Objekt…

… en introduktion

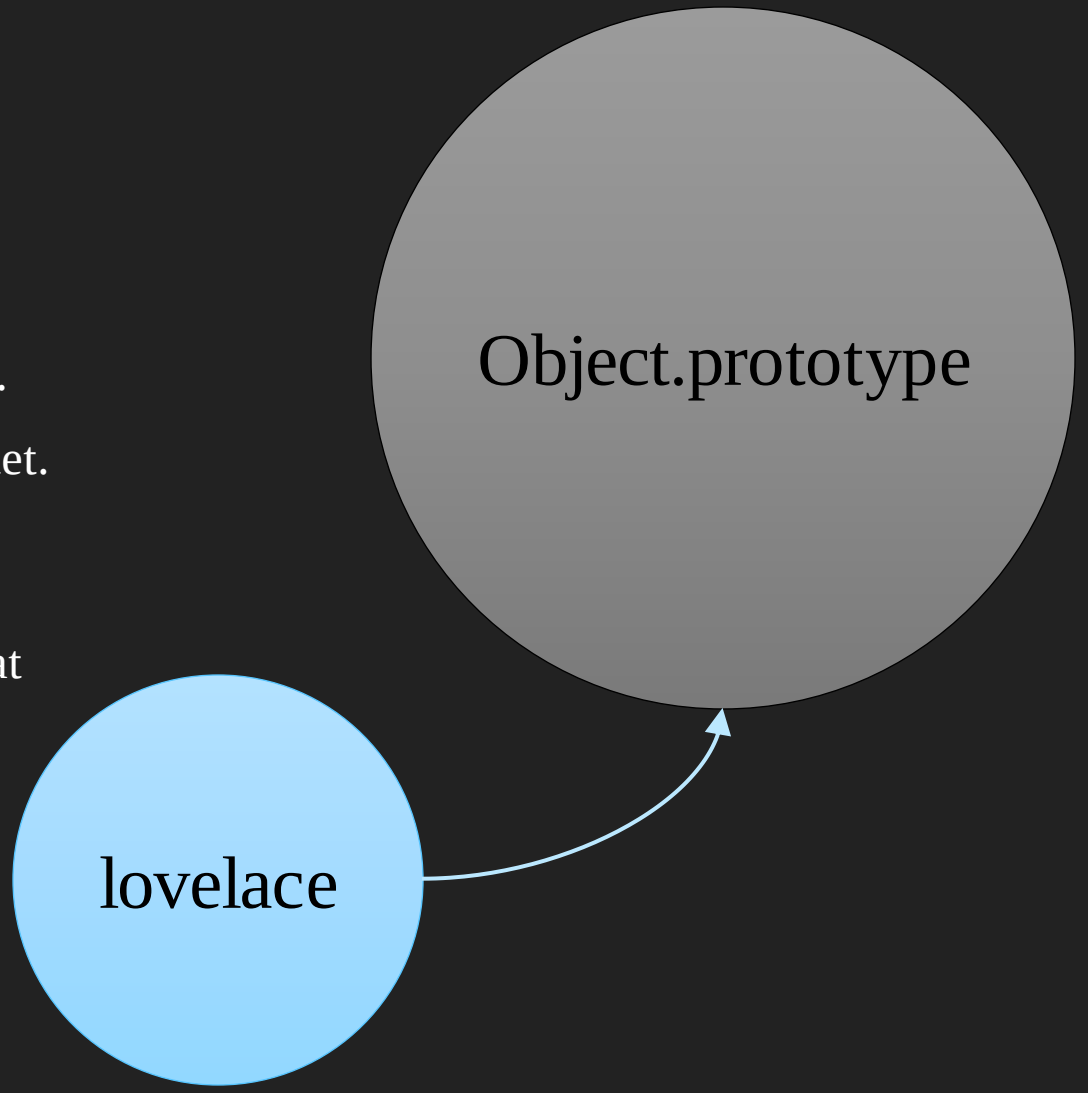
Så skapar du ett nytt objekt

- ✓ Ett sätt att skapa ett objekt är att använda en objektliteral.

```
const lovelace = {}
```

