Regular Expressions

This lesson is a brief introduction to Regular Expressions and their importance in JavaScript.

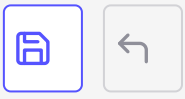
The previous validations were quite primitive: many strings containing a `@` character are not valid email addresses. To perform more advanced checks, you can use a powerful tool: *regular expressions*.

A regular expression defines a *pattern* to which strings are compared, searching for matches. Many programming languages support them. A powerful addition to a programmer's toolbelt, they can nonetheless take quite a bit of time to be comfortable with. What follows is just an introduction to the vast domain of regular expression.

Let's get started by trying to create a regular expression checking for the presence of an `@` character inside a string. Here's the associated JavaScript code.

JavaScript

```
const regex = /@/; // String must contain @
console.log(regex.test(""));  // false
console.log(regex.test("@")); // true
console.log(regex.test("sophie&mail.fr")); // false
console.log(regex.test("sophie@mail.fr")); // true
```



A JavaScript regular expression is defined by placing its pattern between a pair of `/` characters. It's an object whose `test()` method checks matches between the pattern and the string passed as a parameter. If a match is detected, this method returns `true`, and `false` otherwise.

The following table presents some of the numerous possibilities offered by regular expressions.

Pattern	Matches if	Match	No Match
`abc`	String contains `"abc"`	`"abc"`, `"abcdef"`, `"123abc456"`	`"abdc"`, `"1bca"`, `"adbc"`, `"ABC"`
`[abc]`	String contains either `"a"`, `"b"` or `"c"`	`"abc"`, `"daef"`, `"bbb"`, `"12c34"`	`"def"`, `"xyz"`, `"123456"`, `"BBB"`
`[a-z]`	String contains a lowercase letter	`"abc"`, `"12f43"`, `"_r_"`	`"123"`, `"ABC"`, `"_-_"`
`[0-9]` or `\d`	String contains a digit	`"123"`, `"ab4c"`, `"a56"`	`"abc"`
`a.c`	String contains `"a"`, followed by any character, followed by `"c"`	`"abc"`, `"acc"`, `"12a.c34"`	`"ac"`, `"abbc"`, `"ABC"`
`a\.c`	String contains `"a.c"`	`"a.c"`, `"a.cdef"`, `"12a.c34"`	`"ac"`, `"abc"`
`a.+c`	String contains `"a"`, followed by at least one character, followed by `"c"`	`"abc"`, `"abbc"`, `"12a$ùc34"`	`"ac"`, `"bbc"`
`a.*c`	String contains `"a"`, followed by zero or more characters,	`"abc"`, `"abbc"`, `"ac"`	`"ABC"`, `"bbc"`

	followed by `"c"`		

Observing these examples leads us to the following rules:

- Brackets `[]` define a character interval. Any string with at least a character in this interval will match the pattern.
- The `[a-z]` and `[A-Z]` patterns are used to search for the presence of any letter, respectively lowercase and uppercase.
- The `[0-9]` and `\d` patterns are essentially identical and match a digit in a string.
- The `.` character replaces any one character.
- The `\` (backslash) character indicates that the following character should be searched as-if. . For example, `\.` is used to match the . character itself.
- The `+` character matches one or several instances of the preceding expression.
- The `*` character matches zero, one, or several instances of the preceding expression.

T> The site https://regex101.com is useful to understand, test and debug regular expressions.

Let's get back to our example and check the email address field. Here's a possible regular expression (among many others) to test it against:
`/.+@.+\..+/`

Q> Before reading further, can you decode this pattern to understand what conditions a string must respect to match it?

OK, here is the answer. This pattern matches a string that:

- Starts with one or several characters (`.+`)
- Next, contains the `@` character (`@`)
- Next, contains one or several characters (`.+`)
- Next, contains the `.` character (`\.`)
- Finishes with one or several characters (`.+`)

In other words, any string of the form `xxx@yyy.zzz` will match this pattern. This is not the end-all, be-all way to validate an email address, but it's a start.

Check out how to put this solution into practice.

Output

JavaScript

HTML

```
// Check email validity when field loses focus
document.getElementById("emailAddress").addEventListener("blur", e => {
  // Match a string of the form xxx@yyy.zzz
  const emailRegex = /.+@.+\..+/;
  let validityMessage = "";
  if (!emailRegex.test(e.target.value)) {
    validityMessage = "Invalid address";
  }
  document.getElementById("emailHelp").textContent = validityMessage;
});
```