If, Else If and Else Statements

You will now learn how to branch off based on conditional statements. This lesson will teach you about if, else if and else statements.

The time has come, when we can fit the puzzle pieces together. You have learned about different primitive and composite types. You know how to check these types. You know the basic functions and arithmetic (addition, subtraction, multiplication etc.) and logical (true or false) operations in JavaScript.

There are only two things missing.

So far, we can write a sequence of instructions that perform a calculation. In software development, there are three types of control structures:

1. sequence: writing instructions one after the other
2. selection: either execute one set of instructions, or another
3. iteration: execute a set of instructions a finite or infinite number of times

As you already know how to sequence instructions one after the other, it is time to learn about selection and iteration. Let's start with selection:

```
if ( condition ) {
    instruction1;
    instruction2;
    //...
}
```

If statement

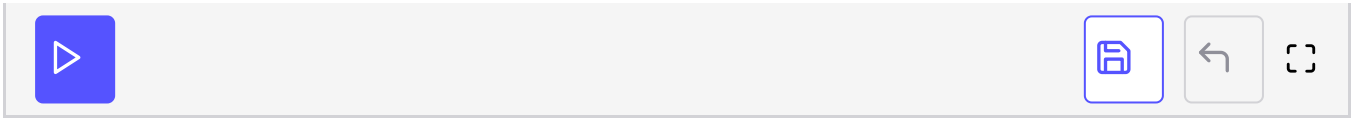
The instructions inside the if-branch are only executed if `condition` is truthy.

Notice the space between the `if` and the opening parentheses. This space is there to indicate that `if` is a *keyword* and not a function. Although this space is not mandatory, I highly recommend it for readability reasons.

```
function printNumber( x )
{

    if ( typeof x === 'number' )
    {
        console.log( x + ' is a number.' );
    }
}

printNumber(5);
console.log(printNumber());
printNumber(3);
```



If you run the code above, the first value is the console log. The second value is the return value of the `printNumber()` function. Remember, if the return value of a function is not specified, it returns `undefined`.

If we call the `printNumber` function with a non-numeric value, it does not print the console log, because the instructions inside the `if` branch are only executed, whenever the condition inside the `if` statement is truthy.

```
> printNumber( '' )
undefined
```

Adding another if statement

It is a bit awkward that we don't print anything in case the input is not a number. You already know everything to write a program that does this. The thought process is the following:

- if `x` is a number, print it
- if `x` is not a number, print the message `"The input has to be a number."`

```
function printNumber( x ) {
    if ( typeof x === 'number' ) {
        console.log( x + ' is a number.' );
    }
    if ( typeof x !== 'number' ) {
        console.log( 'The input has to be a number.' );
    }
}

printNumber(3);
printNumber("blah");
```

Else Statement

The second biggest problem with this solution is that we are writing too much for no reason.

The main problem with this solution is called *lack of abstraction*. Imagine that one day somebody comes and realizes that the condition `typeof x === 'number'` has to be modified. If you forget changing the second condition, you create an inconsistency in your code. We prefer thinking less when we don't have to. Therefore, I recommend another solution, where we only have one condition:

```
function printNumber( x ) {
    if ( typeof x === 'number' ) {
        console.log( x + ' is a number.' );
    } else {
        console.log( 'The input has to be a number.' );
    }
}

printNumber(3);
printNumber("blah");
```

Notice the `else` keyword. The `else` branch is only executed if the original condition is not true.

We can cascade this approach further, because after the `else`, you can have another condition:

```
function logLampColor( state ) {
    if ( state === 1 ) {
        console.log( 'Red' );
    } else if ( state === 2 ) {
        console.log( 'Yellow' );
    } else if ( state === 3 ) {
        console.log( 'Green' );
    } else {
        console.log( 'Wrong lamp state' );
    }
}
```

So, if the first condition fails, it will test the next `else if` condition, if that fails

too, then the next one and so on until all the `else if` fails and it will finally execute the `else` statement.

The beauty of the above code is that you can read it as if it was plain English. If the state is 1, then log Red. Otherwise if the state is 2, then log Yellow. And so on.