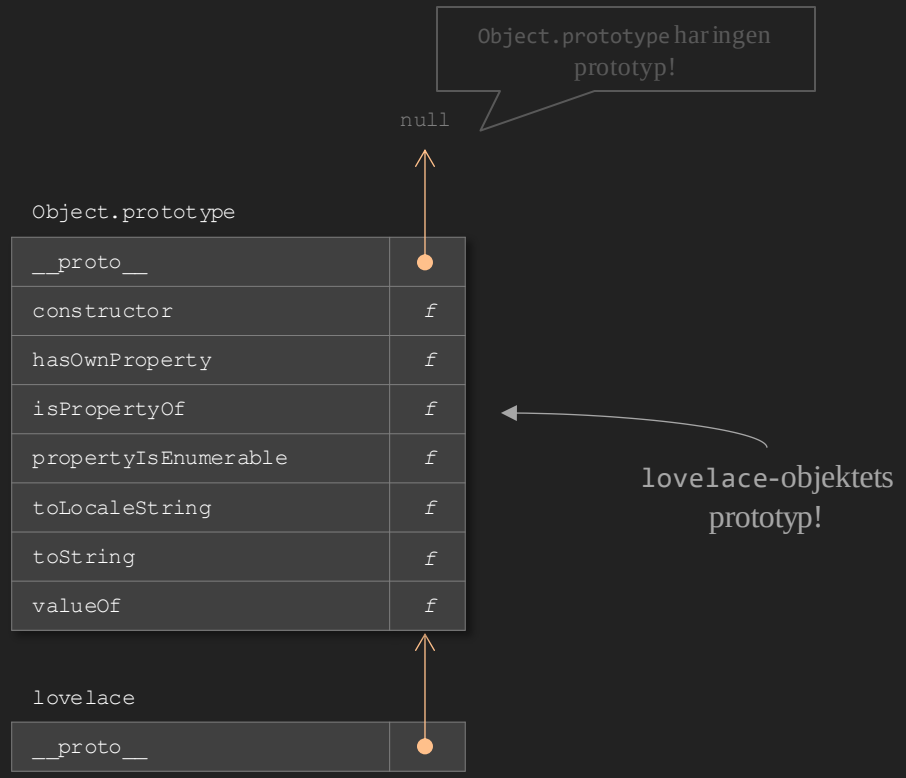
  - ”`{}`” skapar ett nytt tomt objekt av typen `Object`.
  - Konstanten `lovelace` refererar till det nya objektet.

- ✓ Det nya objektet står inte helt ensamt!
  - Det nya objektet `lovelace` refererar till är kopplat till `Object.prototype` och kan därför allt `Object.prototype` kan.



Ett objekt står inte ensamt!

- ✓ Med få undantag så är alltid ett objekt baserat på ett annat objekt, en prototyp.
- ✓ Ett skenbart tomt objekt har den interna egenskapen, `__proto__`, som refererar till prototypen.
  - En egenskap består av ett nyckel/värde-par.
    - nyckel - ett namn som är en sträng (eller `Symbol`-värde)
    - värde – primitivt värde, metod eller referens till objekt
- ✓ Eftersom `lovelace`-objektet är baserat på `Object.prototype` kan dess egenskaper användas genom `lovelace-objeket`.




```
const lovelace = {}

console.log(lovelace.toString()) // OUTPUT: [object Object]
console.log(lovelace.unknownMethod()) // ERROR: TypeError:  lovelace.unknownMethod is not a function
```

Sätta objekts egenskaper

- ✓ Det finns flera sätt att sätta objekts egenskaper.
  - Punktnotation

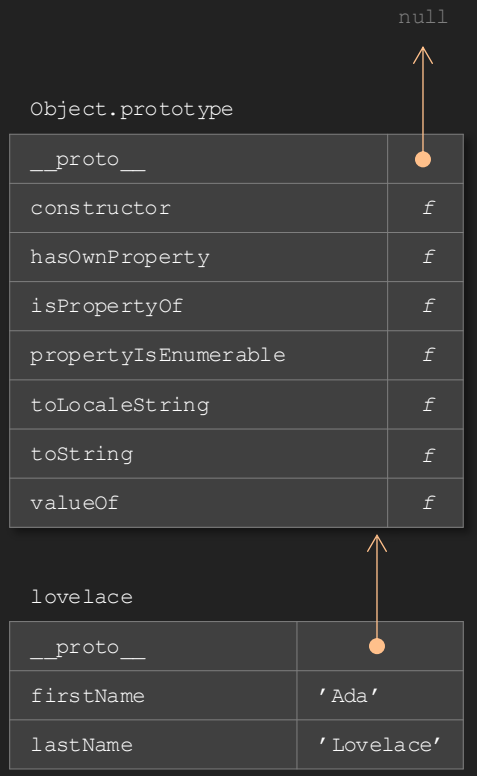
```
const lovelace = {}

lovelace.firstName = 'Ada'
lovelace.lastName = 'Lovelace'
```

