



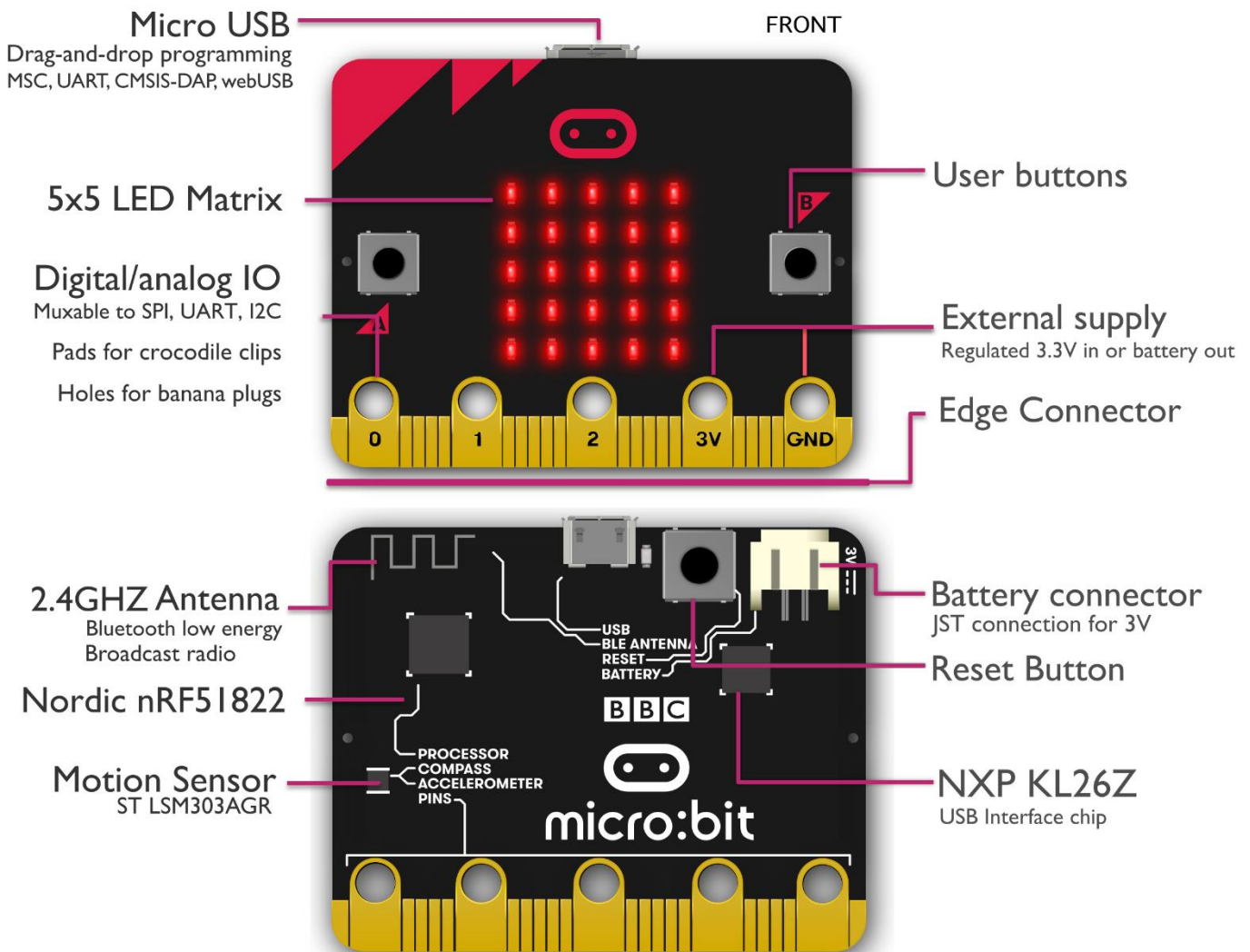
Tutorialr.com

micro:bit

About & Start

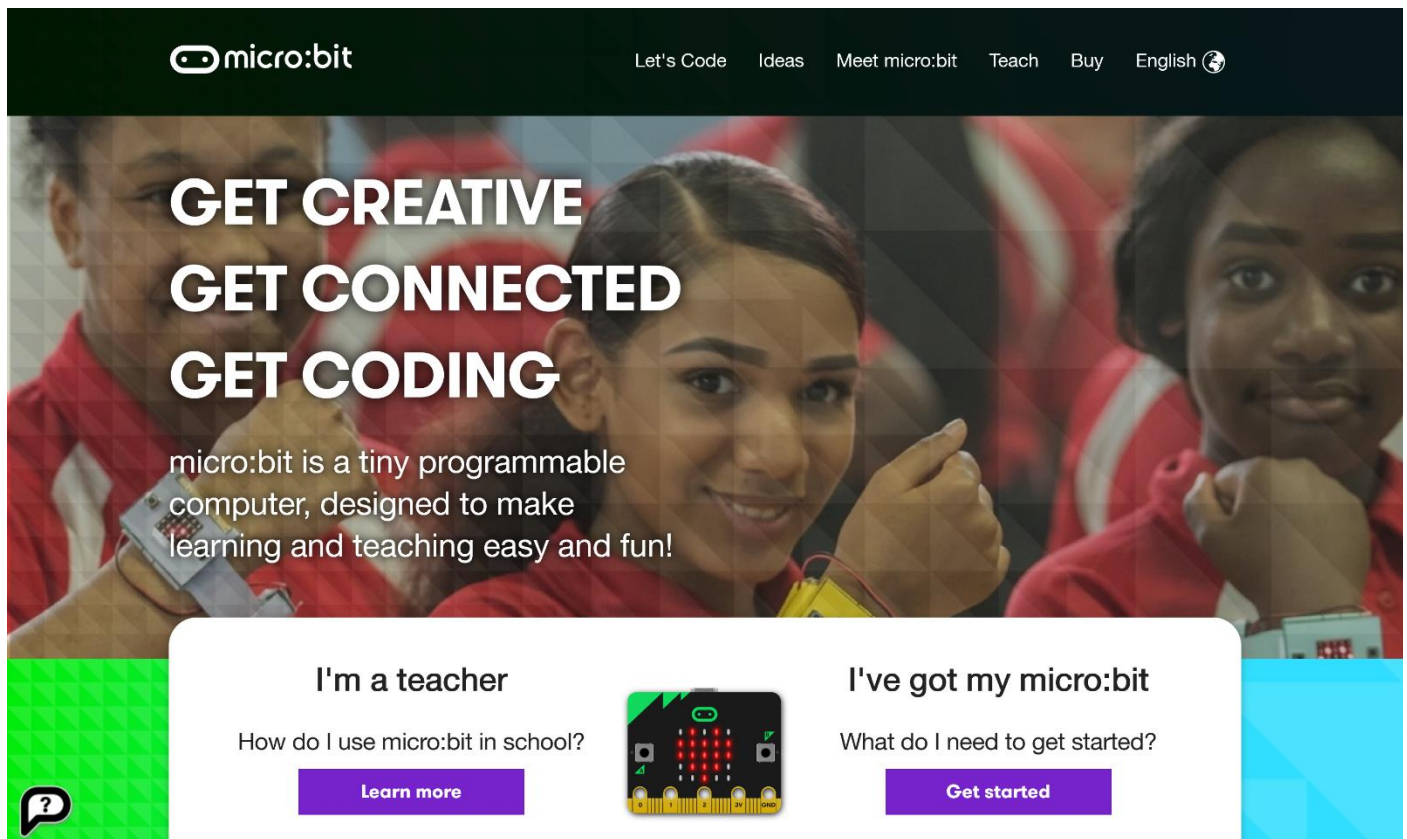
About

micro:bit is a pocket-sized computer that you can code, customise and control to bring all sorts of cool creations to life from robots to musical instruments. **micro:bit** measures 4cm by 5cm and has 25 red LEDs to display messages, two programmable buttons for uses such as controlling games or skipping songs in playlist, it can detect motion and tell which direction it is pointing and can use Bluetooth to communicate with other devices and the Internet.



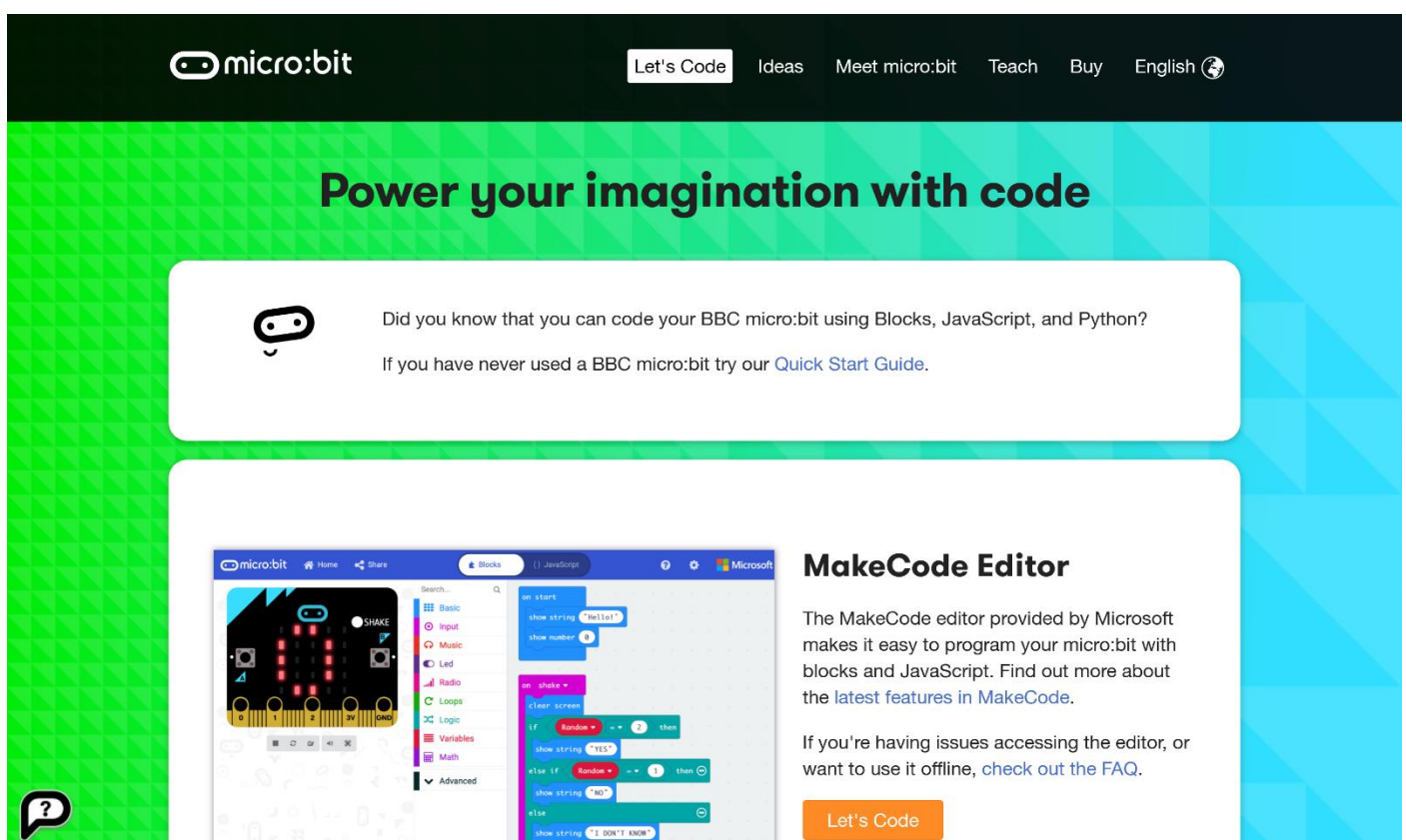
Start

To get started with **micro:bit** and to find out more about it visit microbit.org



The screenshot shows the micro:bit homepage. At the top is a dark navigation bar with the micro:bit logo and links for 'Let's Code', 'Ideas', 'Meet micro:bit', 'Teach', 'Buy', and 'English'. Below this is a large hero section with a background image of three children. The text 'GET CREATIVE', 'GET CONNECTED', and 'GET CODING' is displayed in large white letters. Below this, a paragraph states: 'micro:bit is a tiny programmable computer, designed to make learning and teaching easy and fun!'. At the bottom of the hero section are two white boxes. The left box is titled 'I'm a teacher' and contains the text 'How do I use micro:bit in school?' and a purple 'Learn more' button. The right box is titled 'I've got my micro:bit' and contains the text 'What do I need to get started?' and a purple 'Get started' button. A small micro:bit icon is positioned between the two boxes.

