NodeJS

LABA

CLASSES & OOP

Solvd NodeJS LABA

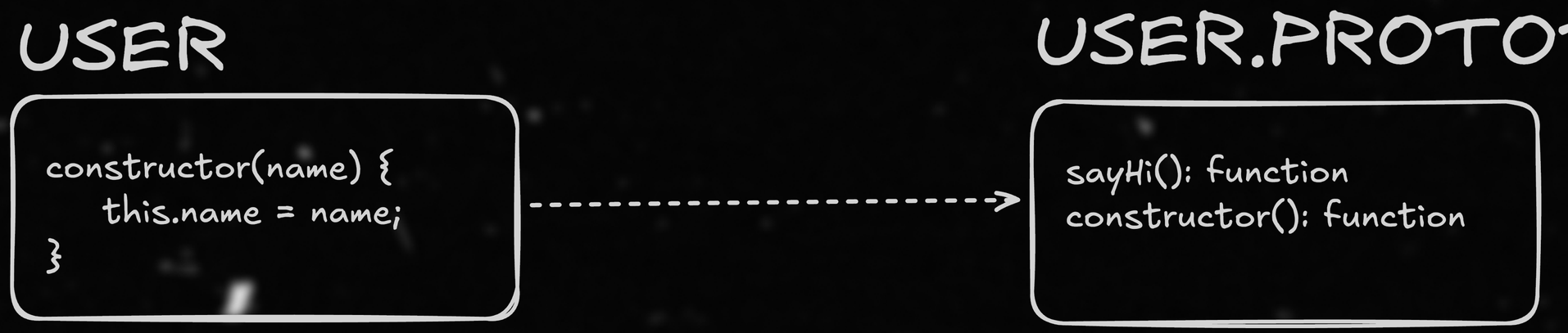
CLASSES

In object-oriented programming, a class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods).

```
class MyClass {
  // class methods
  constructor() { /* ... */ }
  method1() { /* ... */ }
  method2() { /* ... */ }
  method3() { /* ... */ }
  /* ... */
}
```

Solvd NodeJS LABA

CLASSES




```
class User {
  constructor(name) {
    this.name = name;
  }

  sayHi() {
    console.log(`Hi ${this.name}!`);
  }
}
```

