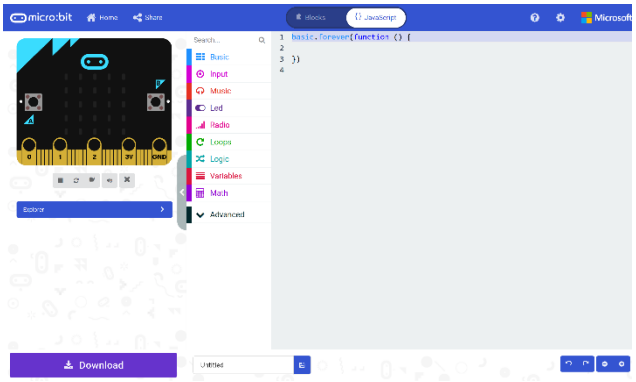
Shaking Dice

Step 1



Go to **microbit.org**, then select the **Let's Code** option, next in the **MakeCode Editor** section select the **Let's Code** button, finally select the **New Project** button and select the **JavaScript** tab

Step 2

With the **JavaScript** tab selected in the **MakeCode Editor** you should remove the following code:

```
basic.forever(() => {

})
```

Then in the **MakeCode Editor** you should enter the following code:

```
const faces: number[][] =
[
    [0, 0, 0,
     0, 1, 0,
     0, 0, 0],
    [1, 0, 0,
     0, 0, 0,
     0, 0, 1],
    [1, 0, 0,
     0, 1, 0,
     0, 0, 1],
    [1, 0, 1,
     0, 0, 0,
     1, 0, 1],
    [1, 0, 1,
     0, 1, 0,
     1, 0, 1],
    [1, 0, 1,
     1, 0, 1,
     1, 0, 1]
];
```

const means a value that doesn't change and this is set to a **number[][]** called **faces** which is special kind of value known as an array, which is a list of items, however is extra special as it is a list of lists of those items as well. These can be identified with the use of two sets of two square brackets, here it is a list of number, where each of those lists is also in a list. Don't worry if that sounds confusing – imagine it like a grid where you can go up and down and side to side, when you get an item from this array with an index, or position, this item is also an array – that's the up and down part, and when you get an item from that array item with an index or position – that's the side to side part.

Step 3

While still in the **JavaScript** tab of the **MakeCode Editor**, below the code entered in the previous step, you should enter the following code:

```
let roll: number = 0;
```

let helps create another kind of value, these can change and are known as variables which can contain a single value. Here there's a number which can store a whole **number** such as 0, 1, 2, 3 and so on, it is called **roll** and is set to **0**.

Step 4

Again, while still in the **JavaScript** tab, below the code entered in the previous step, you should enter the following code:

```
function pip(column: number, row: number, item: number) {
    if (item == 0) {
        led.unplot(column, row);
    }
    else {
        led.plot(column, row);
    }
}

function show(item: number) {
    let face: number[] = faces[item];
    pip(1, 1, face[0]);
    pip(2, 1, face[1]);
    pip(3, 1, face[2]);
    pip(1, 2, face[3]);
    pip(2, 2, face[4]);
    pip(3, 2, face[5]);
    pip(1, 3, face[6]);
    pip(2, 3, face[7]);
    pip(3, 3, face[8]);
}
```

function is a block of code that you can use many times to do the same thing, they can also take in values known as parameters to use in the **function**.

1. The first **function** is called **pip** and this takes three **number** parameters – **column**, **row** and **item**. If **item** is **0** then the LED on the **micro:bit** will be turned off at the position specified by column and row – this is what **led.unplot(column,row)** does. Otherwise if **item** is **1** then the LED on the **micro:bit** will be turned on at the position specified by column and row – this is what **led.plot(column,row)** does
2. The second **function** is called **show** and it takes a **number** parameter called **item**. **let** creates a value to use, here called **face** and is set to **number[]**, this is a special kind of value known as an array, which is a list of values, these can be identified with the use of a set of two square brackets, here it is a list of numbers which can be **1** or **0**. To get a particular value from an array you need an index, which is the position in the array, this starts from 0 for the first item and 1 for second and so on, so for example to get the fifth value from the array you'd use the index of 4. We then use the **function** called **pip** where you provide the position as a pair of values for the columns and rows of the LEDs on the front of the **micro:bit**. As you only want to light up some of them to display each side of the dice and that's what the **face** parameter does which contains which LEDs should be on – represented by a **1** and which should be off – represented by a **0**.

Reference - all possible **micro:bit** LED positions are as follows:

```
0,0 1,0 2,0 3,0 4,0
0,1 1,1 2,1 3,1 4,1
0,2 1,2 2,2 3,2 4,2
0,3 1,3 2,3 3,3 4,3
0,4 1,4 2,4 3,4 4,4
```

Step 5

Once again while still in the **JavaScript** tab, below the code entered in the previous step, you should enter the following code:

```
input.onShake(() => {
    roll = Math.randomRange(0, 5);
})
```

input are things that happen, also known as events, when you do something with the **micro:bit**. The **input** here is an event that will happen when you **shake** the **micro:bit** and when you do it will set the **roll** value to a **random** value between **0** and **5**, this is what **Math.randomRange(0, 5)** does.

