Cheat Sheet – Reflect API

Metaprogramming

Metaprogramming means that you're able to change (parts of) the behavior of the underlying language – JavaScript in this case. This of course is a powerful feature as it allows you to influence the way your code is executed. The Reflect API (like Symbols and Proxies) are important additions which help you with Metaprogramming – something that wasn't really possible in JavaScript before.

What is the Reflect API?

The Reflect API could be described as a collection or "central place" which houses all kind of object and function related functions (for creation, property management etc.). Some of the functionalities added on the Reflect object where available before on the Object object.
But the goal for the future is, to have on central place to store all those methods – the Reflect Object/ API.
Therefore, the Reflect API provides useful methods to create, manipulate and query objects and functions in your JavaScript project.

Object Construction

You can easily create a new object using Reflect.construct().

```
class Person {
}
let person = Reflect.construct(Person, []);
```

The arguments passed are the constructor/ class used to create a new object, arguments passed to this constructor and, optionally as a third argument, another constructor which should be used. Learn more about constructing an object with Reflect here: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Reflect/construct

Prototypes

You may also use Reflect to set the prototype of an object. Simply do this by using the setPrototype() method.

```
class Person {
}
let config = {
    greet() {
        console.log('Hello there!');
    }
}
let person = Reflect.construct(Person, []);
Reflect.setPrototypeOf(person, config)
```

Properties

Reflect also offers methods to define properties on objects (defineProperty()), delete them (deleteProperty()), as well as get (get()) and set (set()) properties.

Functions

You may also use Reflect to execute functions:

```
Reflect.apply(person.greet, person, []);
```

The arguments passed are the function to be executed, what *this* should be referring to and possible function arguments.

Learn More

Find more information on the Reflect API here: https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global_Objects/Reflect