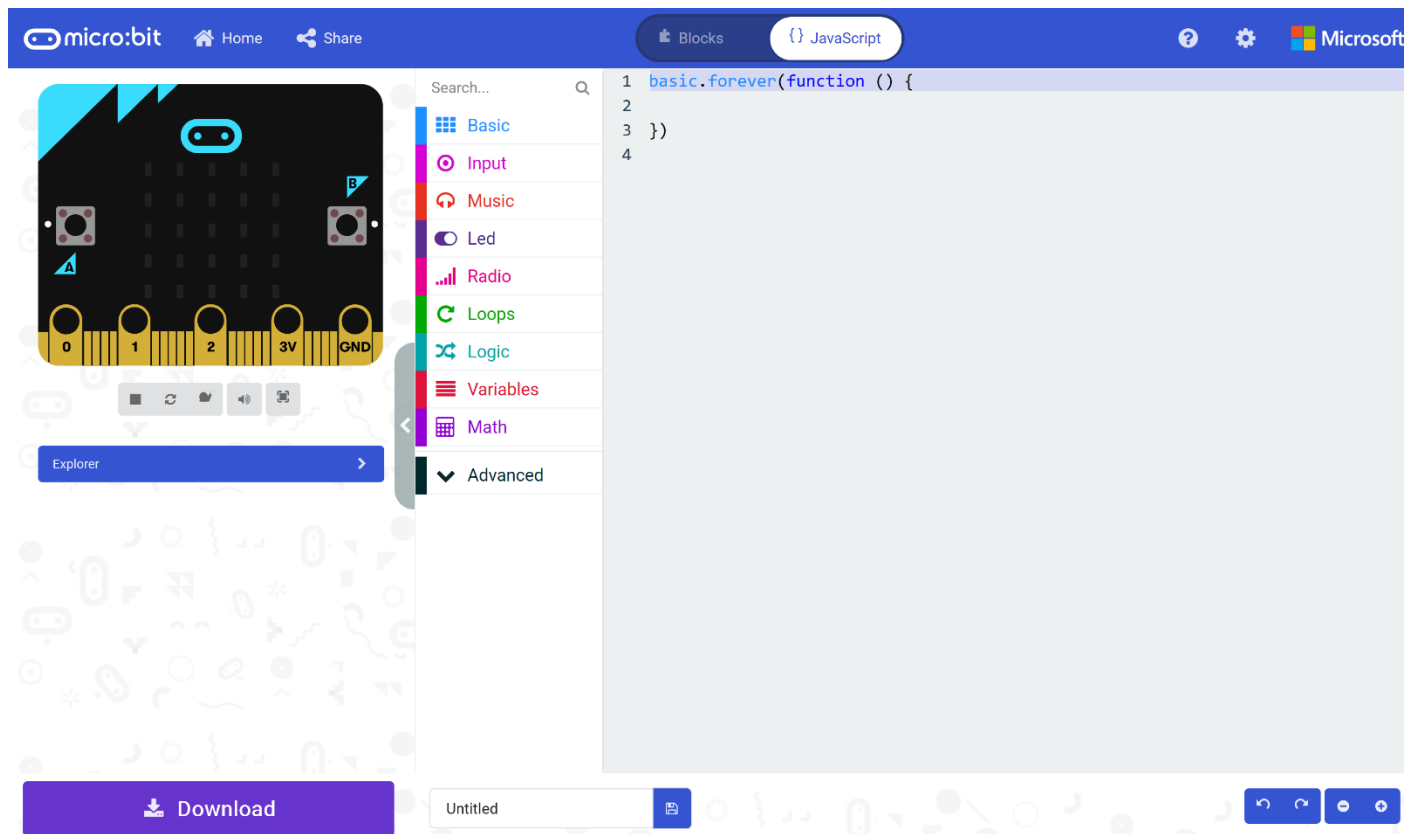
Next to get started with coding with **micro:bit**, once on the website select the **Let's Code** option



