11、对象的新增方法

① Object.is()

- ES5 比较两个值是否相等，只有两个运算符：相等运算符（==）和严格相等运算符（===）。
- 它们都有缺点，前者会自动转换数据类型，后者的NaN不等于自身，以及+0等于-0。
- JavaScript 缺乏一种运算，在所有环境中，只要两个值是一样的，它们就应该相等。
- ES6 提出“Same-value equality”（同值相等）算法，用来解决这个问题。
- Object.is就是部署这个算法的新方法。它用来比较两个值是否严格相等，与严格比较运算符（===）的行为基本一致。

```
Object.is('foo', 'foo') // true
Object.is({}, {}) //false

+0 === -0 //true
NaN === NaN // false

Object.is(+0, -0) // false
Object.is(NaN, NaN) // true
```

② Object.assign()

- Object.assign方法用于对象的合并，将源对象（source）的所有可枚举属性，复制到目标对象（target）。

```
const target = { a: 1 };
const source1 = { b: 2 };
const source2 = { c: 3 };

Object.assign(target, source1, source2);
target // {a:1, b:2, c:3}
```

- Object.assign方法的第一个参数是目标对象，后面的参数都是源对象。
- 注意，如果目标对象与源对象有同名属性，或多个源对象有同名属性，则后面的属性会覆盖前面的属性。

```
const target = { a: 1, b: 1 };

const source1 = { b: 2, c: 2 };
const source2 = { c: 3 };

Object.assign(target, source1, source2);
target // {a:1, b:2, c:3}
```

③ Object.getOwnPropertyDescriptors()

- ES5 的Object.getOwnPropertyDescriptor()方法会返回某个对象属性的描述对象（descriptor）。
- ES2017 引入了Object.getOwnPropertyDescriptors()方法，返回指定对象所有自身属性（非继承属性）的描述对象。

```
const obj = {
  foo: 123,
  get bar() { return 'abc' }
};

Object.getOwnPropertyDescriptors(obj)
// { foo:
//    { value: 123,
//      writable: true,
//      enumerable: true,
//      configurable: true },
//   bar:
//    { get: [Function: get bar],
//      set: undefined,
//      enumerable: true,
//      configurable: true } }
```

- 上面代码中，Object.getOwnPropertyDescriptors()方法返回一个对象，所有原对象的属性名都是该对象的属性名， 对应的属性值就是该属性的描述对象。

④ __proto__属性，Object.setPrototypeOf(), Object.getPrototypeOf()

- JavaScript 语言的对象继承是通过原型链实现的。ES6 提供了更多原型对象的操作方法。
- 具体详见文档

⑤ Object.keys()，Object.values()，Object.entries()

- ES5 引入了Object.keys方法，返回一个数组，
- 成员是参数对象自身的（不含继承的）所有可遍历（enumerable）属性的键名。

```
var obj = { foo: 'bar', baz: 42 };
Object.keys(obj)
// ["foo", "baz"]
```

- ES2017引入了跟Object.keys配套的Object.values和Object.entries，
- 作为遍历一个对象的补充手段，供for...of循环使用。

```
let {keys, values, entries} = Object;
let obj = { a: 1, b: 2, c: 3 };
```

```
for (let key of keys(obj)) {
  console.log(key); // 'a', 'b', 'c'
}

for (let value of values(obj)) {
  console.log(value); // 1, 2, 3
}

for (let [key, value] of entries(obj)) {
  console.log([key, value]); // ['a', 1], ['b', 2], ['c', 3]
}
```