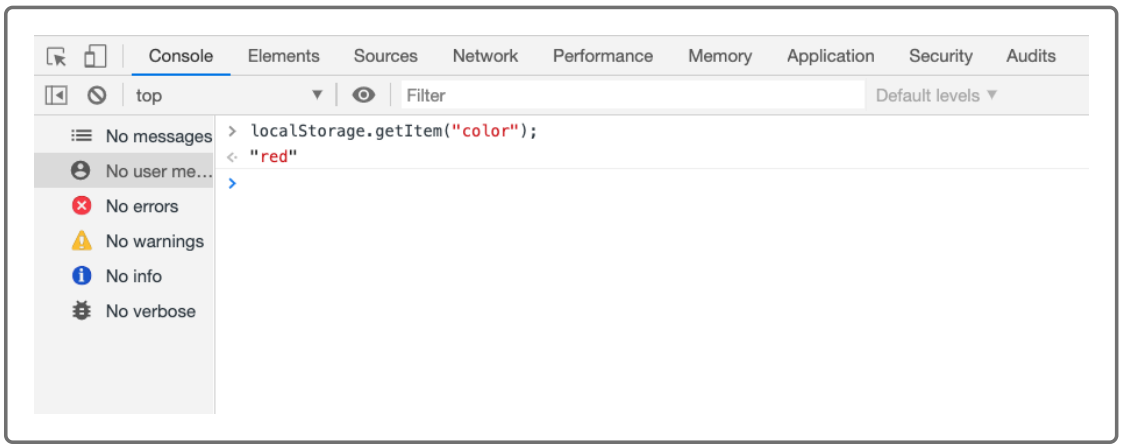


As you can see, the data was stored successfully as a key-value pair. Also note that above the Local Storage option in the left pane is also the option to "Clear storage," which will erase all of the data stored there.

The next step will be to use the Web Storage API to retrieve the value. Let's type the following statement into the console to see our results:

```
localStorage.getItem("color");
```

To retrieve this value from `localStorage`, we use the `getItem` method and the key to return the associated value. The result in the console should look like the following:



Notice from this statement that we receive the value stored at the key as the return, which in this case is "red".

Now that we're familiar with the Web Storage API, how can we use it to store our new high score for our game?

Let's start with pseudocoding this step:

- When the game has ended and we've survived facing all the robots:
    - Retrieve the current high score from `localStorage`
    - Compare the player robot score with the current high score
    - If the current high score is higher
        - Send user the message that the player did not beat the high score
    - If the player score is higher
        - Set new high score object into `localStorage`
        - Set new player robot's name object into `localStorage`
        - Send player the message that they beat the high score

This logic is happening after the game has ended, so which function should contain these operations? That's right—the `endGame()` function!

Use your knowledge of conditional statements and Web Storage APIs to code this operation yourself now. Try not to peek at the code before trying it yourself!

HIDE HINT

Try to retrieve a key from `localStorage` that doesn't exist to see what's returned. You'll get a `null` value. Take into account that no high score has been stored, so if `getItem` retrieves a null, set the current high score to zero.

The final `endGame()` function should now look similar to this:

```
var endGame = function() {
  window.alert("The game has now ended. Let's see how you did!");

  // check localStorage for high score, if it's not there, use 0
  var highScore = localStorage.getItem("highscore");
  if (highScore === null) {
    highScore = 0;
  }
  // if player have more money than the high score, player has new hig
  if (playerInfo.money > highScore) {
    localStorage.setItem("highscore", playerInfo.money);
    localStorage.setItem("name", playerInfo.name);

    alert(playerInfo.name + " now has the high score of " + playerInfo
  }
  else {
    alert(playerInfo.name + " did not beat the high score of " + highS
  }
```

```
  // ask player if they'd like to play again
  var playAgainConfirm = window.confirm("Would you like to play again?

  if (playAgainConfirm) {
    startGame();
  }
  else {
    window.alert("Thank you for playing Battlebots! Come back soon!");
  }
};
```

HIDE PRO TIP

If `highScore` is undefined, then its value is `null`. So, if we want to set `highScore` to 0 if it is undefined, we merely have to test if highScore is null:

```
if (highScore === null) {
  highScore = 0;
}
```