Step 6

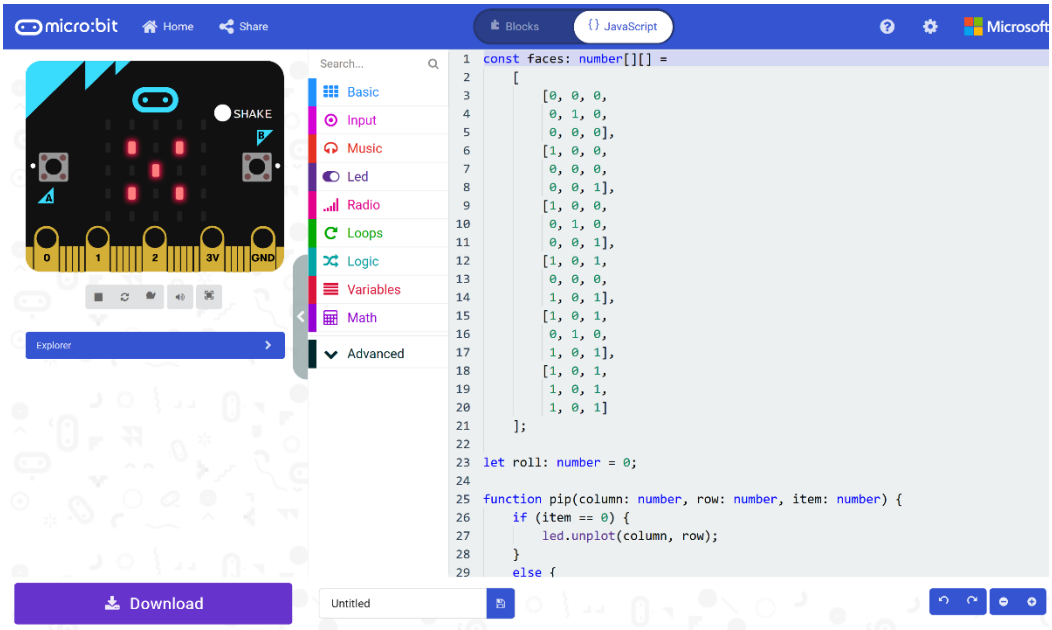
Finally, while still in the **JavaScript** tab, below the code entered in the previous step, you should enter the following code:

```
basic.forever(() => {
    show(roll);
})
```

basic.forever is a **function** which will repeat, or loop, **forever** as long as the example is running on the **micro:bit**. Inside this **function** it will use or call the **function** named **show** – this will display the current value of the **roll** on the **micro:bit** using the LEDs in a pattern that matches the number.

Step 7

Once done the **MakeCode Editor** should appear as follows:

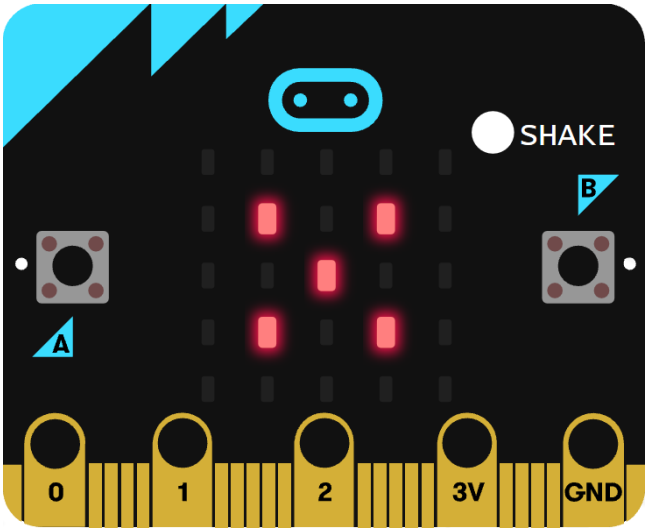


Step 8

That completes the **micro:bit** example, if not done already you can select the **Start the simulator** button to start the example

Step 9

When running on the virtual **micro:bit** you can **shake** the **micro:bit** by pressing the **shake** button on the on-screen **micro:bit** this will randomly select a different face of a dice – it's actually just a "die" in this case as there's just the one. You can shake the real **micro:bit** in your hand to do the same thing.



You can also run the example on an actual **micro:bit** by connecting one to your computer and then choosing the **Download** option in the **MakeCode Editor** to download the example to your computer. Once downloaded you can then copy the **.hex** file from where you've downloaded it to the **micro:bit** the same way you'd copy to another drive or device connected to your computer, then once the example has been copied to the **micro:bit** it should start automatically.