**Guess My Number**

Event Listener

An event listener, listens for a particular event to happen. Here is how to implement one;

A picture containing graphical user interface

Description automatically generated

In this case, we want to ‘listen’ for the point at which the ‘check’ button is clicked by the user.

First we need to select the button. We can do this by using the ‘class’ from our index.html file;

<section class="left">

<input type="number" class="guess" />

<button class="btn check">Check!</button>

</section>

Here we simply need the class ‘.check’. Notice that it is ‘btn check’ but we do not need the first btn.

To select the class we use the following syntax;

document.querySelector(".check")

Now we need to add the ‘event listener’. In this case, it will be when someone clicks on the ‘check’ button. The .addEventListener() is a method so we need to call it using parentheses. Within the parentheses we implement our argument.

document.querySelector(".check").addEventListener('click')

The first argument we pass into parentheses is ‘click’, which will simply listen for the user to click on the button. Now we need to tell the event Listener what to do. We need to specify the reaction to the event being met. We do this by defining a function. The function will be called the **event handler**. This is how the syntax should look;

document.querySelector(".check").addEventListener("click", *function* () {});

Now we need to specify what happens between the curly braces. In this instance, we want to log to the console what has been typed into the input field above the check button.

document.querySelector(".check").addEventListener("click", *function* () {

console.log(document.querySelector(".guess").value);

});

In the console log we select the document and then we select the class which is guess. We then want to print the value to the console, hence; ‘.value’. Remember the event handler is the function that prints to the console, in this instance.

Instead of logging to the console, we could save the value to a variable;

document.querySelector(".check").addEventListener("click", *function* () {

*const* guess = document.querySelector(".guess").value;

});

**Numbers entered by the user are saved as strings**

This is important to remember. Let’s say the user guessed the number 2. We can use the console.log function to show the ‘type’ of variable;

document.querySelector(".check").addEventListener("click", *function* () {

*const* guess = document.querySelector(".guess").value;

console.log(typeof guess);

});

Graphical user interface, text, application

Description automatically generatedAnd here in the console, we can see that 3 is ‘3’ the string and not the number. This is a number guessing game, not a string guessing game. So we will have to convert the string into a number. Here is how;

document.querySelector(".check").addEventListener("click", *function* () {

*const* guess = *Number*(document.querySelector(".guess").value);

console.log(typeof guess, guess);

});

Graphical user interface, application

Description automatically generatedAnd now we can see that we have converted the string to a number using the number() method. Remember that Javascript methods are actions that can be performed on objects. A Javascript method is a property containing a function definition.

**Check for a Value**

This is a number guessing game. Perhaps the user failed to enter a value before clicking check. We can programme a response accordingly. We can use an if statement.

If there is no value added we would get;

Printed to the console. Because 0 is a falsy value, we can use some simple logic to determine if anything has been written in the check box;

We can use the ‘!’ logical not operator;

if (!guess)

Here we have written if the guess value is ‘true’.

Here is how the code block will look;

if (!guess) {

document.querySelector(".message").textContent = `No Number! 🙈`;

}

});

The if statement will only execute if the ‘guess’ statement is evaluated to a truthy value. Because a value of 0 is falsy, in this case, it would evaluate to false. We use the ‘!’ logical not operator to invert the value; in this case, ‘falsy’ would be true. Hence if nothing is entered into the check number box before clicking check, we would end up with the ‘no number’ message.

**Implementing Game Logic**

What happens when we guess correctly, or too low, or too high? The point of game logic is to code rules that account for these scenarios.

The Secret Number

We already have a handler function. Do we want to implement our secret number within this function?

No. The reason is that we want the secret number to be determined when we start the programme. We do not want the secret number to change every time you press the ‘check’ button.

Here is how we can generate a secret number;

*const* number = Math.round(Math.random() \* 20);

The Math.random() operator will generate a random number between 0 and 1. We can then multiply this number by 20 to get a number between 0 and 20. The problem is that this number will not be an integer (a whole number). We can then use the Math.round() operator to round our number to the nearest whole number.

**Assessing our number**

We can use our if/else block from earlier to begin to code some scenarios;

if (!guess) {

document.querySelector(".message").textContent = `No Number! 🙈`;

} else if (guess === secretNumber) {

document.querySelector(".message").textContent =

"🐒Congratulations you guessed correctly!🐦";

} else if (guess >= secretNumber) {

document.querySelector(

".message"

).textContent = `Better luck next time! Your guess was too high. 🙊`;

} else if (guess <= secretNumber) {

document.querySelector(

".message"

).textContent = `Your guess was too low! 🐝`;

}

});

We have now told JavaScript to act in a certain way if our guess was too high or low.

**Changing the Score**

If we guess correctly, we need to increase our score. If we guess incorrectly, we need to decrease our score!

<p class="message">Start guessing...</p>

<p class="label-score">💯 Score: <span class="score">20</span></p>

<p class="label-highscore">

🥇 Highscore: <span class="highscore">0</span>

Here is our html. And here is how our html looks in the browser;

Graphical user interface, text

Description automatically generatedIt is now obvious which code we need to change. The ‘score’ class as well as the ‘highscore’ class. We could write out another if/else block, however that would be time consuming. We can use the existing if/else block.

The easiest way to start manipulating the score is to first create a Score: and a Highscore: variable. That way we can manipulate the number and not have to worry about converting into a string.

**Guessing Correctly**

highScore = highScore + 1;

document.querySelector(".highscore").textContent = highScore;

If we guess correctly, we need the highscore to increase by 1. We also want to present the new high score. Hence the querySelector that changes the highscore class.

**Guessing Incorrectly**

score = score - 1;

document.querySelector(".score").textContent = score;

As above, we need to change the score accordingly.

**Stopping The Score from dropping too low**

Our score could technically keep dropping infinitely. What if we wanted to hard code a limit; say 0? We could also create a message that says, ‘bad luck, you lost’.