

Function - Parte 2

Invocando uma função com
call e **apply**

call e apply

Toda função possui as operações `call()` e `apply()`. Eles são utilizados para indicar em qual escopo uma função deve ser executada.

A diferença é basicamente a forma como é utilizado:

```
fn.call(this, arguments1, arguments2)
```

```
fn.call(this, arguments)
```

```
1. let getTitle = function () {  
2.   return this.title;  
3. };  
4. let book = {  
5.   title: "Refactoring",  
6.   author: "Martin Fowler",  
7.   pages: 342,  
8.   getTitle: getTitle  
9. };  
10. getTitle.call(book);  
11. getTitle.apply(book);
```

Invocando uma função com **call** e **apply**

Invocando uma função por meio do
operador **new**

```
1. var cleanCode = {  
2.   title: "Clean Code",  
3.   pages: 240,  
4.   author: "Robert C. Martin",  
5.   category: "Programming"  
6. };  
  
8. var refactoring = {  
9.   title: "Refactoring",  
10.  pages: 342,  
11.  author: "Martin Fowler",  
12.  category: "Programming"  
13. };
```


Criando objetos **diretamente**


```
1. var createProgrammingBook = function (title, pages, author) {  
2.   return {title, pages, author, category: "Programming"};  
3. };  
4.  
5. var cleanCode = createProgrammingBook("Clean Code, 240, "Robert  
   C. Martin");  
6. var refactoring = createProgrammingBook("Refactoring", 342, "Martin  
   Fowler");
```


Criando objetos com uma
função fábrica

```
1. var ProgrammingBook = function (title, pages, author) {  
2.     this.title = title;  
3.     this.pages = pages;  
4.     this.author = author;  
5.     this.category = "Programming";  
6. };  
7.  
8. var cleanCode = new ProgrammingBook("Clean Code", 240, "Robert  
C. Martin");  
9. var refactoring = new ProgrammingBook("Refactoring", 342, "Martin  
Fowler");
```

Criando objetos com uma
função construtora

cleanCode	
title	Clean Code
pages	240
author	Robert Martin
category	Programming
[[Prototype]]	

refactoring	
title	Refactoring
pages	342
author	Martin Fowler
category	Programming
[[Prototype]]	


ProgrammingBook.prototype	
[[Prototype]]	


Object.prototype	
[[Prototype]]	null




```
1. var ProgrammingBook = function (title, pages, author) {  
2.     this.title = title;  
3.     this.pages = pages;  
4.     this.author = author;  
5. };  
6. ProgrammingBook.prototype.category = "Programming";  
7.  
8. var cleanCode = new ProgrammingBook("Clean Code", 240, "Robert  
   C. Martin");  
9. var refactoring = new ProgrammingBook("Refactoring", 342, "Martin  
   Fowler");
```

Criando objetos com uma
função construtora

cleanCode	
title	Clean Code
pages	240
author	Robert Martin
[[Prototype]]	

refactoring	
title	Refactoring
pages	342
author	Martin Fowler
[[Prototype]]	

ProgrammingBook.prototype	
category	Programming
[[Prototype]]	

Object.prototype	
[[Prototype]]	null





Qual é a diferença entre
[[Prototype]] e prototype?


```
1. var _new = function(f) {  
2.   var res = {};  
3.   if (f.prototype !== null) {  
4.     res.__proto__ = f.prototype;  
5.   }  
6.   var ret = f.apply(res, Array.prototype.slice.call(arguments, 1));  
7.   if ((typeof ret === "object" || typeof ret === "function") && ret !== null) {  
8.     return ret;  
9.   }  
10.  return res;  
11. };
```

new

cleanCode	
title	Clean Code
pages	240
author	Robert Martin
[[Prototype]]	