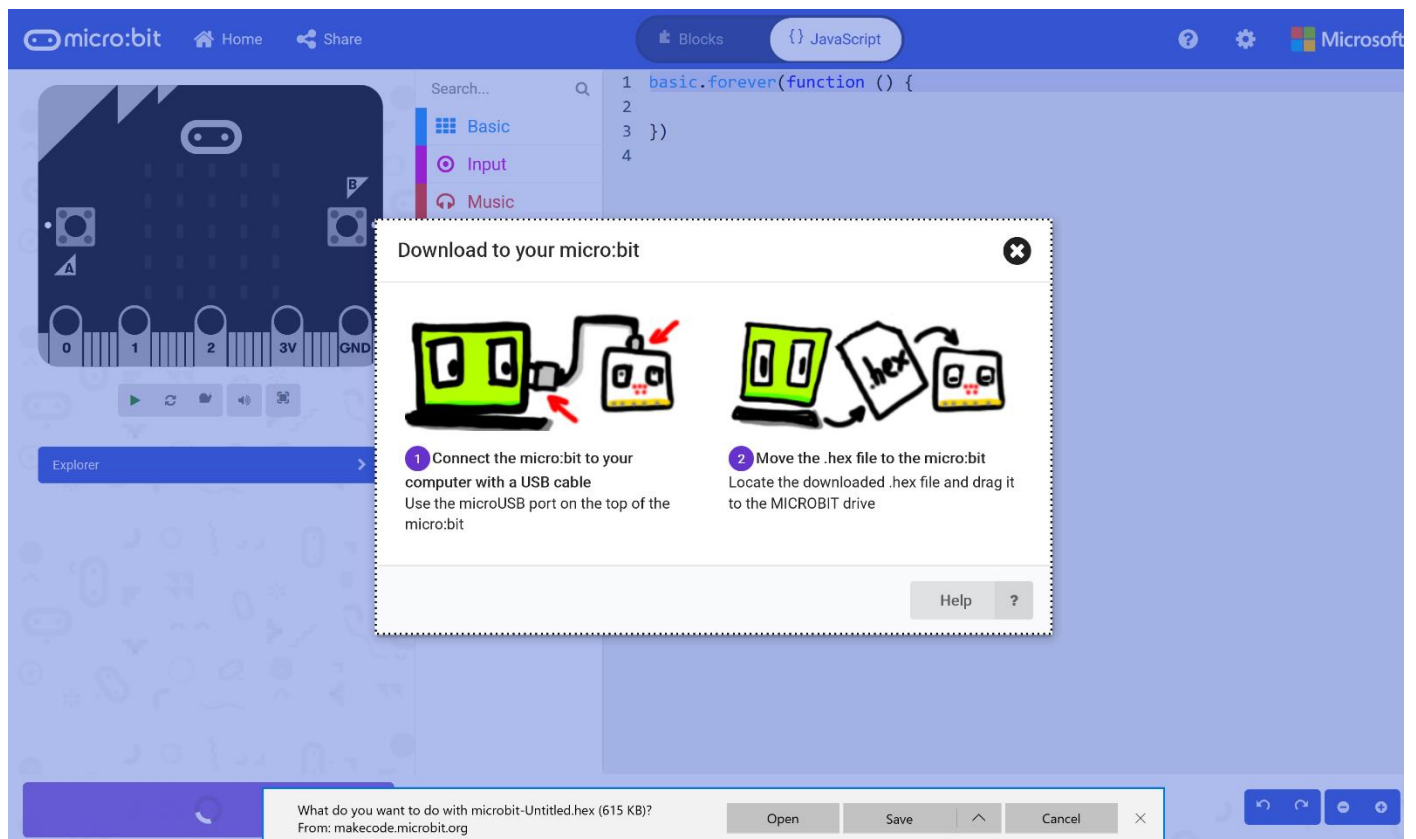
The screenshot shows the 'Let's Code' page on the micro:bit website. The navigation bar is the same as the previous page, but the 'Let's Code' link is highlighted. Below the navigation bar is a green and blue geometric pattern background. The main heading is 'Power your imagination with code'. Below this is a white box containing a micro:bit icon and the text: 'Did you know that you can code your BBC micro:bit using Blocks, JavaScript, and Python? If you have never used a BBC micro:bit try our [Quick Start Guide](#).' Below this is another white box. On the left side of this box is a screenshot of the MakeCode editor interface, showing a micro:bit board and a code editor with JavaScript code. On the right side of this box is the heading 'MakeCode Editor' followed by the text: 'The MakeCode editor provided by Microsoft makes it easy to program your micro:bit with blocks and JavaScript. Find out more about the [latest features in MakeCode](#).' Below this text is a link: 'If you're having issues accessing the editor, or want to use it offline, [check out the FAQ](#).' At the bottom of this box is an orange 'Let's Code' button.

Then from the **Let's Code** page select the **Let's Code** button in the **MakeCode Editor** section

Finally, in the **MakeCode Editor** page select the **New Project** button and then select the **JavaScript** tab

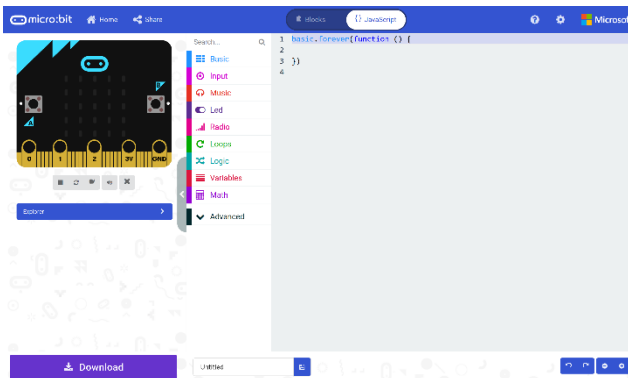


To run an example on a real **micro:bit** select the **Download** button and follow the instructions



Hello World

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 see the following code:

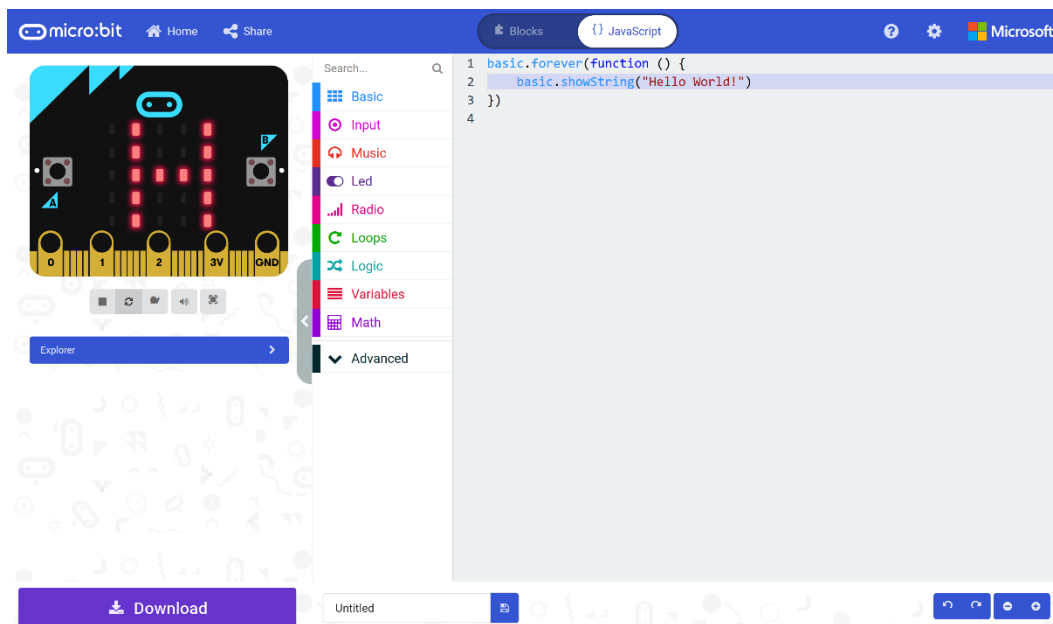
```
basic.forever(() => {  
  
})
```

Within this you should type the following code:

```
basic.showString("Hello World!")
```

