NodeJS

LABA

OBJECTS

Solvd
NodeJS LABA

OBJECTS

An object is a collection of related data and/or functionality

These usually consist of several variables and functions (which are called properties and methods when they are inside objects).

An empty object can be created using one of two syntaxes:

```
// "object constructor" syntax
let user = new Object();
// "object literal" syntax
let user = {};
```

Solvd NodeJS LABA

LITERALS AND PROPERTIES

A property has a key (also known as “name” or “identifier” before the colon “:” and a value to the right of it

Property values are accessible using the dot notation:

`user.name`

To remove a property, we can use the delete operator:

`delete user.age`

```
const user = { // an object
  name: "Jhon", // by key "name" store value "Jhon"
  age: 30, // by key "age" store value 30
};
```

Solvd NodeJS LABA

SQUARE BRACKETS

We can also use multiword property names, but then, they must be quoted:

```
const user = {
  name: "Jhon",
  age: 30,
  "likes birds": true,
};

let key = "likes birds";
// same as user["likes birds"] = true;
user[key] = true;
```

Solvd
NodeJS LABA

PROPERTY VALUE SHORTHAND

```
function makeUser(name, age) {
  return {
    name: name,
    age: age
  }
}

function makeUser(name, age) {
  return {
    name, // same as name: name
    age, // same as age: age
  }
}
```

Solvd
NodeJS LABA

FOR..IN

The `for` … `in` allows you to iterate over every key of an object within a for loop:

```
const user = {
  name: "Jhon",
  age: 26,
}

for (key in user) {
  console.log(key);
  // Output: name
  // Output: age
}
```

Solvd
NodeJS LABA

__PROTO__

```
const animal = {
  eats: true;
};

const rabbit = {
  jumps: true;
};

// sets rabbit.[[Prototype]] = animal
rabbit.__proto__ = animal;
```

Solvd
NodeJS LABA

__PROTO__

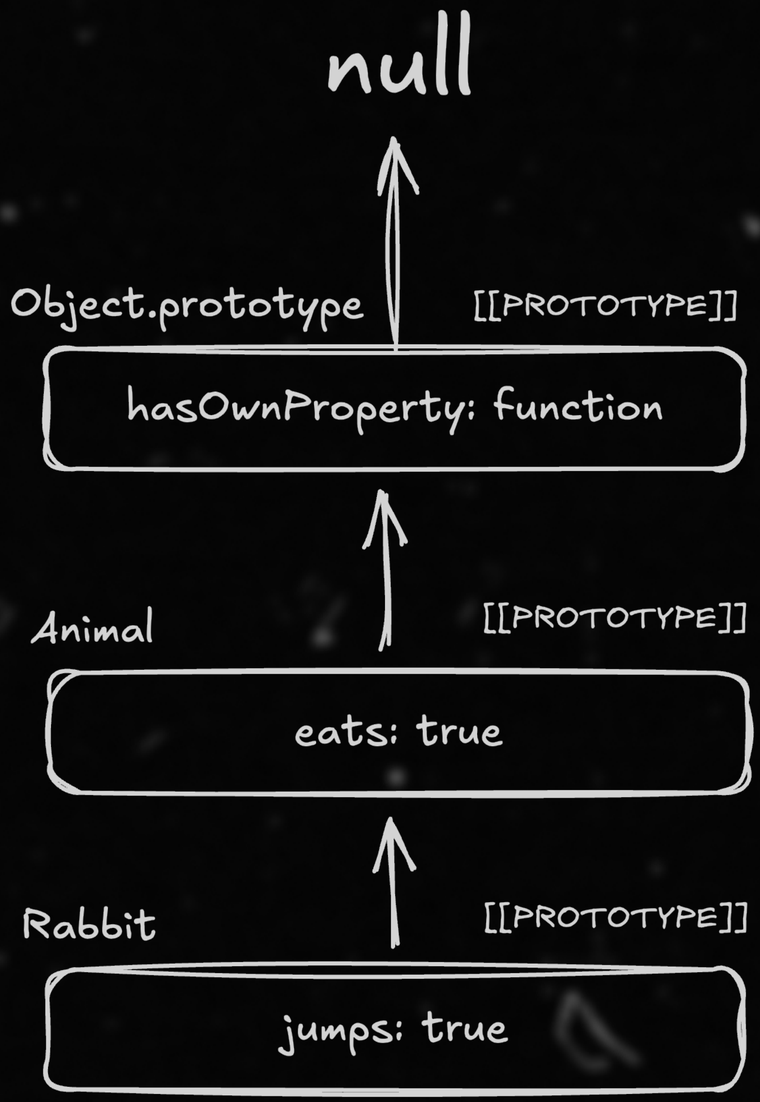
Limitations:

1. The references can’t go in circles. JavaScript will throw an error if we try to assign __proto__ in a circle.
2. The value of __proto__ can be either an object or null. Other types are ignored

There can be only one [[Prototype]]. An object can’t inherit from two others.

Solvd
NodeJS LABA

PROTOTYPE INHERITANCE



Solvd
NodeJS LABA

PROTOTYPE METHODS

The modern methods to get/set a prototype are:

```
Object.getPrototypeOf(obj)
```

returns the [[Prototype]] of obj.

```
Object.setPrototypeOf(obj, proto)
```

sets the [[Prototype]] of obj to proto

Solvd
NodeJS LABA

PROTOTYPE METHODS

There is a special method:

```
Object.create(proto[, descriptors])
```

Creates an empty object with the given proto as [[Prototype]] and optional property descriptors.

FLAGS AND DESCRIPTORS

`writable`

if true, the value can be changed, otherwise its read-only

`enumerable`

if true, then listed in loops, otherwise not listed

`configurable`

if true, the property can be deleted and its attributes can be modified

Solvd
NodeJS LABA

FLAGS AND DESCRIPTORS

The method `Object.getOwnPropertyDescriptor` allows to query the full information about a property.

```
const user = { name: "Jhon" }
const descriptor = Object.getOwnPropertyDescriptor(user, "name");
console.log(descriptor)

// Output
{
  configurable: true,
  enumerable: true,
  value: "Jhon",
  writable: true,
}
```

Solvd
NodeJS LABA

SEALING AN OBJECT GLOBALLY

There are also methods that limit access to the whole object:

```
Object.preventExtensions(obj)
```

Forbids the addition of new properties to the object.

```
Object.seal(obj)
```

Forbids adding/removing of properties. Sets configurable: false for all existing properties.

```
Object.freeze(obj)
```

Forbids adding/removing/changing of properties. Sets configurable and writable to false for all properties.

Solvd
NodeJS LABA

SEALING AN OBJECT GLOBALLY

And there are also tests for them.

```
Object.isExtensible(obj)
```

Returns false if adding properties is forbidden.

```
Object.isSealed(obj)
```

Returns true if adding/removing properties is forbidden.

```
Object.isFrozen(obj)
```

Returns true if adding/removing/changing properties is forbidden.

GETTERS AND SETTERS

```
const obj = {
  get propName() {
    // getter, the code executed on getting obj.propName
  },
  set propName(value) {
    // setter, the code executed on setting obj.propName = value
  },
}
```

Solvd
NodeJS LABA

This presentation is property of Solvd, Inc. It is intended for internal use only and may not be copied, distributed, or disclosed.