Better Unicode Support

handling Unicode strings in ES5 and its alternative in ES6: using the for-of loop and spread operator to process characters

We are going to a lower level in this section.

ES5 handles string operations in two-byte chunks. As a result, handling Unicode strings with a mixture of two and four byte long codes often gets confusing. Characters of Unicode strings were often not printable on their own, and in general, any string operation required extra care.

For instance, if you recall the first exercise of the last section, the reference solution would have failed if a title had started with a three or four-byte long character.

We still calculate the length of a string by dividing the number of bytes allocated by a string by two. However, there are some updates in ES6 that make String handling more user-friendly.

- `codePointAt(n)` returns the code of the character at the `n`th position regardless of whether it is a two byte or a four-byte long character. If `n` points at the second half of a four-byte Unicode character, only the code of the second half is returned

In ES6, it is possible to define four byte long Unicode characters with their code:

```
console.log('\u{1f600}')                          // becomes an emoji
console.log('\u{1f600}'.length)                   // is 2
console.log('\u{1f600}'.charCodeAt(0))            // is 55357
console.log('\u{1f600}'.codePointAt(0))           // is 128512
console.log('\u{1f600}'.charCodeAt(1))            // is 56832
console.log('\u{1f600}'.codePointAt(1))           // is 56832
console.log('\u{1f600}'.startsWith('\u{1f600}'[0])) // is true
```

Note that the `startsWith`, `endsWith`, `includes` methods interpret the result in two-byte chunks.

The for-of loop interprets three and four byte long characters as one unit, scoring another convenience point for ES6. Let's execute the following:

```
let str = '\u{1f600}\u{00fa}é';

for (const ch of str) {
    console.log( ch );
}
```

What is printed to the console? Notice that

- The `for-of` loop prints three characters, an emoji, `ú`, and `é`.
- `[...str]` spreads `str` character by character
- Even though `[...str]` has three elements, the length of the `str` string is `4`. This is because the length of `[...str][0]` is `2`.

> Use the `for-of` loop or the spread operator to process characters of a string regardless of their length in bytes.

This section contains a minimalistic summary of Unicode features of ES6 from a practical point of view. If you have to handle four byte-long characters on a regular basis, you should do your research. Make sure you know the `normalize` method for Unicode normalization. Make sure you know that there is a new `u` flag to influence whether to consider Unicode characters when testing regular expressions.

Normalization and Unicode support for regular expressions are outside the scope of this course.

In the next lesson, we'll talk about using template literals to work with string templates.