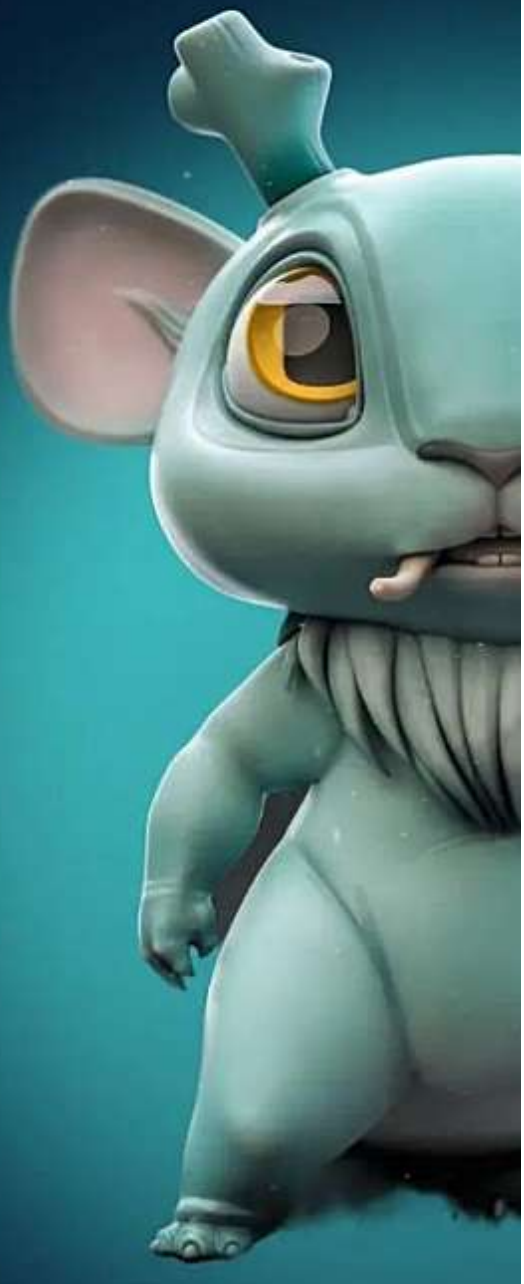
JS

JavaScript Object methods you must know



Code For Real
@codeforreal

Object.assign()

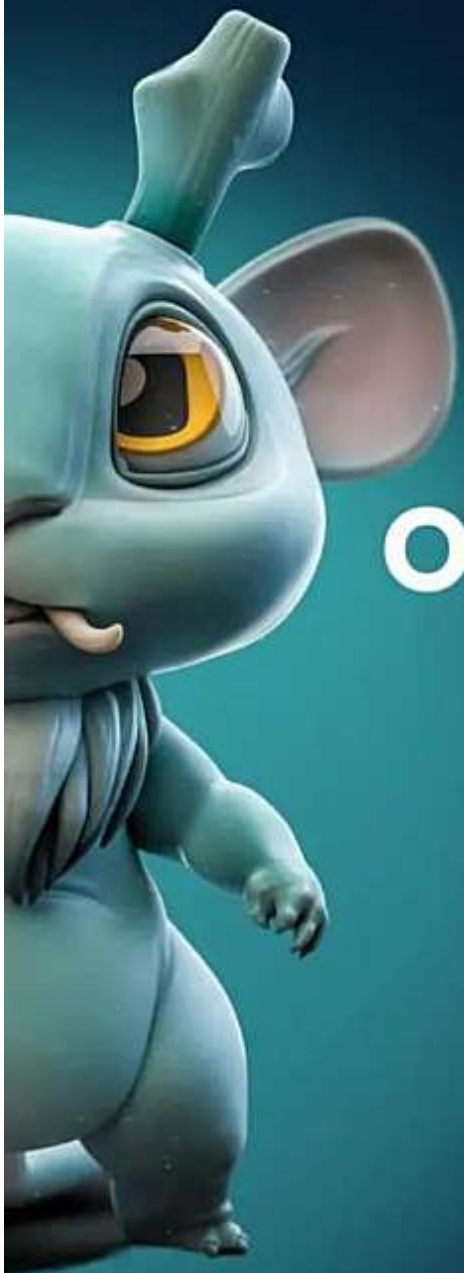
Object.assign() method copies all the entries of source object to a target object. Be **careful** with this one because it will not only return a copied object but it **changes** the **target** object as well.

```
const store1 = {one: 1, two: 2}

const store2 = {three: 3}

const result = Object.assign(store1, store2)

console.log(result) // {one: 1, two: 2, three: 3}

console.log(store1) // {one: 1, two: 2, three: 3}
```

