Capire JavaScript: aspetti avanzati su funzioni e oggetti

Funzioni e oggetti in JavaScript: aspetti avanzati

Paolo Caramanica

Sommario

- Le funzioni come oggetti
  - Le funzioni in JavaScript
  - NFE
  - NFE - Esempio
  - Live demo
- This
  - This
  - This nelle funzioni
- Approfondimenti sugli oggetti
  - Accessor properties
  - Prorotype
  - Da tipo primitivo a oggetto

Le funzioni come oggetti

Le funzioni come oggetti in JavaScript

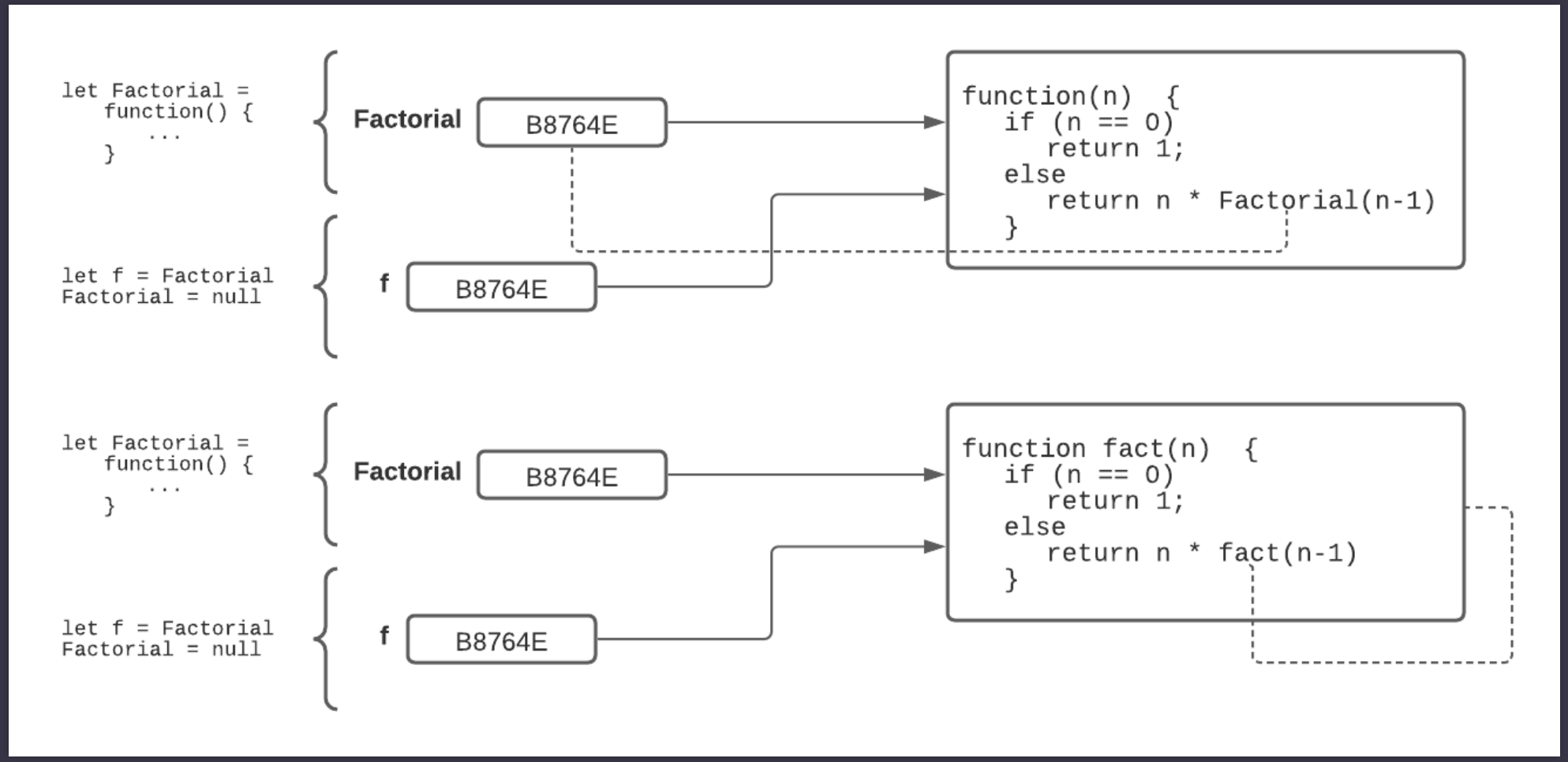
Le funzioni in JavaScript

- In JavaScript, una funzione può essere assegnata ad una variabile, come una stringa o un numero, quindi è un dato di qualche tipo.
- In JavaScript, **le funzioni sono oggetti:**
  - Hanno delle proprietà (name, length).
  - Vi si possono associare delle proprietà custom.