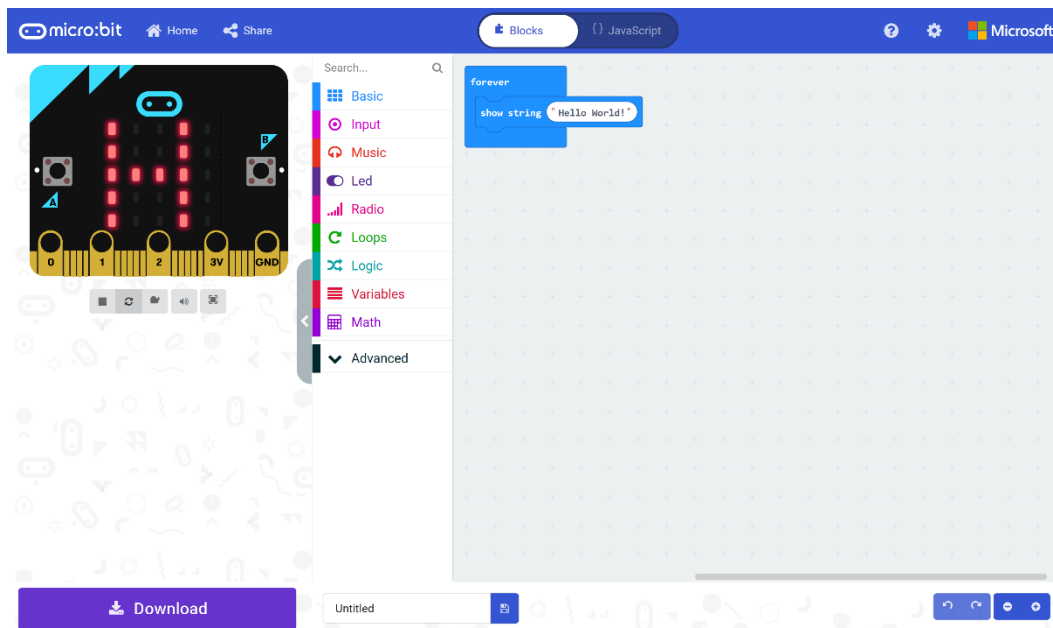
Step 3

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



Step 4

In the **MakeCode Editor** you can also select the **Blocks** tab, which is an easy to use drag-and-drop style of programming for younger or less experienced coders



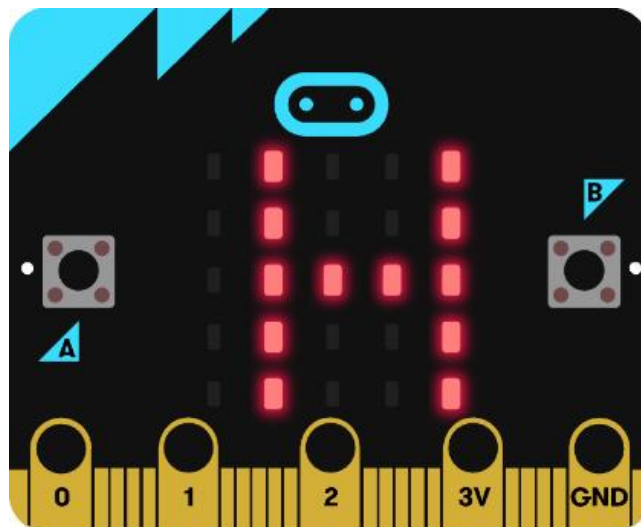
Step 5



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

Step 6

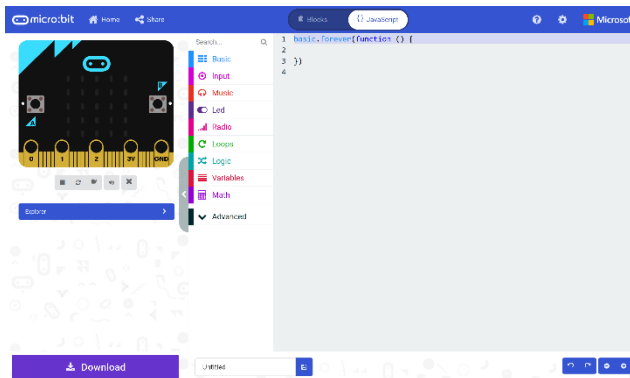
When running on the virtual **micro:bit** the LEDs will scroll through the text **Hello World**



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.

Show Message

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 items: string[] = [  
  "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M",  
  "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z",  
  "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "!", "?", " "  
];
```

const means a value that doesn't change and this is set to a **string[]** called **items** which 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 string. 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.

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 counter: number = 0;
let editing: boolean = false;
let display: string = "";
let message: string = "";
```

let helps create another kind of value, these can change and are known as variables which can contain a single value, there are different types of value used here. First there's a **number** which can store a whole number such as 0, 1, 2, 3 and so on. Then there's a **boolean** this can be either **true** or **false** and here is set to **false**. Then there are two strings, and these can be any kind of text, they have been set to an empty string which is "".

Step 4

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

```
function show(item: string) {
    basic.showString(item)
}
```

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**. The parameter used here is called **item** and it is a **string**. The **function** will use this value to use or call the **basic.showString** built in **function** to display text on the **micro:bit** using the LEDs on the front.

