ToiletPaper #154

jambit
WHERE INNOVATION WORKS

Forgotten JavaScript feature: labels

Author: Santo Pfingsten / Senior Software Architect / Office Leipzig

✕ Problem

We may all be familiar with this situation: You have nested loops (or conditions) and need to break out deeply. Common approaches are e.g.:

- After the inner loop, exit the outer loop by another condition.
- Refactoring the code to use an early-return instead of using `break`.

✓ Solution

In JavaScript, you can solve this differently with an old, largely forgotten feature. Labels make it possible to give loops (and conditions!) a name and then address them with the statements `break` and `continue`.

Without labels

```
for (const row of rows) {
  let breakOuter = false;
  let continueOuter = false;
  for (const cell of row) {
    if (cell === 'break') {
      breakOuter = true;
      break;
    }
    if (cell === 'continue') {
      continueOuter = true;
      break;
    }
  }
  if (breakOuter) {
    break;
  }
  if (continueOuter) {
    continue;
  }
  console.log('-');
}
```

For loop with labels

```
outer: for (const row of rows) {
  for (const cell of row) {
    switch (cell) {
      case 'break':
        break outer;
      case 'continue':
        continue outer;
    }
  }
  console.log('-');
}
```

Conveniently, this also works with conditions:

```
const [a, b, c] = [1, 3, 2];

found: if (a < b) {
  if (c < b) {
    break found;
  }
  console.log('never gets here');
}
console.log('done');
```

➔ Example

When should you use labels?

- JS labels are not the same as GoTo statements. You do not jump to a label but bind a `break` or `continue` statement to a condition or loop. Wild jumping like in C/C++ is not possible.
- As is often the case, it is important to use the right tool at the right time. Sometimes when writing code, there may be better solutions than using labels. But if not, labels are certainly worth considering.

+ Further Aspects

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/label