  - Hakparentesnotation (används sällan med hårdkodad nyckel, tillåts inte av kursens kodstandard)

```
const lovelace = {}

lovelace['firstName'] = 'Ada'
lovelace['lastName'] = 'Lovelace'
```

  - Följande skapar ett objekt som initieras med två egenskaper.

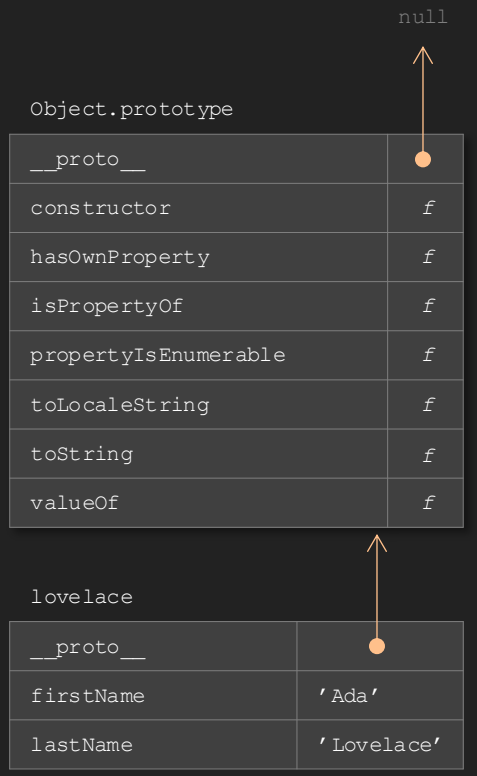
```
const lovelace = {
  firstName: 'Ada',
  lastName: 'Lovelace'
}
```



Hämta objekts egenskaper

- ✓ Ett objekts egenskaper hämtas genom punktnotation.
  - Hakparentesnotation kan också användas. Kursens kodstandard tillåter dock det inte om nyckelns värde hårdkodas.
- ✓ En egenskap som inte är känd ger värdet `undefined`.

```
const lovelace = {
  firstName: 'Ada',
  lastName: 'Lovelace'
}

console.log(lovelace.firstName) // OUTPUT: Ada

console.log(lovelace['firstName']) // OUTPUT: Ada

console.log(lovelace.unknownProperty) // OUTPUT: undefined
```



Så läggs en metod till ett objekt

- ✓ En metod är en egenskap vars värde är en funktion.
- ✓ Metoder kan läggas till objekt då de initieras.