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.onButtonPressed(Button.A, () => {
  if (counter == items.length) {
    counter = 0;
  }
  display = items[counter];
  counter++;
})

input.onButtonPressed(Button.B, () => {
  message += display;
  show("+");
})

input.onButtonPressed(Button.AB, () => {
  editing = !editing;
})
```

input are things that happen, also known as events, when you do something with the **micro:bit**, they are as follows:

1. The first **input** is when you press the **A** button on the **micro:bit** and when you do it will check the **counter** to see if it is the same value as the size or **length** of the **list string[]** array, then it will set **display** to be the item from the array with the same index as the **counter** – if this is the first item it will have the value **"A"**, if second it will be **"B"** and so on, the line **counter++** will increase the value of **counter** by one for the next item.
2. The second **input** is when you press the **B** button on the **micro:bit** and when you do the **message += display** will add the whatever is in the **display** value onto the end of the **message** value, it will display a **"+"** on the **micro:bit** by using the **function show** briefly before continuing.
3. The third **input** is when you press both the **A** and **B** buttons on the **micro:bit** at the same time **editing = !editing** will set the value of **editing** to the opposite of its value, this is what the **!** does which is known as **not**.

Step 6

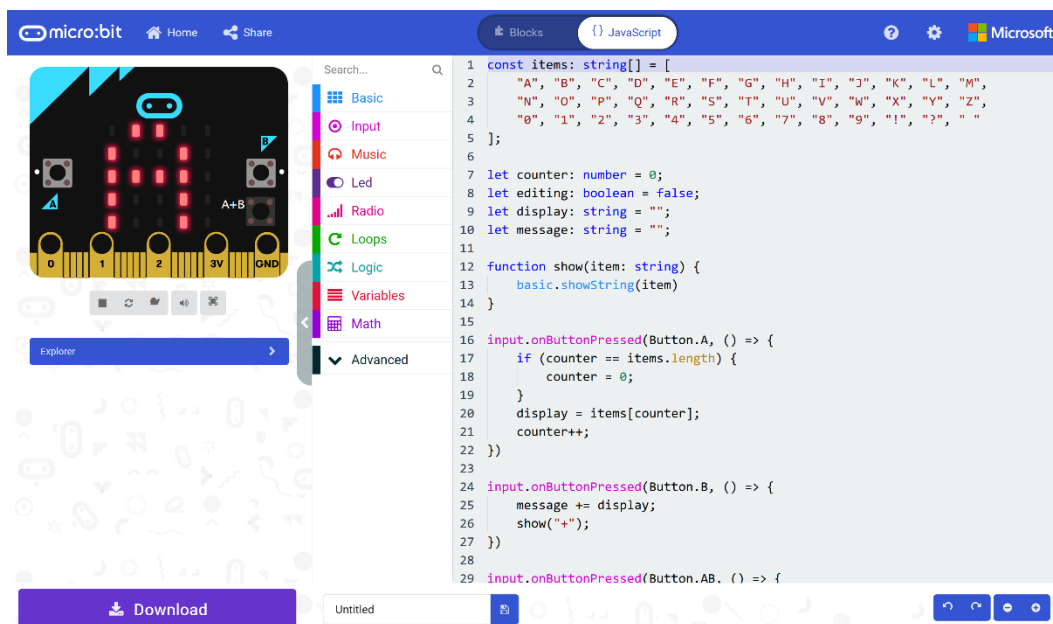
Finally, while still in the **JavaScript** tab, below the code entered in the previous step, you should enter the following code:

```
basic.forever(() => {  
  if (editing) {  
    show(display);  
  }  
  else {  
    show(message);  
  }  
})
```

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 check the **editing** value in an **if** statement, when it is **true** the first part of the code will happen, this will show the **display** value on the **micro:bit** and in the second part, when **editing** is **false** it will show the message value on the **micro:bit** instead.

Step 7

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



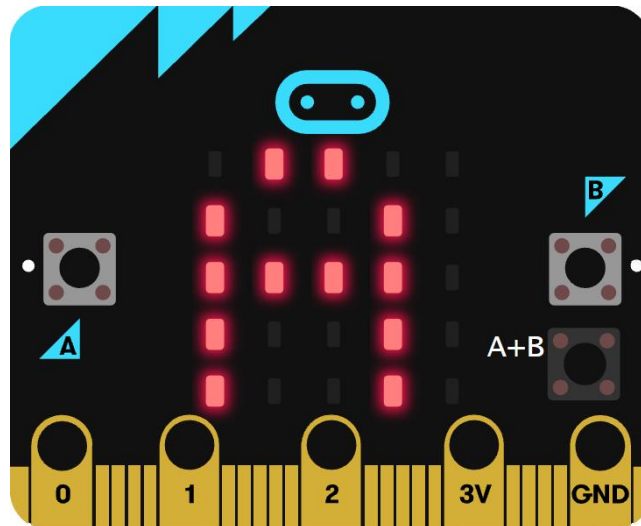
Step 8



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

Step 9

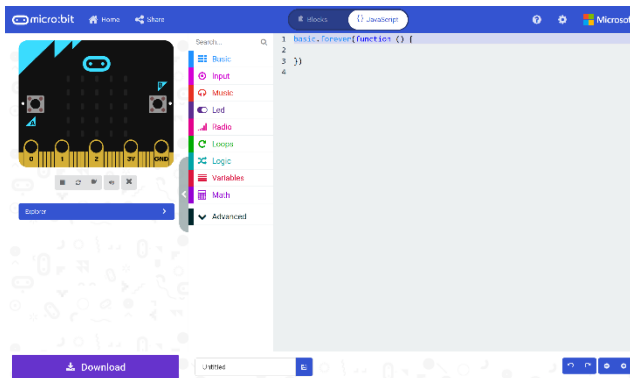
When running on the virtual **micro:bit** you can create a message on the LEDs by pressing the **A+B** button on the **micro:bit** then press the **A** button to cycle through the list of letters and numbers and then press **B** button to add the letter or number to the list to be displayed, you can keep doing this to add more letters and numbers then once the message is complete press **A+B** again to display the full message.



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.

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

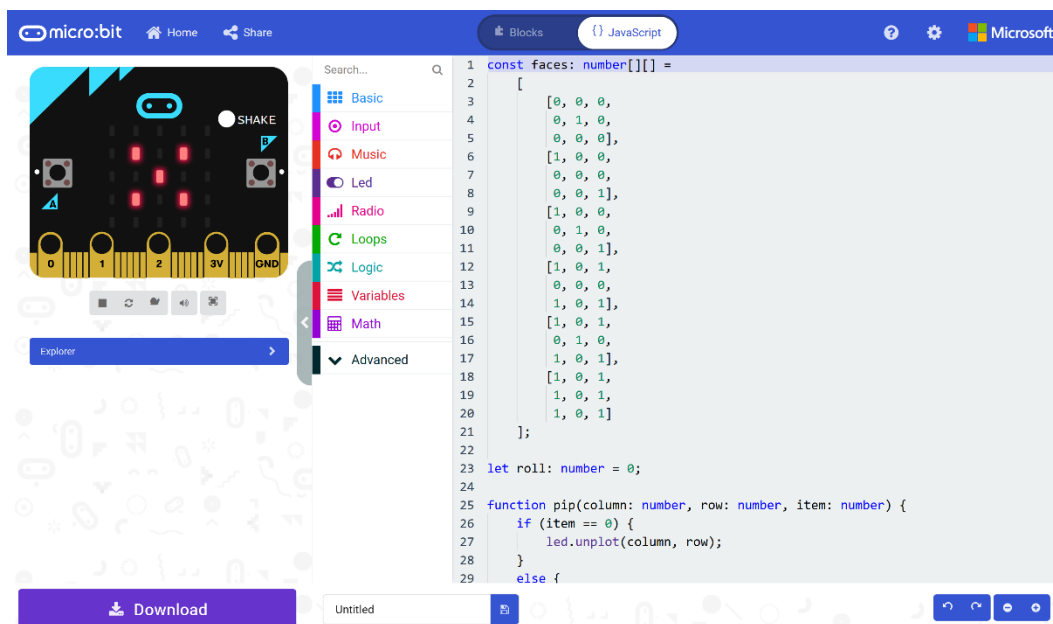
Finally, while still in the **JavaScript** tab, below the code entered in the previous step, you should enter the following code:

