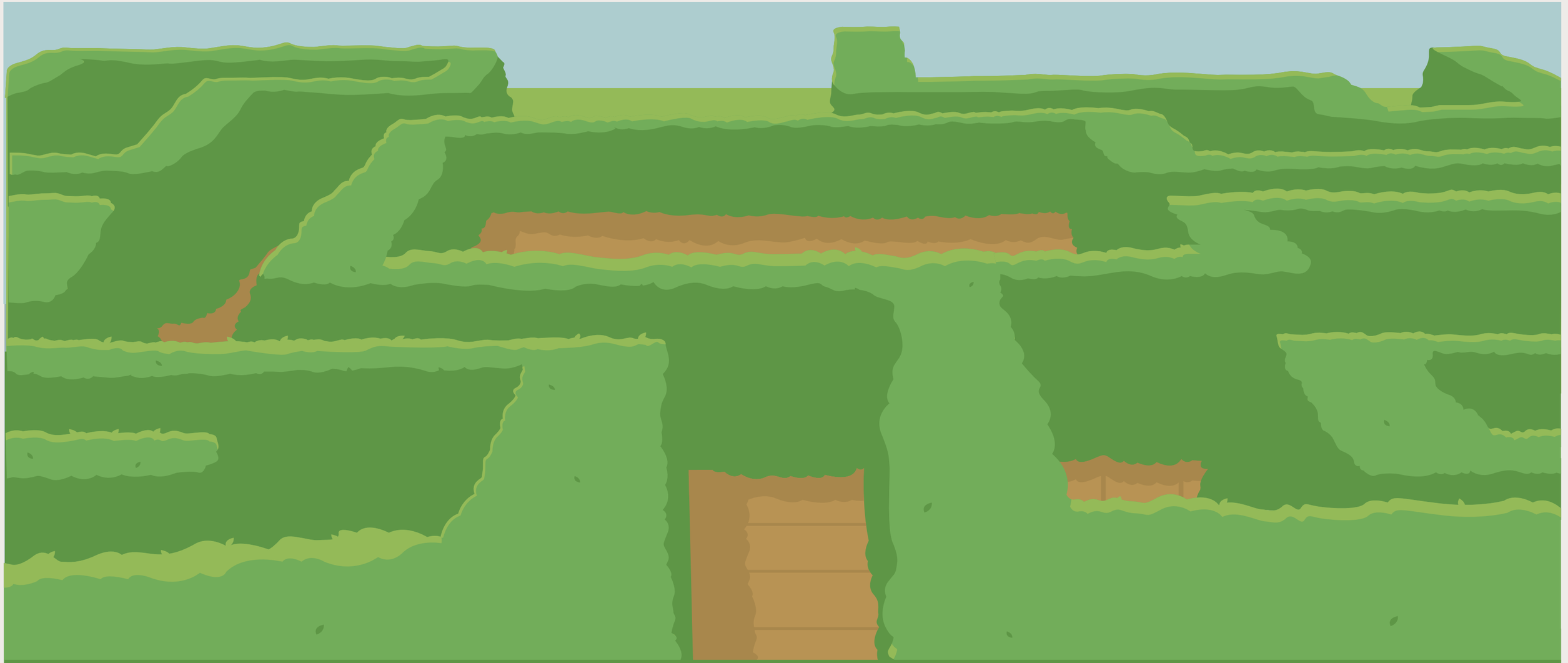


Get Lost In

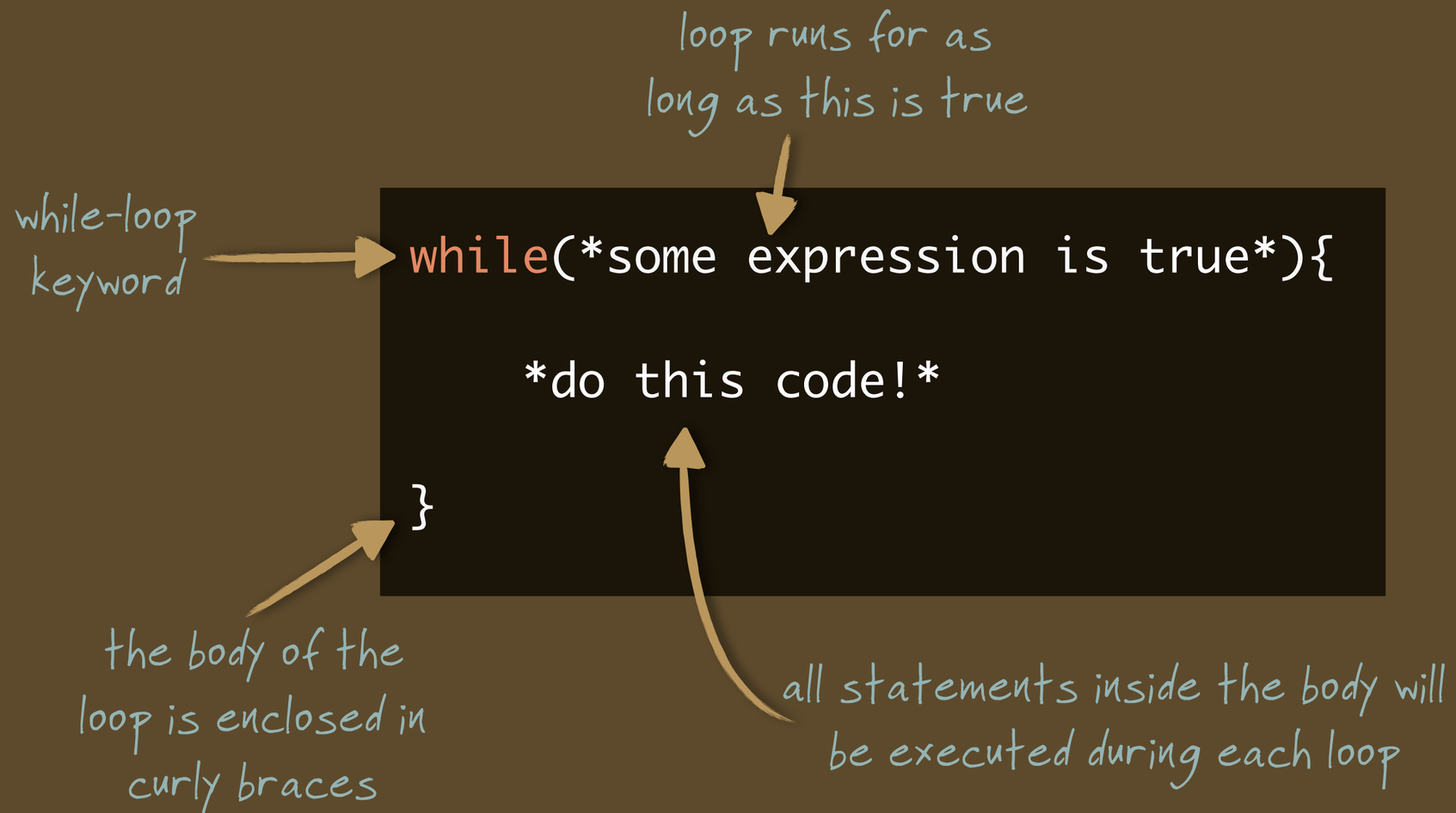
THE LABYRINTH OF LOOPS

LEVEL 1

THE LABYRINTH OF LOOPS

A BASIC WHILE-LOOP

The While-loop runs its code as long as its boolean expression is true



A BASIC WHILE-LOOP

This code runs forever...

```
while(true){

    *do this code!*

}
```

...and this code never runs!

```
while(false){

    *do this code!*

}
```


A BASIC WHILE-LOOP

Let's make a loop that prints the numbers 1-5 in ascending order.

```
let number = 1;
while (number <= 5) {

    console.log(number);
    number++;

}
```

Make sure to control your loop condition! If it never becomes false, you've got a dreaded infinite loop!

→ 1
→ 2
→ 3
→ 4
→ 5

UNDERSTANDING LOOPS

Loops allow us to have code executed repeatedly without extra typing

We want to execute the following code for every running train:

```
console.log("Train #" + <number> + " is running.");
```

Since we know variables can be changed, let's make one that will count our trains:

```
let trainNumber = 1;
```

Initially, we set the number to 1 for the first train.