This type of value check is very common, so there is a shorthand notation called a **short circuit conditional statement**:

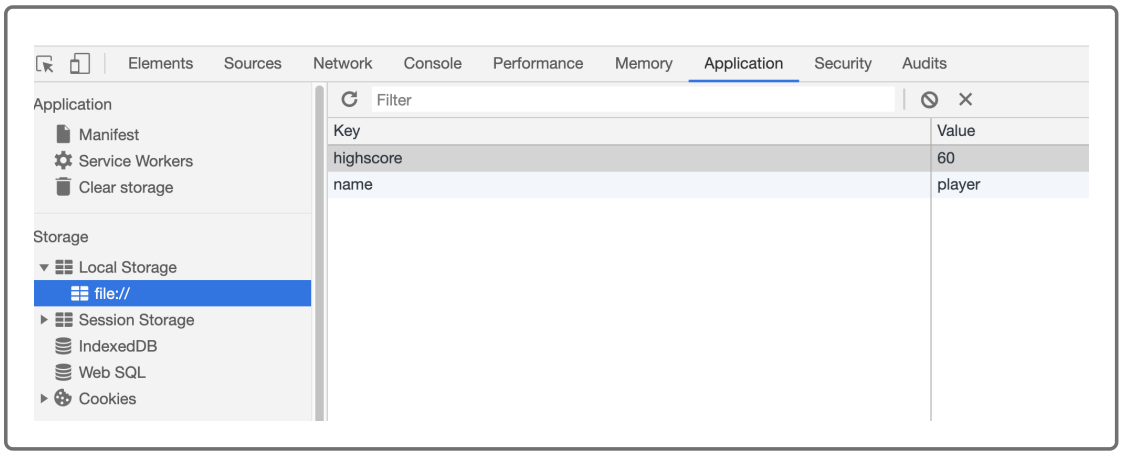
```
highScore = highScore || 0;
```

Here we're using "truthy" and "falsy" values in a short-circuit evaluation and assignment: If the variable on the left of the `||` is truthy, then use it for the assignment value. But if that variable is falsy, then use the value on the right of the `||` for the assignment value.

So, our code means that if the `highScore` value is falsy (e.g., `null`), then assign zero to `highScore`. If not, retain whatever value is currently stored in `highScore`.

For more information, see the **MDN web docs on shortcuts for conditional values (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_Operators)**.

Let's test this latest feature by running the game and seeing if we can get the high score. Save the file, then refresh `index.html` in the browser. Check the console and the Application pane to see if the high score persisted in `localStorage`:



Kudos on a job well done! Make sure this gets into the `develop` branch after you add, commit, and push. Then merge those changes into the `master` branch before the game jam deadline. The judges will surely be impressed with how much you were able to achieve with such a limited amount of time and experience.

Remember to close this issue and then bask in the glory of writing your first JavaScript application! Learning JavaScript can feel difficult at times, but you were able to build this application using some of the most important tools the language has to offer!

There are many resources to learn JavaScript and a lot of them come with opinions on how it should be used, which could lead to us thinking we're not writing our code correctly. Mozilla Developer Network (MDN) is an

authoratative resource, so we'll refer to it often as we learn more about JavaScript. To see how it can be used to your benefit, check out this video on using MDN to learn JavaScript:

With this knowledge, let's take a quick assessment of what we have learned:

Which statement correctly stores data into the Web Storage API?

- ◯ `localStorage.getItem("lunch", "sandwich");`
- ◯ `localStorage.setItem("lunch", "sandwich");`
- ◯ `getItem.localStorage.("lunch", "sandwich");`
- ◯ `setItem.localStorage("lunch", "sandwich");`

Check Answer

Which of the following is NOT a reason to validate a user's responses?

- ◯ Offers the user an opportunity to enter a correct response.
- ◯ Reduces bogus answers getting stored in the database.
- ◯ Improves the user experience.
- ◯ Increases the overall quality of the user data.

Check Answer

Which statement does NOT guarantee that `number` will be non-negative?

- ◯
```
number = Math.max(1, highScore);
```
- ◯
```
if (number < 0) {
  number = 1;
}
```
- ◯
```
number = Math.random();
```
- ◯
```
number = Math.min(10, highScore);
```

Check Answer

Finish ▸