Hacks in ES5

optional function arguments and their replacement, limitations of ES5 shortcuts

In some cases, function arguments are optional. For instance, let's check the following code:

```
function addCalendarEntry( event, date, len, timeout ) {
    date = typeof date === 'undefined' ? new Date().getTime() : date;
    len = typeof len === 'undefined' ? 60 : len;
    timeout = typeof timeout === 'undefined' ? 1000 : timeout;

    // ...
}
addCalendarEntry( 'meeting' );
```

Three arguments of `addCalendarEntry` are optional.

A popular shorthand for optional parameters in ES5 uses the `||` (logical or) operator. You can make use of the shortcuts for logical operations.

```
function addCalendarEntry( event, date, len, timeout ) {
    date = date || new Date().getTime();
    len = len || 60;
    timeout = timeout || 1000;
    // ...
}
addCalendarEntry( 'meeting' );
```

The value `value || defaultValue` is `value`, whenever `value` is true. If the first operand of a `||` expression is true, the second operand is not even evaluated. This phenomenon is called a logical shortcut.

When `value` is false, `value || defaultValue` becomes `defaultValue`.

While this approach looks nice on paper, shortcuts are sometimes not flexible enough. All false values are treated in the same way, including `0`, `''`, `false`. Sometimes, we may want to treat a `0` differently than an `undefined` indicating the absence of a value.

Now, let’s see how we handle this using ES6.