Para eu copiar dados de um objeto para outro objeto basta eu usar o spread operator:

```
const product = { name: 'Developer', whatdo: 'solve problems'};
const product2 = {
    ...product,
    single: 'no'
};

product2.name = 'Leonardo';
product2.whatdo = 'help people';
console.log(product);
console.log(product2);
```

Resultado:

```
{ name: 'Developer', whatdo: 'solve problems' }
{ name: 'Leonardo', whatdo: 'help people', single: 'no' }
```

Mas eu posso fazer isso de outra forma usando o Object.assign(des, any):

```
const product2 = Object.assign({}, product);
```

Eu posso mandar quantos objetos eu quiser para a variável que está recebendo os valores copiados.

Eu também posso mandar assim:

```
const product2 = {name: product.name, whatdo: product.whatdo};
```

Essa não é a melhor forma mas pode ser útil caso eu precise pegar só uma propriedade de um objeto.

Object.keys mostra as chaves do Objeto:

```
const product = { name: 'Developer', whatdo: 'solve problems'};
const product2 = {name: product.name, whatdo: product.whatdo};

console.log(Object.keys(product));
```

Resultado:

```
[ 'name', 'whatdo' ]
```

Object.freeze indica que nada pode ser alterado no objeto.

O getOwnPropertyDescriptor vai mostrar os valores das propriedades do DefineProperty:

```
const product = { name: 'Developer', whatdo: 'solve problems'};
console.log(Object.getOwnPropertyDescriptor(product, 'name'));
```

Resultado:

```
{
  value: 'Developer',
  writable: false,
  enumerable: true,
  configurable: false
}
```

Eu posso usar também DefineProperty para alterar as propriedades:

```
const product = { name: 'Developer', whatdo: 'solve problems'};
Object.defineProperty(product, 'name', {
    writable: false,
    configurable: false,
});
console.log(Object.getOwnPropertyDescriptor(product, 'name'));
product.name = 'Any';
delete product.name;
console.log(product);
```

Ele não altera o nome do meu Objeto.

```
{
  value: 'Developer',
  writable: false,
  enumerable: true,
  configurable: false
}
{ name: 'Developer', whatdo: 'solve problems' }
```

E aí eu posso alterar os descritores das minhas propriedades (DefineProperty, DefineProperties). Então o vai retornar o descrito da propriedade dentro do objeto se é writable, value, enumerable e configurable.

```
Object.values()
```

Ele retorna os valores das propriedades do Objeto;

Se eu quiser as chaves e os valores eu uso Object.entries em arrays:

```
const product = { name: 'Developer', whatdo: 'solve problems'};
console.log(Object.entries(product));
```

```
[ [ 'name', 'Developer' ], [ 'whatdo', 'solve problems' ] ]
```

Eu posso fazer um For of para pegar cada array com 2 objetos:

```
for (let entry of Object.entries(product)) {
    console.log(entry);
}
```

```
[ 'name', 'Developer' ]
[ 'whatdo', 'solve problems' ]
```

E eu posso fazer Desctruring:

```
for (let [key, value] of Object.entries(product)) {
    console.log(key, value);
}
```

```
name Developer
whatdo solve problems
```

Resumo:

Object.values;
Object.entries;
Object.getOwnPropertyDescriptor(o, ‘prop’);
Object.assign(des, any)
… (spread)

Object.keys (retorna as chaves);
Object.freeze (congela o objeto);
Object.defineProperty (define uma propriedade)
Object.defineProperties (define várias propriedades)