The ES6 way

default values and using a variable number of arguments in functions

ES6 supports default values. Whenever an argument is not given, the default value is substituted. The syntax is quite compact:

```
function addCalendarEntry(
    event,
    date = new Date().getTime(),
    len = 60,
    timeout = 1000 ) {

    return len;
}
var add=addCalendarEntry( 'meeting' );
console.log(add); //outputs the default value set earlier
```

Suppose function `f` is given with two arguments, `a` and `b`.

```
function f( a = a0, b = b0 ) { ... }
```

When `a` and `b` are not supplied, the above function is equivalent to

```
function f() {
    let a = a0;
    let b = b0;
    ...
}
```

Default arguments can have arbitrary types and values.

All considerations for let declarations including the temporal dead zone hold. `a0` and `b0` can be any JavaScript expressions, in fact, `b0` may even be a function of `a`. However, `a0` cannot be a function of `b`, as `b` is declared later.

> Use default arguments at the end of the argument list as optional arguments. Document their default values.

The `arguments` array is not affected

In earlier versions of JavaScript, we often used the `arguments` array to handle a variable number of arguments:

```
function printArgs() {
    console.log( arguments );
}

printArgs( 'first', 'second' );
```

Bear in mind that the `arguments` array is not affected by the default parameter values in any way.

```
function printArgs( first = 'No arguments' ) {
    console.log( arguments );
}

printArgs();
```

To get a better understanding, see Exercise 3 in the next lesson.