while-loops

Learn how while-loops differ from for-loops. Learn when to use each loop and the advantages of each.

While-loops#

While-loops are another tool in our toolbelt. They're used less frequently than for-loops.

Let's see how to do the same thing we did in the previous lesson with a while-loop. We want to log the numbers 0 - 9 to the console.

Structure#

While-loops have a different structure.

```
while(/*condition*/) {
    // body
}
```

They take in only one piece of information, the condition. We have to do the other parts ourselves. To make this while loop equivalent to our for-loop from the previous lesson:

```
1  let i = 0;
2
3  while(i < 10) {
4      console.log(i);
5      i++;
6  }
```

for vs. while#

for-loop advantages#

A for-loop is completely self-contained. Everything the loop needs to know is inside the parentheses. A while loop feels a little looser. We have to put the initial variable, `i`, and the operation to be performed, `i++`, in different places. This is why for-loops are generally preferred.

while-loop advantages#

We sometimes have to repeat things an unknown number of times. This is when while-loops shine. We only give them a condition and when the condition is no longer met, the loop stops. In this way, they're more flexible than for-loops.

while-loop disadvantages#

Infinite loops are much easier to create when writing while-loops. It's easy to forget to increment a variable that we're using in our condition.

It's frustrating when this happens, as you'll usually have to kill whatever engine you're using to make the infinite loop stop. In a browser, this can mean having to close your entire browser.

Back

for-loops, continued

Next

for-in loops

Mark as Completed

Report an Issue