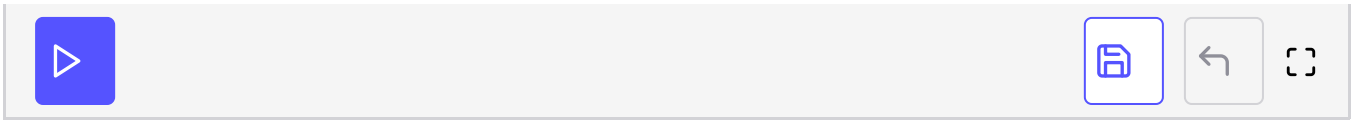
Object.assign

Here we'll cover 'Object.assign', a new utility method that makes copying objects and object properties easier.

`Object.assign` is meant to assign properties from one object to another. The first parameter is the object that will have properties added to it. The next parameters are all objects whose properties will be placed onto the first object. The return value is the object that was passed in as the first parameter.

```
const obj1 = { key1: 'val1'};
const obj2 = { key2: 'val2' };
const obj3 = { key3: 'val3' };

const obj = Object.assign(obj1, obj2, obj3);
console.log(obj); // -> { key1: 'val1', key2: 'val2', key3: 'val3' }
console.log(obj1 === obj); // -> true
```



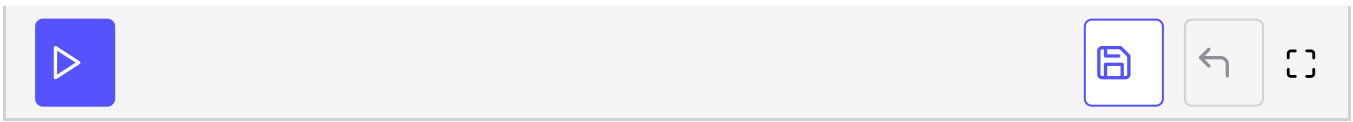
It creates shallow copies, meaning that references are copied directly. If a property of an object is an object or array, `Object.assign` will not recursively go through that item. As a result, the new object shares the same reference as the object being copied from.

```
const obj1 = { obj1key: 'obj1val' };
const obj2 = { outerKey: { innerKey: 'val' } };

Object.assign(obj1, obj2);

console.log(obj1);
// -> { obj1key: 'obj1val', outerKey: { innerKey: 'val' } }

console.log(obj1.outerKey === obj2.outerKey); // -> true
```

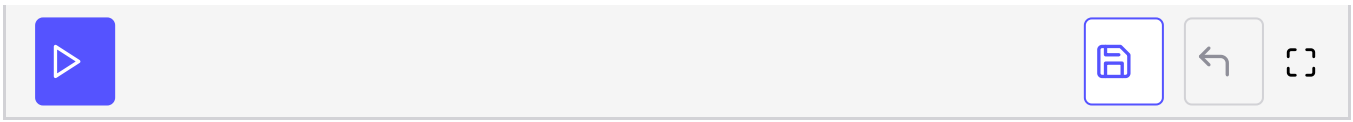


One of its most useful abilities is the ability to create a shallow copy of an object. Calling `Object.assign` with an empty object and another object makes

a copy of the second object.

```
const obj1 = {
    name: 'Alex Smith',
    age: 30,
    address: 'CA, USA',
};

const obj2 = Object.assign({}, obj1);
console.log(obj2);
// -> { name: 'Alex Smith', age: 30, address: 'CA, USA' }
```



`obj2` is a replica of `obj1`.

`...`

Using `Object.assign` is a functional equivalent to using object spread.

That's it.