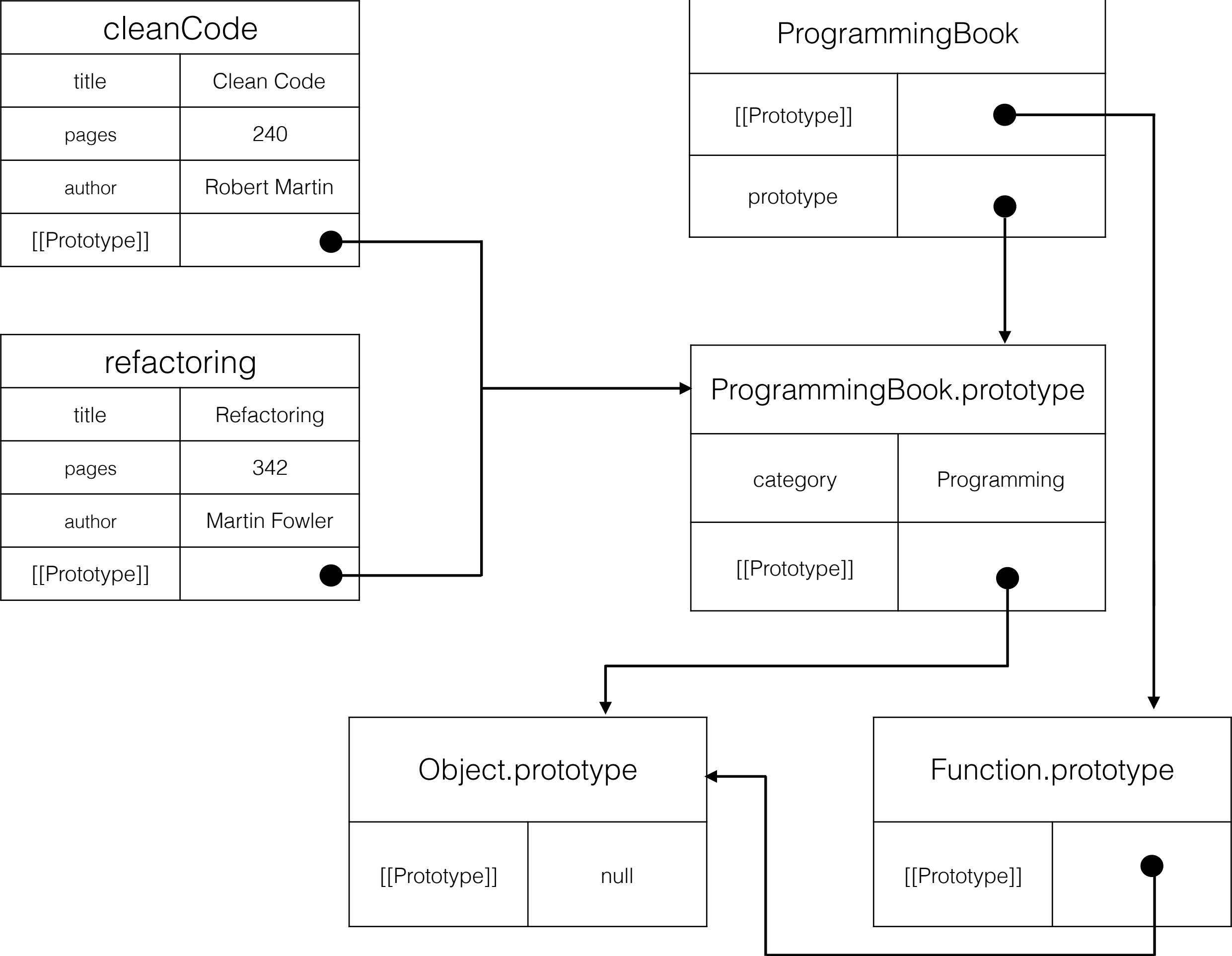
refactoring	
title	Refactoring
pages	342
author	Martin Fowler
[[Prototype]]	

ProgrammingBook	
[[Prototype]]	
prototype	

ProgrammingBook.prototype	
category	Programming
[[Prototype]]	

Object.prototype	
[[Prototype]]	null

Function.prototype	
[[Prototype]]	





Não esqueça de utilizar o operador **new**
quando utilizar funções construtoras



Qual é a diferença entre o
instanceOf e o typeof?



Executando um bloco de código
com **eval**

A função eval executa um bloco de código representado por meio de String



Utilize eval apenas se não existir
outra alternativa


```
1. var printHelloWorld = "console.log('Hello World!');";
```

Executando um bloco de código
com a função **eval**

1. `var printHelloWorld = "console.log('Hello World!');";`
2. `eval(printHelloWorld);`

Executando um bloco de código
com a função **eval**