Detalhes do modelo de objeto

Samuel Amaro

4 de junho de 2021

Detalhes do modelo de objeto

JavaScript é uma linguagem orientada a objetos com base em protótipos, em vez de ser baseada em classes. Devido a essa base diferente, pode ser menos evidente como o JavaScript permite criar hierarquias de objetos e ter herança de propriedades e seus valores. Este capítulo tenta esclarecer essa situação.

Este capítulo assume que você já está um pouco familiarizado com JavaScript e que você já tenha usado funções JavaScript para criar simples objetos.

Linguages baseadas em classe vs. baseada em protótipo

Linguagens orientadas a objetos baseadas em classe, como Java e C + +, são fundadas no conceito de duas entidades distintas: classes e instâncias.

- Uma classe define todas as propriedades (considerando-se os métodos e campos em Java, ou membros em C++, para ser propriedades) que caracterizam um determinado conjunto de objetos. Uma classe é algo abstrato, ao invés de qualquer membro particular do conjunto de objetos que descreve. Por exemplo, a classe `Employee` poderia representar o conjunto de todos os funcionários.
- Uma instância, por outro lado, é a instanciação de uma classe, ou seja, um dos seus membros. Por exemplo, `Victoria` poderia ser uma instância da classe `Employee`, o que representa um indivíduo em particular como um empregado. Uma instância tem exatamente as propriedades de sua classe pai (nem mais, nem menos).

Uma linguagem baseada em protótipo, como JavaScript, não faz essa distinção: ele simplesmente tem objetos. Uma linguagem baseada em protótipo tem a idéia de um objeto prototípico, um objeto usado como um modelo do qual obtém as propriedades iniciais para um novo objeto. Qualquer objeto pode especificar suas próprias propriedades, quando você o cria ou em tempo de execução. Além disso, qualquer objeto pode ser associado como um protótipo de outro objeto, permitindo ao segundo objeto compartilhar as propriedades do primeiro objeto.

Definindo uma classe

Em linguagens baseadas em classe, você define uma classe em uma definição de classe separada. Nessa definição, você pode especificar métodos especiais, chamados de construtores, para criar instâncias da classe. Um método construtor pode especificar valores iniciais para as propriedades da instância e executar outros processamentos apropriados no momento da criação. Você pode usar o operador `new`, em associação com o método construtor para criar instâncias de classe.

O JavaScript segue um modelo semelhante, mas não têm uma definição da classe separada do construtor. Em vez disso, você define uma função de construtor para criar objetos com um conjunto inicial particular de propriedades e valores. Qualquer função JavaScript pode ser usado como um construtor. Você pode usar o operador `new` com uma função de construtor para criar um novo objeto.

Subclasses e Herança

Em uma linguagem baseada em classe, você cria a hierárquia de classes através de sua definição. Em uma definição de classes, você pode especificar que a nova classe é uma subclasse de outra já existente. A subclasse herda todas as propriedades da superclasse e pode adicionar novas propriedades ou modificar propriedades herdadas. Por exemplo, assuma que a classe `Employee` tem somente duas propriedades `name` e `dept` , e `Manager` é uma subclasse of `Employee` que adiciona a propriedade `reports`. Neste caso, uma instância da classe Manager terá todas as três propiedades: `name`, `dept`, and `reports`.

Em JavaScript, a herança é implementada associando um objeto prototípico a qualquer função de construtor. Então, você pode criar exatamente o mesmo exemplo: `Employee` — `Manager`, mas utilizando uma terminologia ligeramente diferente. Primeiro, define-se a função de construtor de `Employee`, especificando as propriedades `name` e `dept`. Depois, define-se a função de construtor de `Manager`, especificando a propriedade `reports`. Finalmente, associa-se

um objeto `new Employee` como `prototype` para a função de construtor `Manager`. Então, quando voc criar um objeto `new Manager`, ele herdará as propriedades `name` e `dept` do objeto `Employee`.

Adicionando e removendo propriedades

Em uma linguagem baseada em classe, você normalmente cria uma classe em tempo de compilação e então vincula as instâncias da classe em tempo de compilação, ou tempo de execução. Você não pode alterar o número ou o tipo de propriedade de uma classe após definí-la. Em javaScript, no entanto, você pode adicionar ou remover propriedades de qualquer objeto. Se você adiciona uma propriedade a um objeto que é usado como o protótipo para um conjunto de objetos, os objetos no qual ele é protótipo herdarão as novas propriedades.

Sumário das diferenças

A tabela a seguir apresenta um breve resumo de algumas dessas diferenças. O restante deste capítulo descreve os detalhes do uso de construtores e protótipos JavaScript para criar uma hierarquia de objetos e compara isso à maneira como você faria em Java.

Comparação de objetos do sistema baseados em classes (Java) e baseado em protótipo (JavaScript)

Baseados em classes (Java)	**Baseados em protótipos (JavaScript)**
Define uma classe com uma definição de classe; cria um objeto - como instância da classe	Define e cria um conjunto de objetos com funções construtoras.
Cria um único objeto com o operador new.	Faz o mesmo.
Constroi uma hierarquia de objetos usando definição de classe para definir subclasses de classes existentes.	Constrói uma hierarquia de objetos atribuindo um objeto como o protótipo associado com uma função de construtor.
Herda propriedade seguindo a cadeia de classe.	Herda propriedade seguindo a cadeia de protótipo.
Definição de classe especifica todas as propriedades de todas as instâncias de uma classe. Não é possível adicionar propriedades dinamicamente em tempo de execução.	Função construtor ou protótipo especifica um conjunto inicial de propriedades. Pode adicionar ou remover propriedades de forma dinâmica para objetos individuais ou para todo o conjunto de objetos.