```
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:



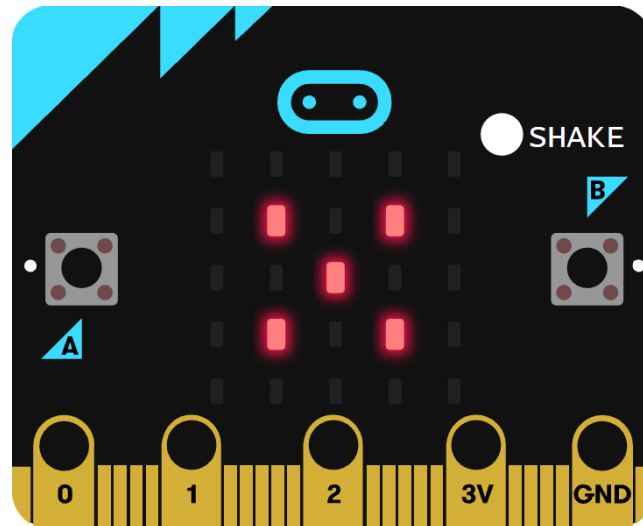
Step 8



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

Step 9

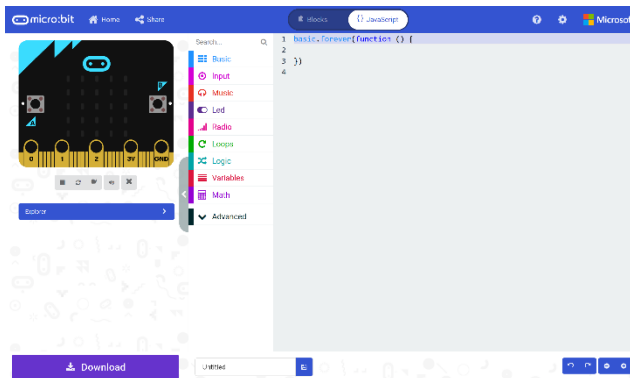
When running on the virtual **micro:bit** you can **shake** the **micro:bit** by pressing the **shake** button on the on-screen **micro:bit** this will randomly select a different face of a dice – it's actually just a "die" in this case as there's just the one. You can shake the real **micro:bit** in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

Compass Game

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:

```
let compass: boolean = true;  
let current: string = '';  
let pattern: string[] = [];
```

let helps create a value which can change and are known as variables which can contain a single value. Here there's a **boolean** which can store either **true** or **false** it is called **compass** and is set to **true**, then there is string called current and is set to an empty value or '', finally there's an array, or list of **string** values which is denoted by the use of two square brackets or **[]** and is set to an empty array which is also a pair of square brackets or **[]**.

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:

```
function heading(bearing: number): string {
  let result: string = "N";
  if (bearing < 45) {
    result = "N";
  } else if (bearing < 135) {
    result = "E";
  } else if (bearing < 225) {
    result = "S";
  } else if (bearing < 315) {
    result = "W";
  }
  return result;
}

function show(item: string) {
  basic.showString(item)
}
```

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 **heading** and this a **number** parameter called **bearing**. Inside there is a **let** which is what will be returned and is called **result** and is set to **"N"**. Following this is an **if** statement which uses the value of **bearing** to make a choice – when this value is less than **45** then the **result** will be set to **"N"**, otherwise when the bearing is less than **135** then the **result** will be set to **"E"** – there are other checks for other values and these all relate to the number of degrees around a circle that represent the points on a compass.
2. The second **function** is called **show** and the parameter used here is called **item** and it is a **string**, the **function** will use this value to use or call the **basic.showString** built in **function** to display text on the **micro:bit** using the LEDs on the front.

