Nested if-else

Let's learn how to implement nested if-else expressions.

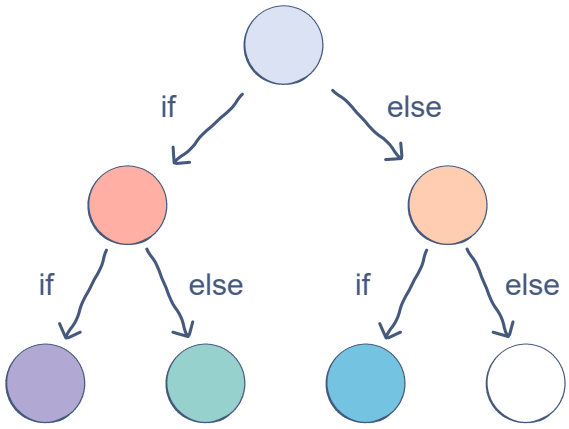
WE'LL COVER THE FOLLOWING

- Purpose of Nesting
- Example

Purpose of Nesting

Reason supports the use of nested `if-else` expressions. This is useful because it allows us to create a decision tree branching towards different outcomes.

Think of the `if-else` expression as a chance to go two ways. If we create new `if-else` conditionals within these expressions, we are basically allowing the program to go into deeper paths.



Example

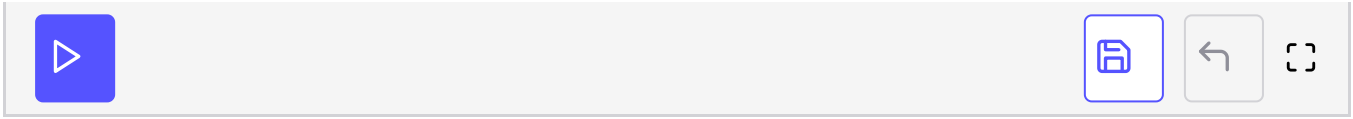
Let’s take a look at nesting in `if-else` expressions.

```
let arr = [| 20, 50, 10, 90, 70, 40|];
Js.log(arr);

if (Array.length(arr) < 10){
  if (arr[0] mod 2 == 1) {
```

```
    arr[0] = 1000;
  }
  else{

    arr[0] = 2000;
  };
}
else{
  arr[0] = 3000;
};
Js.log(arr);
```



The code above changes the value of `arr`'s first element based on the conditions we specify. It goes into the nested `else` expression since the length of `arr` is less than `10` and `arr[0] mod 2` is not equal to `1`.

We're done with the basics of the `if-else` expression.

Next, we'll discuss the `switch` conditional expression.