# Miscellaneous New APIs

## Overview

In this lab you'll get a chance to explore various new APIs available in modern JavaScript to process numbers, manipulate strings, and perform pattern matching with regular expressions.

## Source folders

* C:/JsDeepDive/Labs/Student/03-APIs
* C:/JsDeepDive/Labs/Solutions/03-APIs

## Roadmap

There are 3 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Familiarization with the solution application
2. Implementing the student application
3. (If Time Permits) Additional suggestions

## Exercise 1: Familiarization with the solution application

Open a Command Prompt window, go to the C:/JsDeepDive folder, and run the following command to start the Babel transpiler:

npx gulp

Now go to the **solution** folder for this lab and run the web page. The page asks the user to enter credit card details, and performs various validation and processing tasks. Note the following points:

* The *amount* must be a number. If the user enters a valid number, the amount in pennies is displayed next to the text box.
* The *name* can be anything, but it must not be empty. If the user enters any name, it's displayed next to the text box.
* The *card number* must be a 16-digit number, starting with 4 or 5. If the string includes any hyphens, spaces, or periods, these are ignored. If the user enters a valid card number, it's displayed in XXXX-XXXX-XXXX-XXXX format next to the text box.
* The *expiry date* must be a 4-digit number in mmyy format. The month must be between 01 and 12 inclusive, and the year must be between 18 and 23 inclusive. If the user enters a valid expiry date, the month and year are displayed separately next to the text box.
* The *CVV* must be a 3-digit number. If the user enters a valid CVV number, it's displayed next to the text box.

## Exercise 2: Implementing the student application

Switch to the **student** folder and add code to es6scripts/script.js to implement the logic you just saw in the solution. We suggest you take things one function at a time, to make sure you trap all the possible error conditions.

Note we've provided some helper functions to make it easier to interact with HTML elements in your code:

* getValue(selector)
* setHtml(selector, html)
* setHtmlAll(selector, html)

## Exercise 3 (If time permits): Additional suggestions

Take another look through the chapter and see if there are any other APIs you fancy having a go at. See how you can add these features into your application.