NFE

- Essendo oggetti, le funzioni in JavaScript sono salvate in memoria come gli oggetti, cioè **per riferimento**:
  - Questo, in alcuni casi, può generare dei comportamenti imprevisti.
- Per gestire tali casistiche, possono essere utili le **NFE** (**Named Function Expression**), che permettono di assegnare un secondo nome alle funzioni:
  - Utilizzabile internamente per referenziare la funzione stessa.
  - Non visibile esternamente.

NFE - Esempio


let Factorial =
function() {
...
}
Factorial
B8764E
function(n) {
if (n == 0)
return 1;
else
return n * Factorial(n-1)
}
let f = Factorial
Factorial = null
f
B8764E
let Factorial =
function() {
...
}
Factorial
B8764E
function fact(n) {
if (n == 0)
return 1;
else
return n * fact(n-1)
}
let f = Factorial
Factorial = null
f
B8764E

Live demo

This

Il funzionamento di this in JavaScript

This

- In Javascript, il this permette a un metodo di un oggetto di accedere all'oggetto stesso.
- A differenza di altri linguaggi, this è **unbound** (non associato):
  - Non referenzia necessariamente l'oggetto nel quale il metodo è stato creato.
  - Se il metodo viene passato come argomento ad una funzione, il this viene perso (**losing this**)
- Per capire quale oggetto rappresenta il this in un metodo, non è importante dove quest'ultimo è stato definito, ma dove viene chiamato.

This nelle funzioni

- Se una funzione viene definita all'esterno di un oggetto, il this rappresenta il Global Object (window, nei browser) oppure undefined.
- Se all'interno di un metodo viene definita una funzione (che non sia una arrow function), in essa il this non rappresenta l'oggetto a cui appartiene il metodo, ma il Global Object.
- Le arrow function non hanno this e prendono quello dello scope più esterno: se una arrow function è definita all'interno del metodo di un oggetto, il this sarà l'oggetto stesso.

Live demo

Approfondimenti sugli oggetti

Accessor, Prototype e conversione in oggetti

Accessor properties

- Le **accessor properties** sono metodi speciali di un oggetto JavaScript, che esternamente appaiono come delle proprietà, alle quali si può accedere in lettura e/o in scrittura:
  - **Getter**: per l'accesso in lettura
    `get prop() { … }`
  - **Setter**: per l'accesso in scrittura
    `set prop(val) { … }`
- Possono essere richiamate come delle normali proprietà:

```
let a = obj.prop; // in lettura
obj.prop = 15; // in scrittura
```

- Sono utili per aggiungere della logica nella lettura e scrittura delle proprietà.

Prototype

- Talvolta è necessario creare un oggetto a partire da un altro, che **erediti** le proprietà e i metodi di quest'ultimo e ne aggiunga di propri.
- In JavaScript, si può raggiungere questo scopo attraverso una proprietà nascosta che hanno tutti gli oggetti, definita [[**Prototype**]]:
  - Può contenere null o un riferimento ad un altro oggetto.
  - Se si cerca di accedere ad una proprietà e questa non viene trovata nell'oggetto, viene cercata nell'oggetto referenziato da [[Prototype]].
  - E' accessibile attraverso __proto__

Da tipo primitivo a oggetto

- JavaScript permette di operare con i tipi primitivi (stringhe, numeri) come se fossero oggetti.
  - Ad esempio, su una variabile s di tipo stringa, può essere richiamata la proprietà length, che ne contiene la lunghezza (s.length), o il metodo toUpperCase, che la restituisce in maiuscolo (s.toUpperCase()).
- Per far questo, JavaScript effettua in automatico il **wrapping** della variabile di tipo primitivo in un oggetto, che fornisce proprietà e metodi utili.

Contatti

Email: paolocaramanica@gmail.com

Linkedin: https://it.linkedin.com/in/paolo-caramanica-436942a/it-it

Facebook: https://it-it.facebook.com/paolo.caramanica

Twitter: https://twitter.com/PaoloCaramanica

Sito web: http://www.paolocaramanica.net

Q&A