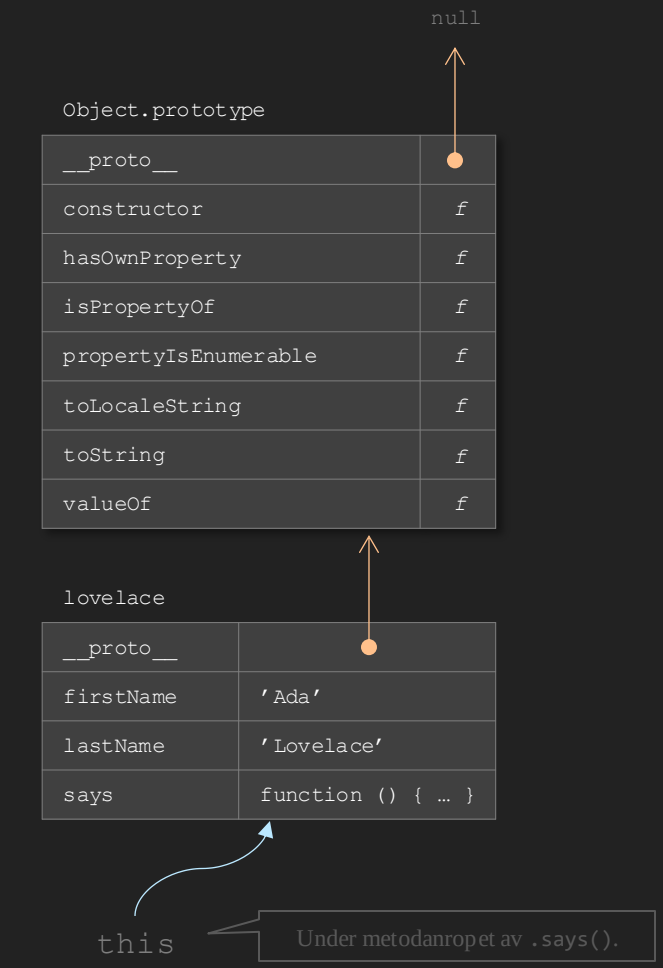
Det går självklart att lägga till metoder till ett objekt även efter att det skapats och initierats.

```
const lovelace = {
  firstName: 'Ada',
  lastName: 'Lovelace',
  says (text) {
    return `${this.firstName} says "${text}"` // (A)
  }
}

console.log(lovelace.says('Hi there!')) // OUTPUT: Ada says "Hi there!"