Code For Real
@codeforreal

Code For Real
@codeforreal

Give a follow



Swipe to see some of the must know JavaScript Object methods

Dont' forget to save it

Save

Object.seal()

Object.seal() is similar to Object.freeze() but a sealed object can **only change/update** the values of entry. We cannot **add** or **delete** the existing items.

You can detect whether an object is sealed or not by using **Object.isSealed**(variable)

```
const store = Object.seal({
    one: 1,
    two: 2,
    three: 3
})

store.four = 4 // add -> Not allowed
store.one = 11 // update -> Allowed
delete store.two // delete -> not allowed

console.log(store)
/* {
    one: 11,
    two: 2,
    three: 3
} */
```

Code For Real
@codeforreal

Save

Object.fromEntries()

Object.fromEntries() converts an array of key value pair into an object. See an example below:

```
const keyValuePair = [ ['one',1], ['two', 2] ]

const store = Object.fromEntries(keyValuePair)

console.log(store)
/*
    {
        one: 1,
        two: 2
    }
*/
```

Code For Real
@codeforreal

Save

Object.freeze()

Object.freeze() in JavaScript is used to make an object **immutable**. That from the freezed object, any entry cannot be **updated**, any new entry cannot be **added** or any existing entry cannot be **deleted**.

You can detect whether an object is frozen or not by using **Object.isFrozen**(variable)

```
const store = Object.freeze({
    one: 1,
    two: 2,
    three: 3
})

store.four = 4 // add
store.one = 11 // update
delete store.two // delete

console.log(store) // No change at all
/* {
    one: 1,
    two: 2,
    three: 3
} */
```

Code For Real
@codeforreal

Save

Object.defineProperty()
Object.defineProperties()

Object.defineProperty() is used to **add** new property or **modify** existing property. This is actually preferred method compared to what you saw in previous slide i.e `***store.four = 4***`, because you can manage descriptor of the property

.

Object.defineProperties() is used to add/modify **multiple** properties at once.

```
const store = {
    one: 1,
}

Object.defineProperty(store, 'two', {
  enumerable: false,
  configurable: false,
  writable: false, // Cannot change in strict mode
  value: 2
})

console.log(store)
/* {
    one: 1,
    two: 2,
} */

store.two = 5 // Throws error in strict mode
```

```
Object.defineProperties(store, {
  three:{
    enumerable: false,
    configurable: false,
    writable: true,
    value: 3
  },
  four:{
    enumerable: false,
    configurable: false,
    writable: true,
    value: 4
  }
})

console.log(store)
/* {
    one: 1,
    two: 2,
    three: 3,
    four: 4
} */
```

Code For Real
@codeforreal

Save

5/10

Object.preventExtensions()

Object.preventExtensions() method does **not** allow to **add** any new entry to the object. It **allows** to **change/update** the entries and also **delete** the extisting entries but it will not allow to add any new entry after it has been created.

```
const store = Object.preventExtensions({
    one: 1,
    two: 2,
    three: 3
})

store.four = 4 // add -> Not allowed
store.one = 11 // update -> Allowed
delete store.two // delete -> Allowed

console.log(store)
/* {
    one: 11,
    three: 3
} */
```

Code For Real
@codeforreal

Save

Object.keys()
Object.values()
Object.entries()

Object.keys() returns all the keys of an object as an array. Keys are property names.

Object.values() returns all the values of an object as an array.

Object.entries() returns both keys and values of an object as an array.

```
const store = {one: 1, two: 2, three: 3}

Object.keys(store) // ['one', 'two', 'three']

Object.values(store) // [1, 2, 3]

Object.entries(store) // [['one',1],['two',2], ['three',3]]
```

Code For Real
@codeforreal

Save

Did you find it Useful?

Leave a comment!

Alamin CodePapa

@CodePapa360

FOLLOW FOR MORE

Like Comment Repost