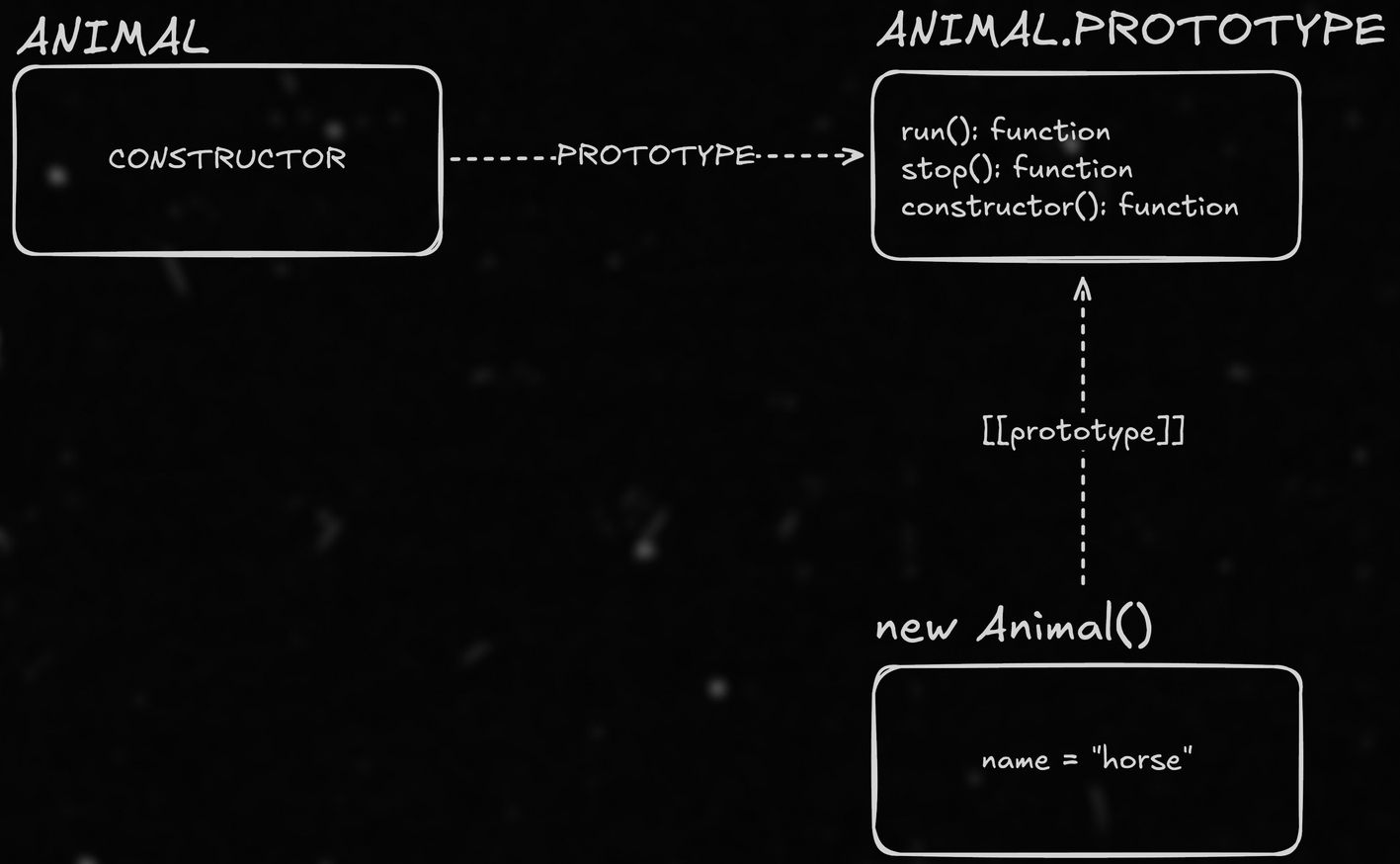
Solvd
NodeJS LABA

CLASSES VS FUNCTIONS

Differences:

1. A function created by class is labelled by a special internal property `[[IsClassConstructor]]: true`. So it's not entirely the same as creating it manually. The language checks for that property in a variety of places.
2. Class methods are non-enumerable. A class definition sets enumerable flag to false for all methods prototypes.
3. Classes always use strict.

CLASSES VS FUNCTIONS



CLASSES VS FUNCTIONS

Solvd
NodeJS LABA

SUPER

Classes provide "super" keyword for that.

- super.method(...) to call a parent method.
- super(...) to call a parent constructor (inside our constructor only).

GENERATED CONSTRUCTOR

```
class Rabbit extends Animal {
  // generated for extending classes without own constructors
  constructor(...args) {
    super(...args);
  }
}
```

Solvd
NodeJS LABA

CONSTRUCTORS

In JavaScript, there’s a distinction between a constructor function of an inheriting class (so-called “derived constructor”) and other functions. A derived constructor has a special internal property `[[ConstructorKind]]: “derived”.`, That affects its behavior with new.

- When a regular function is executed with new, it creates an empty object and assigns it to this.
- But when a derived constructor runs, it doesn’t do this. It expects the parent constructor to do this job.

ACCESS MODIFIERS

In object-oriented programming, properties and methods are split into two groups:

- Internal interface – methods and properties, accessible from other methods of the class, but not from the outside.
- External interface – methods and properties, accessible also from outside the class.

ACCESS MODIFIERS

In JavaScript, there are two types of object fields (properties and methods):

- Public: accessible from anywhere. They comprise the external interface.
- Private: accessible only from inside the class. These are for the internal interface.