```

dataegenskaper

metod

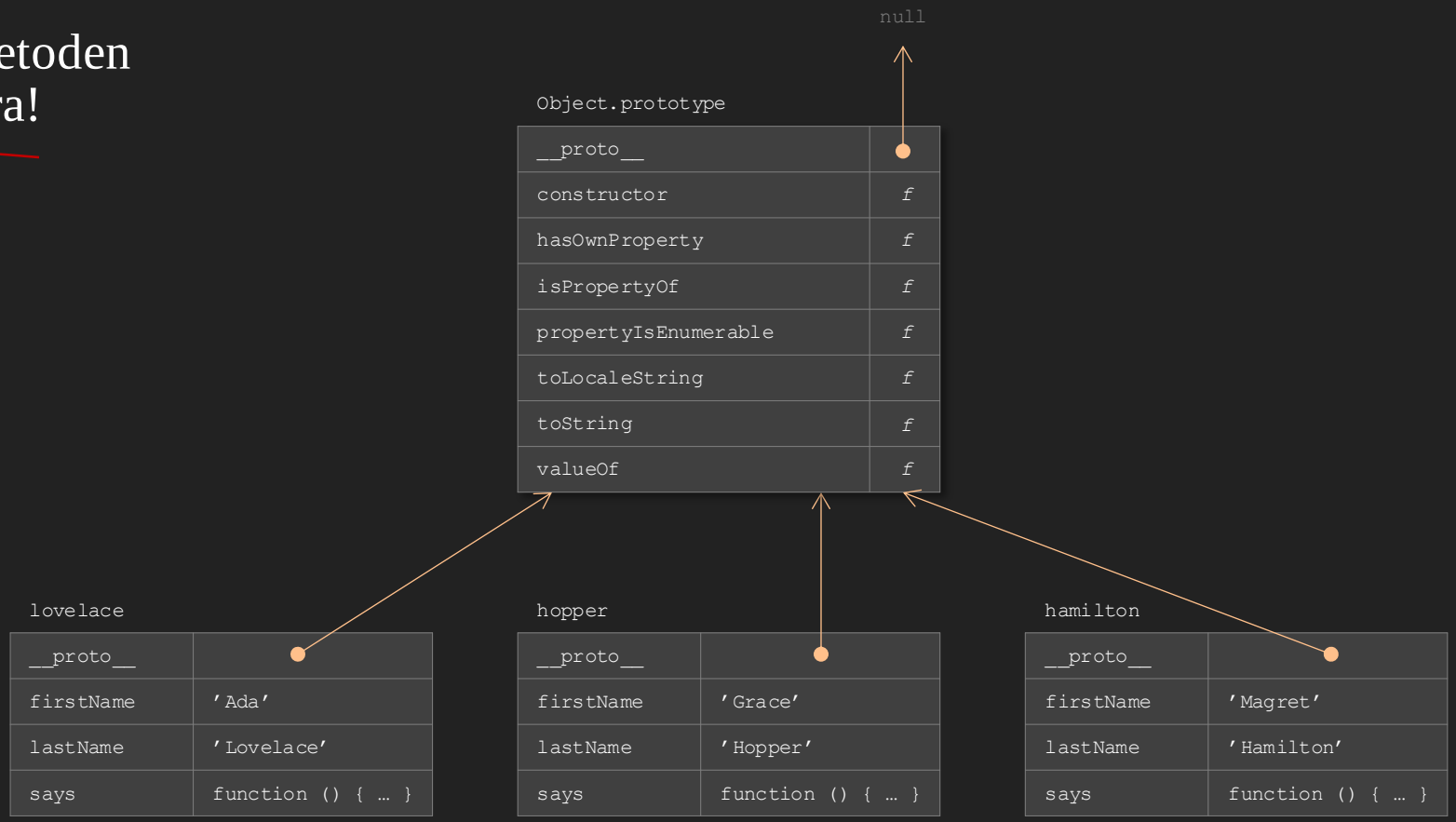
- ✓ Under metodanropet `lovelace.says('Hi there!')`, sägs `lovelace` vara objektet metoden anropas för.
- ✓ `this` sätts att referera till det objekt en metod anropas för. Detta gör det möjligt för metoden `.says()` att genom `this` komma åt egenskapen `.firstName` på rad A.