Step 4

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

```
input.onButtonPressed(Button.A, () => {
  pattern.push(current);
  compass = false;
  show("+");
  basic.pause(500);
  compass = true;
})

input.onButtonPressed(Button.B, () => {
  let display: string = '';
  compass = false;
  show(display);
  for (let i: number = 0; i < pattern.length; i++) {
    display += pattern[i];
  }
  show(display);
  basic.pause(1000);
  compass = true;
})

input.onButtonPressed(Button.AB, () => {
  pattern = [];
})
```

input are things that happen, also known as events, when you do something with the **micro:bit**.

3. The first **input** is when you press the **A** button on the **micro:bit** and when you do it will use **push** on the **pattern** list to add an item to the list or array. It then sets the **compass** value to **false** then it uses **show** to display a + on the **micro:bit** LEDs then is followed by a delay of half a second – this is what **basic.pause(500)** does, finally it sets the **compass** value back to **true**.
4. The second **input** is when you press the **B** button on the **micro:bit** – when you do it has a new value or variable called **display** which is set to an empty string or "" then it sets the **compass** value to **false** then it uses **show** with this empty string to clear the LEDs. There is a **for** loop, which allows something to be repeated – in this case it repeats from **0** to the number of items in the **pattern** array which is what **pattern.length** does, then inside this **for** loop it appends the item at the position in **pattern** array of the value of **i** which is set by the loop to the **display** value. After this it then uses **show** to output the value of **display** to the **micro:bit** using the LEDs. It then has a delay of one second – which is what **basic.pause(1000)** does, then finally it sets the **compass** value back to **true**.
5. The third **input** is when you press both the **A** and **B** buttons on the **micro:bit** at the same time **pattern** will be reset to an empty array – this is what **[]** is.

Step 5

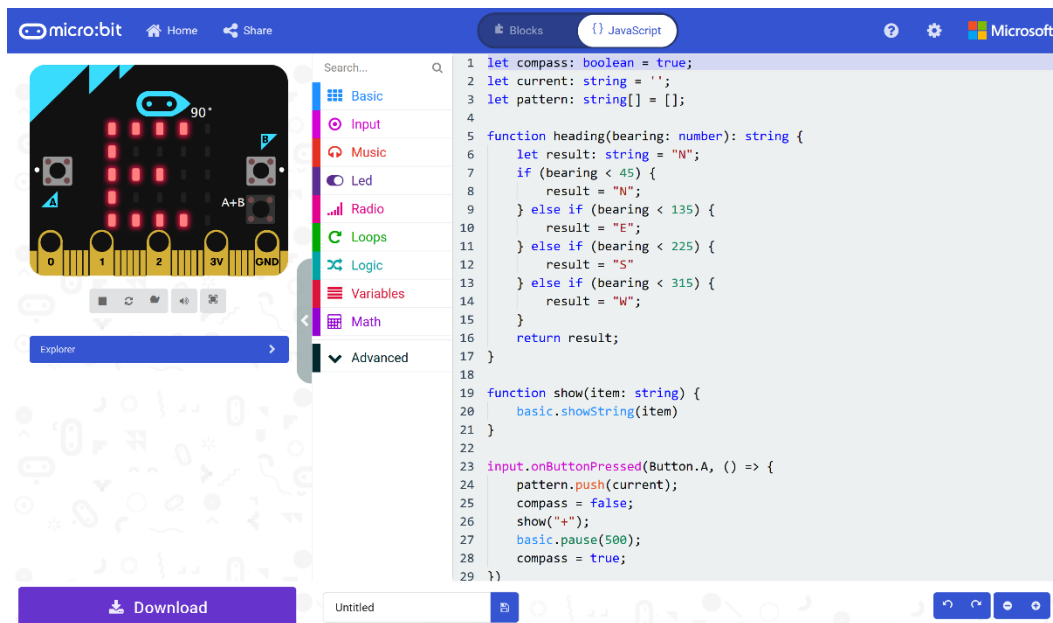
Finally, while still in the **JavaScript** tab, below the code entered in the previous step, you should enter the following code:

```
basic.forever(() => {  
    if (compass) {  
        current = heading(input.compassHeading());  
        show(current);  
    }  
})
```

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 check the value of **compass** and when this is **true** it will then set the **current** value using the function called **heading** where the parameter **bearing** has been provided with the current direction the **micro:bit** is pointing – that is what **input.compassHeading()** does. Then it will use or call the function named **show** – this will display value of **current** on the **micro:bit** using the LEDs.

Step 6

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



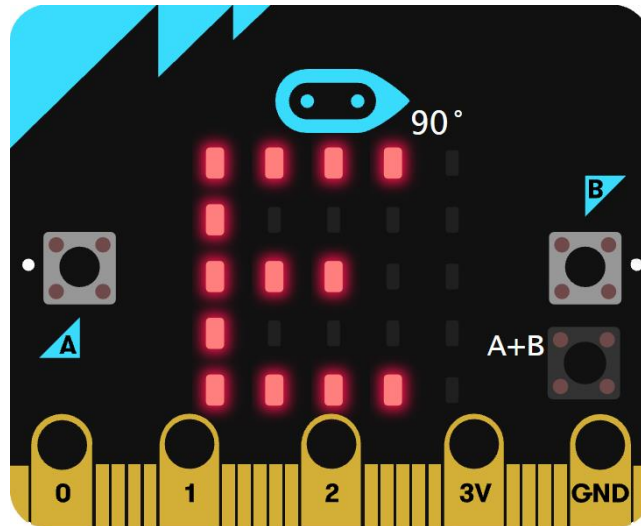
Step 7



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

Step 8

When running on the virtual **micro:bit** you can set the compass by moving around the **micro:bit** logo this will display the direction it is pointing in, for example **S** for South – you can turn a real **micro:bit** in your hand to point in a particular direction do the same thing, you'll need to calibrate it by waving it around in a figure-of-eight in the air until the **micro:bit** indicates this has completed. You can then press the **A** button to add a direction to a list which can be displayed at any time by pressing the **B** button which will display all the directions added – to clear the list at any time you just press the **A** and **B** buttons together.



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.