if-statements, continued

Learn what truthy and falsey mean in JavaScript. Learn how to make if-statements more powerful.

“Truthy” and “Falsey”#

If-statements don’t work only with booleans. We can use any variable type inside the if-statement parentheses.

If we don’t use `true` or `false` variables, JavaScript will forcibly coerce whatever we put inside the parentheses to `true` or `false`. It’ll then decide whether to run the code or not.

A value that coerces to `true` is referred to as “truthy”. One that coerces to `false` is “falsey”.

So how do we determine what a value will coerce to? It’s actually pretty simple. The following values are “falsey” and will coerce to `false`, meaning the code in the if-statement won’t run.

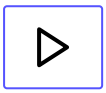
- `false`
- `null`
- `undefined`
- `''` or `""` (empty, 0-length string)
- `0` (the number zero)
- `NaN`

All other values are truthy. This means that all numbers except 0 and `NaN` and all strings that are not empty are truthy.

Try setting `itemToTest` equal to different values in the code block below. Test them out.

```
1  // Change 'true' in the next line to whatever you like.
2  let itemToTest = true;
3
```

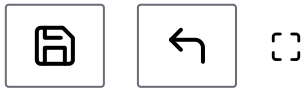
```
4  if(itemToTest) {
5      console.log(itemToTest + ' is truthy!');
6  } else {
7      console.log(itemToTest + ' is falsey.');
8  }
```




No Brackets#

We can write an if-statement without the curly brackets `{}`. Let's say we want to run a *single line* of code if some condition is true. We could write it like this.

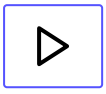
```
let trueFlag = true;
if(trueFlag) console.log('This will print!');
```

We can do the same thing with the else-block.

```
let falseFlag = false;

if(falseFlag) console.log('This will not print.');
else console.log('This will print!');
```




This happens because **the if-statement technically only runs the next item it sees**. If we write code in a block using `{}`, that entire block is the item.

If we write code without brackets, the statement will run only the next line of code it sees.

This means we can chain else-blocks.

Chaining else-blocks#

We can write complex conditional logic using the rules we've discussed so far. Because the else-block only runs the next item it sees after the condition, the

following three code blocks are equivalent. All we're doing as we go down is removing brackets.

```
let flag1 = false;
let flag2 = false;
let flag3 = true;

if (flag1) {
    console.log('This will not print.');
} else {
    if (flag2) {
        console.log('This will not print either.');
    } else {
        if (flag3) {
            console.log('This will print!');
        }
    }
}
```

```
let flag1 = false;
let flag2 = false;
let flag3 = true;

if (flag1) {
    console.log('This will not print.');
} else if (flag2) {
    console.log('This will not print either.');
} else if (flag3) {
    console.log('This will print!');
}
```

```
let flag1 = false;
let flag2 = false;
let flag3 = true;

if (flag1) console.log('This will not print.');
else if (flag2) console.log('This will not print either.');
else if (flag3) console.log('This will print!');
```

The first code block above is difficult to read and understand. The second and third are much clearer.

Using Single-Line Statements#

If we ever need to convert a single-line if-statement to a multiline block, we'll need to add in the brackets. It's easier to write if-statements with the brackets and keep the brackets there, so that's what I generally prefer.

It doesn't matter too much and it's ultimately your choice.

We'll go deeper into if-statements in the next lesson.

Quiz#

Feel free to test your understanding.

1 What will the following code print?

```
if(5 - 3 - 2) {
        console.log('A');
} else {
        console.log('B');
}
```

- A) A
- B) B
- C) A

  B

Submit Answer

Question 1 of 4
0 attempted

Reset Quiz

Back

Introduction to if-statements

Next

Logical Operators: !, ||, &&

Completed

Report an Issue