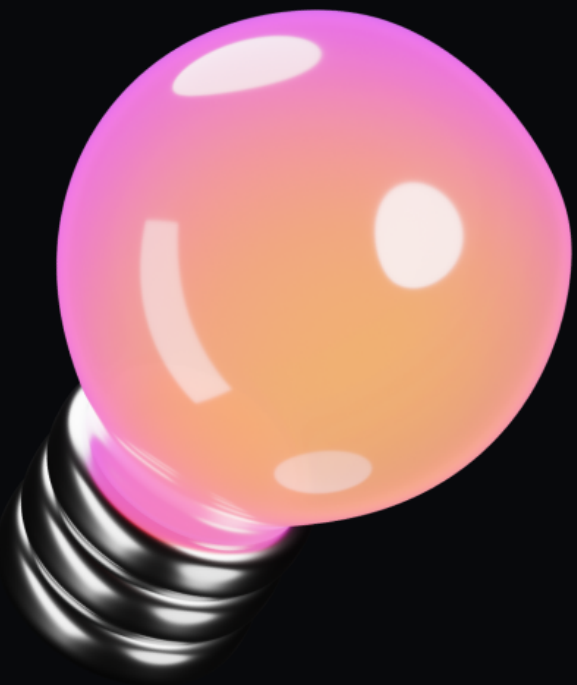
deeecode

OBJECT DESTRUCTURING

Extract properties from objects easily

JS



swipe left

deeecode

JS

OBJECT DESTRUCTURING

Object Destructuring makes it easier and quicker to extract properties from an object and assign the values to variables.

This approach is quicker than using object keys. To create variables out of object properties, you'd normally do this:

```
const object = {
  name: "Dillion",
  ig: "deeecode",
  language: "javascript",
}

const name = object.name
// "Dillion"

const instagram = object.ig
// "deeecode"
```

You can see the **name** and **instagram** variables declared on different lines, and using the keys of the object properties.

swipe left

deeecode

JS

OBJECT DESTRUCTURING

Using object destructuring, you can declare these variables on the same line:

```
const object = {
  name: "Dillion",
  ig: "deeecode",
  language: "javascript",
}

const { name, ig: instagram } = object

console.log(name, instagram)
// "Dillion"
// "deeecode"
```

For **name**, since the variable is not different from the key, you can just extract the key which will be the variable name.

For **instagram**, since the variable is different from the key (**ig**), you can use colon to assign the property to a new variable.

swipe left

JS

OBJECT DESTRUCTURING

If you try to extract a property that does not exist in the object, the variable will be **undefined**:

```
const object = {
  name: "Dillion",
  ig: "deeecode",
  language: "javascript",
}

const { name, ig: instagram, age, tw: twitter } = object

console.log(name, instagram, age, twitter)
// "Dillion"
// "deeecode"
// undefined
// undefined
```

Since the **age** and **tw** properties do not exist in the object, **undefined** is assigned to both the **age** and **twitter** variables.

swipe left

deeecode

JS

OBJECT DESTRUCTURING

You can destructure nested objects also:

```
const object = {
  name: "Dillion",
  ig: "deeecode",
  languages: {
    first: "javascript",
    second: "css",
  },
}

const {
  name,
  ig: instagram,
  languages: { first: firstLanguage, second },
} = object

console.log(name, instagram, firstLanguage, second)
// "Dillion"
// "deeecode"
// "javascript"
// "css"
```

Using the key of the nested object, you can destructure the nested object.

deeecode

JS

OBJECT DESTRUCTURING

You can pass default values when you destructure an object:

```
const object = {
  name: "Dillion",
  ig: "deeecode",
  language: "javascript"
}

const {name, ig: instagram = "myname", age = 50} = object

console.log(name, instagram, age)
// "Dillion"
// "deeecode"
// 50
```

You can assign default values to variables using the equal sign. In the case of **instagram**, the default value “**myname**” is not used because the **ig** property exists in the object. In the case of **age**, it does not exist in the object, so **50**, the default value is used.

swipe left