**Javascript Pig Game**

A picture containing diagram

Description automatically generated

We are making a pig game that is essentially a dice rolling game. Here is what it looks like.

**Create random number generator**

*let* dice = Math.ceil(Math.random() \* 6);

This code will generate a random number between 0 and 1. It will then multiply it by 6 and then we use the Math.ceil method to round up. We don’t want to round down otherwise we might get the number 0. In effect, this will generate a random number between 1 and 6.

We will have to select players repeatedly throughout our script. We can do this simply by creating variables that store the selectors.

In our HTML script we can see the following:

<body>

<main>

<section class="player player--0 player--active">

<h2 class="name" id="name--0">Player 1</h2>

<p class="score" id="score--0">43</p>

For our player 0 we can see that score has an id of score- - 0. We can use this to manipulate the score.

Score Selector Variables

*const* score0El = document.querySelector("#score--0");

*const* score1El = document.getElementById("score--1");

Here are our score selector variables. Notice that the syntax is different. While they both do the same job, ‘getElementById’ is slightly faster and more appropriate when we are working with 1000’s of different id’s.

Now we can use this to perform certain functions.

score0El.textContent = 0;

score1El.textContent = 0;

This would change the displayed scores to 0.

Text

Description automatically generated

Remove The Dice

If we want to remove the dice from the starting position we first need to add a hidden class to the CSS.

.hidden {

*display*: none;

}

Now we need to use this CSS code;

diceEl.classList.add('hidden');And we do this by adding this line of code to our script.js

Diagram

Description automatically generatedIf we look at our flow chart. We can see that one of our first steps is to roll the dice and generate a random number. We did this earlier with the following code:

*let* dice = Math.ceil(Math.random() \* 6);

Now we need a way of displaying this dice roll.

Display the Dice Roll

When we roll the dice, we want to display the dice!

We could do this by removing the hidden class.

Firstly we need to perform a function when we click on our buttons. For simplicity; it would be sensible to create variables that save our buttons;

*const* btnNew = document.querySelector(".btn--new");

*const* btnRoll = document.querySelector(".btn--roll");

*const* btnHold = document.querySelector('.btn--hold');

Now we can use the saved variables to perform specific tasks.

When we roll the dice we will interact with the ‘roll’ button. We will then need to select an eventListener; which in this case will be ‘click’ followed by a function;

btnRoll.addEventListener("click", *function* () {

//1. Generate random dice roll.

*let* dice = Math.ceil(Math.random() \* 6);

//2. Display dice

//3. Check for rolled 1. If true, switch to next player.

});

Here we have implemented our instructions.

**Random Dice Roll**

We want the dice roll to be a local variable and not a global variable. This is so the number will continually change every time we click on the btnRoll.

**Display Dice**

The first thing we will need to do is remove the starting condition. Remember that the starting condition specified that the dice should be hidden?

//2. Display dice

diceEl.classList.remove('hidden');

Now we need to display one of the saved images we have.

Text

Description automatically generated

Here are our images. If we look at our html file we can see that the source of the original file is;

<img src="dice-5.png" alt="Playing dice" class="dice" />

Which means it will be displaying our dice 5.png. The src attribute is how we define which image should be displayed.

It is the src attribute that we need to manipulate if we want to show a different image.

diceEl.src = `dice-${dice}.png`;

And here is how we do that. Notice that we are not manipulating a classList, we are manipulating the src property. We then use a template literal so that we can add ${dice} variable.

Now when we roll the dice our dice number should displayer in the middle of the game;

Timeline

Description automatically generated with low confidence

Great, it works.

**Rolling a 1**

The purpose of the game is to NOT roll a 1. If we rolled a 6, we would add this number to the current score.

Firstly we need to save our score value. We can’t do it in the btnRoll function as this would only effect the local scope. We need to store it in the global scope. We can do this by creating a variable.

//Starting Conditions; Score 0.

score0El.textContent = 0;

score1El.textContent = 0;

diceEl.classList.add("hidden");

*let* currentScore = 0;

We can save our currentScore variable within the starting conditions block.

Now we have saved our variable we need to change it;

if (dice !== 1) {

//If dice score is NOT 1, add number to current score.

currentScore += dice;

} else {

//Stay on player.

}

});

Every time we roll the dice if the number is not a 1 we add that number to currentScore.

We now need to display the score.

**Display the Score**

Let us take player0 as an example. We need to see how the score is saved. Let’s look at the ‘current’ score card;

Graphical user interface, application, website

Description automatically generated

Notice we can use the browsers built in functionality to determine that the id for player 1 is; ‘current—0’.

We now need to create two new variables;

*const* current0El = document.getElementById('current--0');

*const* current1El = document.getElementById('current--1');

These variables will allow us to manipulate the classid’s.

if (dice !== 1) {

//If dice score is NOT 1, add number to current score.

currentScore += dice;

current0El.textContent = currentScore;

We now need to set current0El to our currentScore value. This is the correct syntax.