Reflect

1. Reflect.get (target, propertyKey, receiver)

This method returns the value of the property specified by the propertyKey on the target object. The receiver parameter is an optional parameter that specifies the object that should be used as the receiver for the get operation.

Example:

```
let obj = { name : "Eman" };

let name = Reflect.get (obj , "name" );

console.log(name); // Output: " Eman "
```

2. Reflect.set (target, propertyKey, value, receiver)

This method sets the value of the property specified by the propertyKey on the target object. The receiver parameter is an optional parameter that specifies the object that should be used as the receiver for the set operation.

Example:

```
let obj = {name: "Eman"};

Reflect.set (obj, "name", "Eman");

console.log (obj.name); // Output: " Eman "
```

3. Reflect.has (target, propertyKey)

This method returns a Boolean indicating whether the target object has the specified property.

Example:

```
let obj = {name: "Eman"};

let hasName = Reflect.has (obj, "name");

console.log (hasName); // Output: true
```

4. Reflect.deleteProperty(target, propertyKey)

This method deletes the specified property from the target object.

Example:

```
let obj = {name: "Eman"};

Reflect.deleteProperty (obj, "name");

console.log (obj.name); // Output: undefined
```

5. Reflect.getOwnPropertyDescriptor(target, propertyKey)

This method returns the property descriptor for the specified property on the target object.

Example:

```
let obj = {name: "Eman"};

let descriptor = Reflect.getOwnPropertyDescriptor(obj, "name");

console.log (descriptor);
// {
//   value: 'John',
//   writable: true,
//   enumerable: true,
//   configurable: true
// }
```

6. Reflect.defineProperty(target, propertyKey, attributes)

This method sets the property descriptor for the specified property on the target object.

Example:

```
let obj = {};

Reflect.defineProperty(obj, "name", {
  value: "Eman",
  writable: true,
  enumerable: true,
  configurable: true
 });
 console.log (obj.name); // Output: Eman
```

7. Reflect.construct(target, argumentsList, newTarget)

This method creates an instance of the target object, using the argumentsList as the arguments to the constructor.

The newTarget parameter is an optional parameter that specifies the constructor to be used as the constructor for the created instance.

Example:

```
class Person {

  constructor(name) {
    this.name = name;
  }
}
let person = Reflect.construct(Person, ["Eman"]);

console.log (person instanceof Person); // Output: true
```

8. Reflect.apply(target, thisArgument, argumentsList)

This method is used to call the target function with the given this argument and arguments. It works similarly to the Function.prototype.apply() method.

Example:

```
function sum(a, b) {
  return a + b;
}
let result = Reflect.apply(sum, this, [1, 2]);

console.log(result); // Output: 3
```

9. Reflect.getPrototypeOf(target)

This method is used to get the prototype of an object. It works similarly to the Object.getPrototypeOf() method.

Example:

```
let obj = { name: "Eman" };

let prototype = Reflect.getPrototypeOf(obj);

console.log (prototype === Object.prototype); // Output: true
```

10. Reflect.setPrototypeOf(target, prototype)

This method sets the prototype of an object. It returns a Boolean value indicating whether the prototype was successfully set.

Example:

```
const obj = { };

const proto = {};

console.log (Reflect.setPrototypeOf(obj, proto)); // Output: true
console.log (Reflect.getPrototypeOf(obj) === proto); // Output: true
```

11. Reflect.isExtensible(target)

This method returns a boolean indicating whether an object is extensible, meaning that new properties can be added to it.

Example:

```
const obj = { };

console.log (Reflect.isExtensible(obj)); // Output: true
```

12. Reflect.preventExtensions(target)

This method makes an object non-extensible, meaning that new properties cannot be added to it. It returns a boolean indicating whether the object was successfully made non-extensible.

Example:

```
const obj = { };

console.log (Reflect.preventExtensions(obj)); // Output: true

console.log (Reflect.isExtensible(obj)); // Output: false
```

13. Reflect.ownKeys(target)

This method returns an array of all own properties (including non-enumerable properties) of an object.

Example:

```
let obj = { name: "Eman" , age: 23 };

console.log (Reflect.ownKeys (obj));

//Output: [ "name", "age" ]
```