Regular Expression: Character Classes

Summary: in this tutorial, you'll learn about character classes in regular expressions to match a set of characters including digits, whitespace, and word characters.

Introduction to the character classes

A character class allows you to match any symbol from a character set. Note that a character class is also known as a character set.

Suppose that you have a phone number like this:

```
+1-(408)-555-0105
```

And you want to turn it into a plain number:

```
14085550105
```

Character classes in regular expressions can help you to do this.

Let's explore the digit character class first. The digit character class is denoted by `\d` which matches any single digit:

```
\d
```

The following example uses the `\d` to match the first number in the phone number:

```
let phone = '+1-(408)-555-0105';
let re = /\d/;

console.log(phone.match(re));
```

Output:

```
['1', index: 1, input: '+1-(408)-555-0105', groups: undefined]
```

When you add the global flag (`g`), the regular expression will search for all numbers, not the first one:

```
let phone = '+1-(408)-555-0105';
let re = /\d/g;

console.log(phone.match(re));
```

Output:

```
["1", "4", "0", "8", "5", "5", "5", "0", "1", "0", "5"]
```

Now, you can turn the phone number into a plain number as follows:

- Use the `match()` method to return an array containing numbers.
- Use the `join()` method to concatenate elements of the array into a string.

For example:

```
let phone = '+1-(408)-555-0105';
let re = /\d/g;

let numbers = phone.match(re);
let phoneNo = numbers.join('');

console.log(phoneNo);
```

Output:

```
14085550105
```

To make it more concise, you can chain the `match()` and `join()` methods like this:

```
console.log('+1-(408)-555-0105'.match(/\d/g).join(''));
```

Besides the digit character class (`\d`), regular expressions support other character classes.

The most commonly used character classes are:

- `\d` – match a single digit or a character from 0 to 9.
- `\s` – match a single whitespace symbol such as a space, a tab (`\t`), a newline (`\n`).
- `\w` – `w` stands for word character. It matches the ASCII character `[A-Za-z0-9_]` including Latin alphabets, digits, and the underscore (`_`)

In practice, you often combine the character classes to form a more powerful match.

For example `\w\d` matches any word followed by a digit like `O2` :

```
let str = 'O2 is oxygen';
let re = /\w\d/g

console.log(str.match(re));
```

Output:

```
['O2']
```

A pattern may contain both regular characters and character classes. For example, the `ES\d` regular expression matches `ES` followed by a digit like `ES6` :

```
let str = 'ES6 Tutorial';
let re = /ES\d/g

console.log(str.match(re));
```

Output:

```
['ES6']
```

Inverse Classes

A character class has an inverse class with the same letter but in the uppercase e.g., `\D` is the inverse of `\d` .

The inverse class matches all the other characters. For example, the `\D` match any character except a digit (or `\d`). The following are the inverse classes:

- `\D` – matches any character except a digit e.g., a letter.
- `\S` – matches any character except a whitespace e.g., a letter
- `\W` – matches any character except a word character e.g., non-Latin letter or space.

Back to the phone number example, you can use the `\d` with the global flag (`g`):

```
let phone = '+1-(408)-555-0105';
let re = /\d/g;

console.log(phone.match(re).join(''));
```

Output:

```
14085550105
```

Alternatively, you can remove the non-digit using the `\D` inverse class and replace all non-digit characters with blank, like this:

```
let phone = '+1-(408)-555-0105';
let re = /\D/g;

console.log(phone.replace(re,''));
```

Output:

```
14085550105
```

The dot (.) character class

The dot (`.`) is a special character class that matches any character except a newline:

```
let re = /E.6/
console.log('ES6'.match(re));
```

Output:

```
['ES6', index: 0, input: 'ES6', groups: undefined]
```

However, the following example returns `null` :

```
let re = /ES.6/
console.log('ES\n6'.match(re));
```

If you want to use the dot (`.`) character class to match any character including the newline, you can use the `s` flag:

```
let re = /ES.6/s
console.log('ES\n6'.match(re));
```

Output:

```
['ES\n6', index: 0, input: 'ES\n6', groups: undefined]
```

Summary

- Character classes match any symbol from certain character sets e.g., `\d` , `\s` , and `\w` .

- The character classes `\d`, `\s`, and `\w` have the inverse classes `\D`, `\S` and `\W` that match other characters except `\d`, `\s` and `\w`.
- The dot(`.`) matches any character except the newline character. Use the `s` flag to make the dot (`.`) character class matches any character including the newline.