Then, we need a way to make our train counter increase on every repetition:

```
trainNumber++;
```

Incrementing trainNumber will move forward to the next train.

So now, the following code needs to "loop" until all running trains have been listed:

Loop back

```
console.log("Train #" + trainNumber + " is running.");
trainNumber++;
```

We might also say that this code needs to run as long as: `trainNumber <= 8`

TRACING THROUGH THE LOOP WE NEED

Before the loop, start by initializing

```
let trainNumber = 1;
```

Loop only if

```
trainNumber <= 8
```

Perform this in each loop

```
console.log("Train #" + trainNumber + " is running.");
```

At the end of each loop

```
trainNumber++;
```

Advances the train number

```
let trainNumber = 1;
while( trainNumber <= 8){
  console.log("Train #" + trainNumber + " is running.");
  trainNumber++;
}
```

TRACING THROUGH THE LOOP WE NEED

```
let trainNumber = 1;
while( trainNumber <= 8){
  console.log("Train #" + trainNumber + " is running.");
  trainNumber++;
}
```

trainNumber	trainNumber <= 8 ?	Loop Output
1	TRUE	Train #1 is running.
2	TRUE	Train #2 is running.
3	TRUE	Train #3 is running.
4	TRUE	Train #4 is running.
5	TRUE	Train #5 is running.
6	TRUE	Train #6 is running.
7	TRUE	Train #7 is running.
8	TRUE	Train #8 is running.
9	FALSE	STOP!

BUILDING OUR WHILE-LOOP IN TRAINS.JS

Using variables instead of values to control our train-loop

By using variables, our loop is flexible. We could quickly change how many operational trains there are, and our loop would still print them all.

trains.js

```
let trainsOperational = 8;
let trainNumber = 1;

while (  trainNumber <= trainsOperational  ){

      console.log("Train #" + trainNumber + " is running.");
      trainNumber++;

}
```

THE FOR-LOOP

A different way of producing the same looping behavior

We typically use this spot for initialization.

This is usually the ++ or -- statement that controls the loop.

```
for (    *start with this*     ;    *loop if this expression is true*     ; *do this after each loop* ) {

    *in each loop, do this code!*

}
```

This is just like the condition that the While-loop checks before looping

THE FOR-LOOP

A different way of producing the same looping behavior

```
for ( let trainNumber = 1;  trainNumber <= trainsOperational;      trainNumber++      ) {

     console.log("Train #" + trainNumber + " is running.");

}
```


THE FOR-LOOP

Visualizing the flow of our for-loop

The loop counter is first initialized as part of the loop itself.

The loop checks an expression to see if another loop should execute.

Before another loop, the middle expression is checked again.

```
for ( let trainNumber = 1; trainNumber <= trainsOperational; trainNumber++ ) {


    console.log("Train #" + trainNumber + " is running.");


}
```

If the loop should run, the body code will execute.

After body code is executed, the final expression is executed before moving on. This statement usually affects whether the loop will run again.

FOR-LOOP IN ACTION

Printing numbers in descending order

Semicolons separate expressions.

This loop runs as long as number is still greater than zero.

```
for(let number = 5; number > 0; number--) {

    console.log(number);

}
```

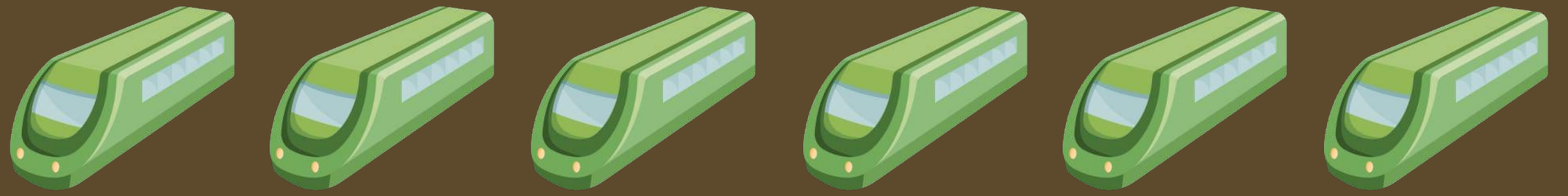
Decreases number's value by one at the end of the loop.

