Creating Objects

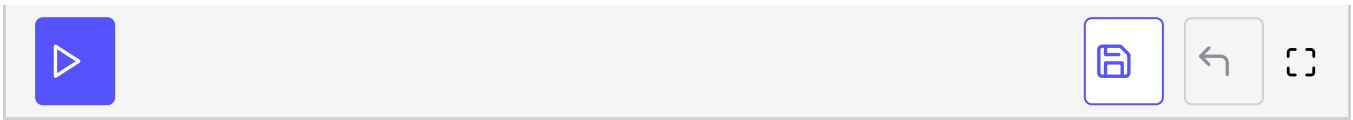
Reflect.construct and its behavior

We can instantiate classes with `Reflect.construct`. Consider the following code:

```
let target = class Account {
    constructor( name, email ) {
        this.name = name;
        this.email = email;
    }
    get contact() {
        return `${this.name} <${this.email}>`;
    }
};
let args = [
    'Zsolt',
    'info@zsoltnagy.eu'
];

let myAccount = Reflect.construct(
    target,
    args );

console.log(myAccount.contact);
//> "Zsolt <info@zsoltnagy.eu>"
```



Arguments of `Reflect.construct`:

- `target`: the function we will construct
- `args`: array of arguments
- `newTarget`: `new.target` value (optional)

With the following example, we have full control over the value of `new.target` in the constructor:

```
let target = class Account {
    constructor( name, email ) {
```

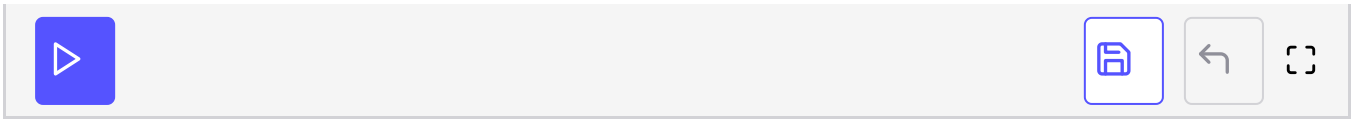
```
        this.name = name;
        this.email = email;
    }

    get contact() {
        return `${this.name} <${this.email}>`;
    }
};
let args = [
    'Zsolt',
    'info@zsoltnagy.eu'
];
let newTarget = class PrivateAccount {
    get contact() {
        return 'Private';
    }
}

let myAccount = Reflect.construct(
    target,
    args,
    newTarget );

console.log(myAccount.contact);
//> "Private"

console.log(myAccount);
//> PrivateAccount {name: "Zsolt", email: "info@zsoltnagy.eu"}
```



Now, let's talk about how we can manipulate prototypes of objects using Reflect API.