What is JavaScript `Reflect`?

JavaScript Reflect is an inbuilt ES6 global object that provides the ability to manipulate properties, variables, and object methods at runtime. It is not a constructor, therefore you cannot use the new operator with it.

Methods:

1. Reflect.apply()

- Syntax : Reflect.apply(target, thisArgument, argumentsList)
- Description
    - In ES5, you typically use the Function.prototype.apply() method to call a function with a given this value and arguments provided as an array (or an array-like object).
- Parameters :
    - Target: The target function to call.
    - thisArgument: The value of this provided for the call to target.
    - argumentsList: An array-like object specifying the arguments with which target should be called.
- Return Value
    - The result of calling the given target function with the specified this value and arguments.
- Examples:

```
/* Return the highest value in the array */

const arr = [3,5,20,3,31]
const a = Reflect.apply(Math.max, undefined, arr)
console.log(a)

// Result: 31
```

2. Reflect.construct()

- Syntax : Reflect.construct(target, argumentsList) || Reflect.construct(target, argumentsList)
- Description:
    - allows you to invoke a constructor with a variable number of arguments. (This would also be possible by using the spread syntax combined with the new operator.)
- Parameters :
    - Target: The target function to call.
    - argumentsList: An array-like object specifying the arguments with which target should be called.
    - newTarget (optinal): The constructor whose prototype should be used. See also the new.target operator. If newTarget is not present, its value defaults to target.
- Return Value:
    - A new instance of target (or newTarget, if present), initialized by target as a constructor with the given argumentsList.
- Examples:

```
function summation(x,y,z){
  this.add = x + y +z
}
const sum = Reflect.construct(summation, [1,2,3,4,5])
console.log(sum)

// Result: summation {add: 6}
```

3. Reflect. defineProperty ()

- Syntax : Reflect.defineProperty(target, propertyKey, attributes)
- Description:
    - The Reflect.defineProperty method allows precise addition to or modification of a property on an object. For more details, see the Object.defineProperty which is similar.
- Parameters :
    - Target: The target object on which to define the property.
    - propertyKey: The name of the property to be defined or modified.
    - attributes: The attributes for the property being defined or modified.
- Return Value:
    - A Boolean indicating whether or not the property was successfully defined.
- Example:

```
const obj = {}
Reflect.defineProperty(obj, 'prop', {value: 70})
console.log(obj.prop)
// Result: 70
```

4. Reflect.deleteProperty()

- Syntax : Reflect.deleteProperty(target, propertyKey)
- Description:
    - The Reflect.deleteProperty method allows you to delete a property on an object. It returns a Boolean indicating whether or not the property was successfully deleted. It is almost identical to the non-strict delete operator.
- Parameters :
    - Target: The target object on which to delete the property.

    - propertyKey: The name of the property to be deleted.
- Return Value:
    - A Boolean indicating whether or not the property was successfully deleted.
- Examples:

```
Reflect.deleteProperty(obj3, 'age');
console.log(obj3)
```

5. Reflect.get()

- Syntax : Reflect.get(target, propertyKey) || Reflect.get(target, propertyKey, receiver)
- Description: The Reflect.get method allows you to get a property on an object. It is like the property accessor syntax as a function
- Parameters :
    - Target: The target object on which to get the property.
    - propertyKey: The name of the property to get.
    - Reciever (optional) : The value of this provided for the call to target if a getter is encountered. When used with Proxy, it can be an object that inherits from target.
- Return Value: The value of the property.
- Examples:

```
// with array
const b = [10,11,12,13,14]
console.log(Reflect.get(b, 2))
// Result: 12

// with object
const obj = {name: "Pascal", age: 23}
console.log(Reflect.get(obj, 'age'))
// Result: 23
```

6. Reflect. getOwnPropertyDescriptor

- Syntax: Reflect.getOwnPropertyDescriptor(target, propertyKey)
- Parameters
  - Target: The target object in which to look for the property.
  - propertyKey: The name of the property to get an own property descriptor for.
- Return value
  - A property descriptor object if the property exists in target object; otherwise, undefined.
- Exceptions
  - A TypeError, if target is not an Object.
- Description
  - The Reflect.getOwnPropertyDescriptor method returns a property descriptor of the given property if it exists in the target object, undefined otherwise. The only difference to Object.getOwnPropertyDescriptor() is how non-object targets are handled.
- Examples

```
const obj = {
  car: "Rolls Royce",
  color: "black",
  get (){
    return `I have a ${color} ${car} car`
  }
};

console.log(Reflect.getOwnPropertyDescriptor(obj, 'car').value);
// "Rolls Royce"
```

7. Reflect.set()

- Syntax:
  - Reflect.set(target, propertyKey, value)
  - Reflect.set(target, propertyKey, value, receiver)
- Parameters
  - Target: The target object on which to set the property.
  - propertyKey:The name of the property to set.
  - Value: The value to set.
  - receiver Optional: The value of this provided for the call to the setter for propertyKey on target. If provided and target does not have a setter for propertyKey, the property will be set on receiver instead.
- Return value
  - A Boolean indicating whether or not setting the property was successful.
- Exceptions
  - A TypeError, if target is not an Object.
- Description
  - The Reflect.set method allows you to set a property on an object. It does property assignment and is like the property accessor syntax as a function.
- Examples

```
const arr1 = [];
Reflect.set(arr1, 0, 'first');
Reflect.set(arr1, 1, 'second');
Reflect.set(arr1, 2, 'third');
console.log(arr1);
```

8. Reflect.has()

- Syntax: Reflect.has(target, propertyKey)
- Parameters
  - Target: The target object in which to look for the property.
  - propertyKey : The name of the property to check.
- Return value
  - A Boolean indicating whether or not the target has the property.
- Exceptions
  - A TypeError, if target is not an Object.
- Description
  - The Reflect.has method allows you to check if a property is in an object. It works like the in operator as a function.
- Examples:

```
const obj = {
  name: "Douglas"
};

console.log(Reflect.has(obj, 'name'));
// true

console.log(Reflect.has(obj, 'age'));
// false

console.log(Reflect.has(obj, 'toString'));
// true
```

9. Reflect.ownKeys()

- Syntax: Reflect.ownKeys(target)
- Parameters
  - Target: The target object from which to get the own keys.
- Return value
  - An Array of the target object's own property keys.
- Exceptions
  - A TypeError, if target is not an Object.
- Description
  - The Reflect.ownKeys method returns an array of the target object's own property keys. Its return value is equivalent to Object.getOwnPropertyNames(target).concat(Object.getOwnPropertySymbols(target)).
- Examples

```
const obj = {
  car: "Rolls Royce",
  color: "black"
};

const array1 = [];

console.log(Reflect.ownKeys(obj));
// ["car", "color"]

console.log(Reflect.ownKeys(array1));
// ["length"]
```