Flera objekt säger samma sak

- ✓ Tre objekt som samtliga innehåller metoden `.says()` – upprepning av kod, inte bra!

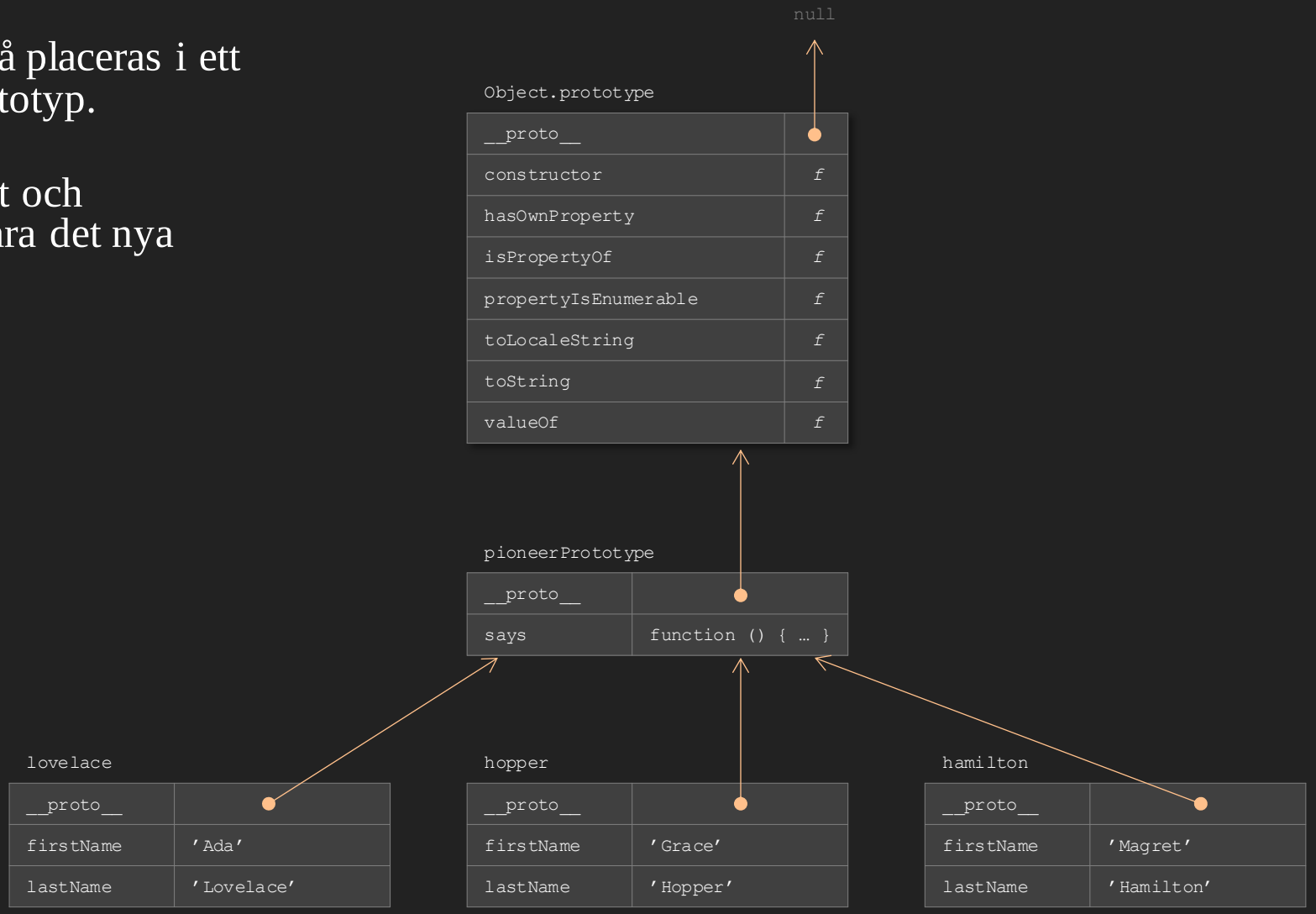
```
const lovelace = {
  firstName: 'Ada',
  lastName: 'Lovelace',
  says (text) {
    return `${this.firstName} says "${text}"`
  }
}

const hopper = {
  firstName: 'Grace',
  lastName: 'Hopper',
  says (text) {
    return `${this.firstName} says "${text}"`
  }
}

const hamilton = {
  firstName: 'Magret',
  lastName: 'Hamilton',
  says (text) {
    return `${this.firstName} says "${text}"`
  }
}
```