Value of number	number > 0?	Loop Output
5	TRUE	5
4	TRUE	4
3	TRUE	3
2	TRUE	2
1	TRUE	1
0	FALSE	STOP!

IDENTIFYING THE NON-OPERATIONAL TRAINS

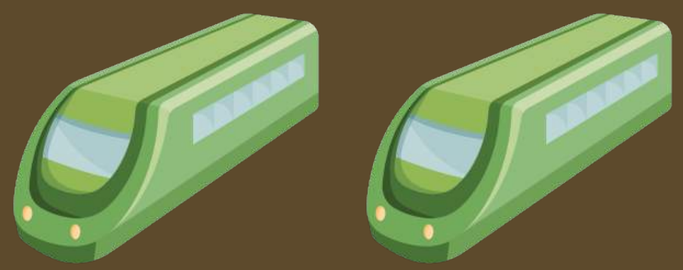
We’ve printed the running trains; now we turn to the stopped trains.

1 2 3 4 5 6



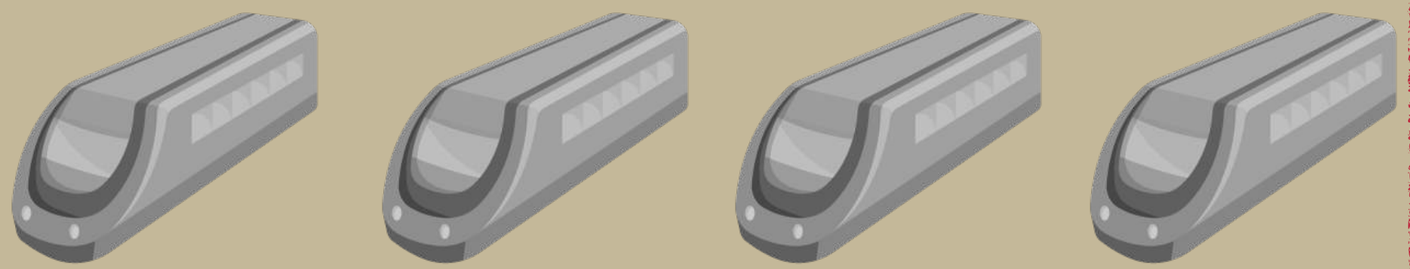
✓ ✓ ✓ ✓ ✓ ✓

7 8



✓ ✓

9 10 11 12



✗ ✗ ✗ ✗

WHICH TRAINS ARE NOT RUNNING?

Let's use the For-loop to list the non-operational trains

```
let trainsOperational = 8;
```

```
let totalTrains = 12;
```

Set stoppedTrain to the train number immediately after our last running train, i.e. #9

The loop should run only until we have reached the maximum # of trains

```
for(let stoppedTrain = trainsOperational + 1; stoppedTrain <= totalTrains; stoppedTrain++){

    console.log("Train #" + stoppedTrain + " is not operational.");

}
```

Wahoo, stopped trains only!

```
Train #9 is not operational.
Train #10 is not operational.
Train #11 is not operational.
Train #12 is not operational.
```

ADDING OUR FOR-LOOP TO TRAINS.JS

trains.js

```
let totalTrains = 12;
let trainsOperational = 8;


let trainNumber = 1;
while(trainNumber <= trainsOperational){
        console.log("Train #" + trainNumber + " is running.");
        trainNumber++;
}

for(let stoppedTrain = trainsOperational + 1; stoppedTrain <= totalTrains; stoppedTrain++){
        console.log("Train #" + stoppedTrain + " is not operational.");
}
```


RUNNING OUR CURRENT SOLUTION

Elements Resources Network Sources Timeline Profiles Audits Console

```
> Train #1 is running.
  Train #2 is running.
  Train #3 is running.
  Train #4 is running.
  Train #5 is running.
  Train #6 is running.
  Train #7 is running.
  Train #8 is running.
  Train #9 is not operational.
  Train #10 is not operational.
  Train #11 is not operational.
  Train #12 is not operational.
```

<top frame> <page context> All Errors