En ny egen prototyp

- ✓ Egenskaper som flera objekt kan dela på placeras i ett nytt objekt som sedan används som prototyp.
- ✓ `Object.create()` skapar ett nytt objekt och argumentet ger vilket objekt som ska vara det nya objektets prototyp.

```
const pioneerPrototype = {
  says (text) {
    return `${this.firstName} says "${text}"`
  }
}

const lovelace = Object.create(pioneerPrototype)
lovelace.firstName = 'Ada'
lovelace.lastName = 'Lovelace'

const hopper = Object.create(pioneerPrototype)
hopper.firstName = 'Grace'
hopper.lastName = 'Hopper'

const hamilton = Object.create(pioneerPrototype)
hamilton.firstName = 'Magret'
hamilton.lastName = 'Hamilton'

console.log(lovelace.says('Good day!'))
console.log(hopper.says('Hello'))
console.log(hamilton.says('Hi!'))
```



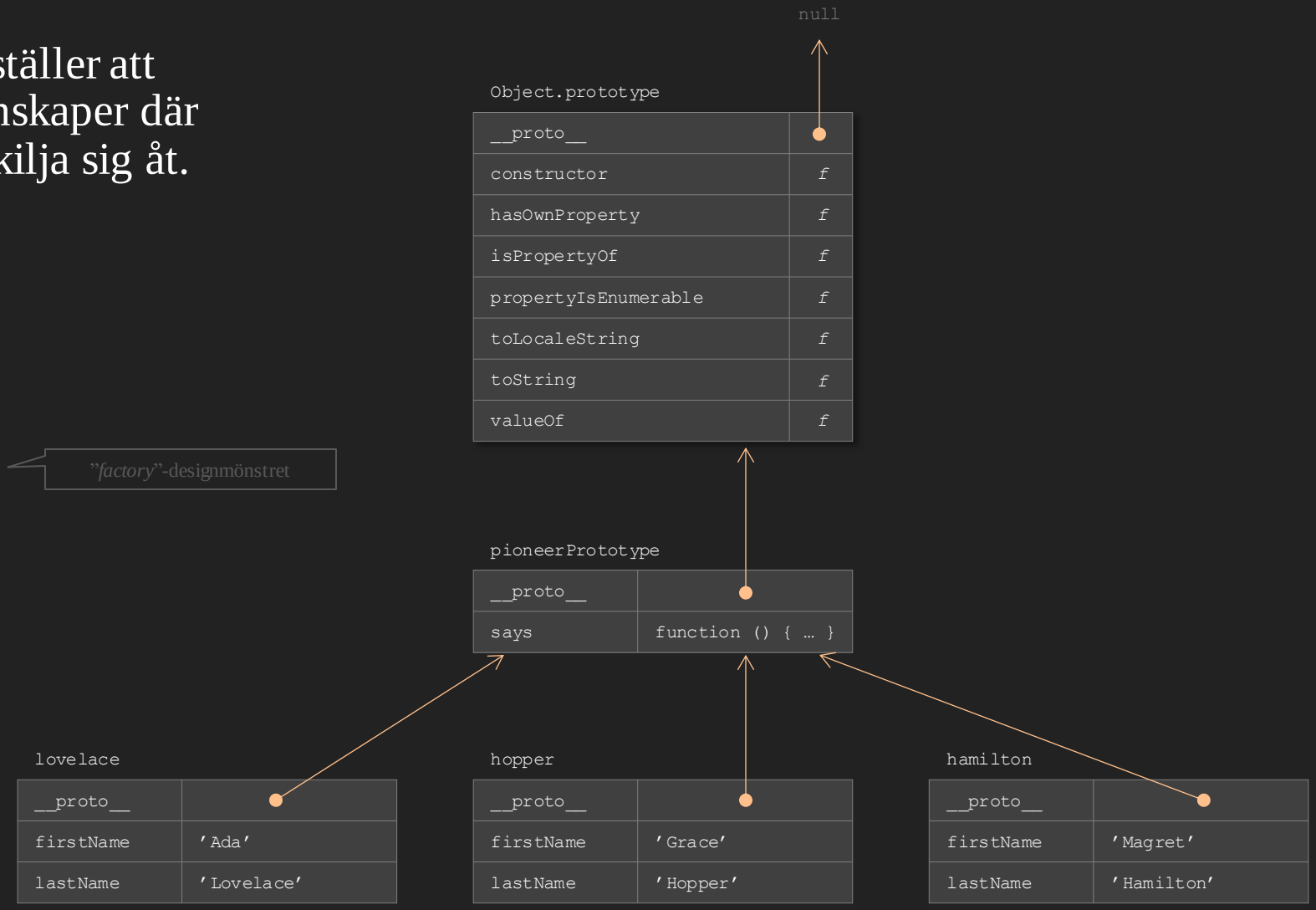
Fabrik för många objekt

”Literal for One, Factories for Many”
Från <https://medium.com/javascript-scene/javascript-factory-functions-with-es6-4d224591a8b1>

- ✓ Funktionen `createPioneer()` säkerställer att returnerade objekt har exakt lika egenskaper där endast dataegenskapers värden kan skilja sig åt.

```
const pioneerPrototype = {
  says (text) {
    return `${this.firstName} says "${text}"`
  }
}

const createPioneer = function (fName, lName) {
  const pioneer = Object.create(pioneerPrototype)
  pioneer.firstName = fName
  pioneer.lastName = lName

  return pioneer
}

const lovelace = createPioneer('Ada', 'Lovelace')
const hopper = createPioneer('Grace', 'Hopper')
const hamilton = createPioneer('Magret', 'Hamilton')

console.log(lovelace.says('Good day!'))
console.log(hopper.says('Hello'))
console.log(hamilton.says('Hi!'))
```

”*factory*”-designmönstret



Vilka egenskaper har ett objekt?

- ✓ Iterera genom ett objekts samtliga egenskapers namn (inklusive dess prototyper utom `Object`) med [`for...in`].

```
for (const key in lovelace) {
   console.log(`${key}: ${lovelace[key]}`)
}
// OUTPUT
// firstName: Ada
// lastName: Lovelace
// says: says (text) {
//    return `${this.firstName} says "${text}"`
//  }
```

- ✓ Iterera igenom ett objekts egna egenskaper med hjälp av metoden [`Object.keys()`], som returnerar en array innehållande objektets egna egenskapers namn.

```
for (const key of Object.keys(lovelace)) {
  console.log(`${key}: ${lovelace[key]}`)
}
// OUTPUT
// firstName: Ada
// lastName: Lovelace
```