Aula Introdução a objetos em Javascript

Samuel Amaro

24 de abril de 2021

Anotações importantes sobre a aula

Em JavaScript, quase tudo é objeto. Desde as características padrão, como strings e arrays, até as APIs para navegadores baseados na linguagem. Até você pode criar seus objetos próprios. Você pode encapsular funções e variáveis relacionadas de uma forma eficiente. Os objetos agem como uma espécie de manipuladores de dados. Compreender a natureza da orientação a objetos do JavaScript é crucial para aprofundar os conhecimentos acerca da linguagem. Por isso, nós elaboramos esse módulo para auxiliá-lo. Ensinaremos a teoria de objetos em detalhes. Crie seus objetos próprios!

Programação orientada a objetos - o básico

Com o básico fora do caminho, agora vamos nos concentrar no JavaScript orientado a objetos (OOJS) — Este artigo apresenta uma visão básica da teoria de programação orientada a objeto (OOP), em seguida, explora como o JavaScript emula as classes de objetos através de funções de construtor e como criar instâncias de objeto.

Para começar, vamos dar uma visão simplista e de alto nível do que é programação orientada a objeto (OOP). Dizemos simplista, porque a OOP pode rapidamente se tornar muito complicada, e dar a ela um tratamento completo agora provavelmente confundiria mais do que ajuda. A idéia básica da OOP é que usamos objetos para modelar coisas do mundo real que queremos representar dentro de nossos programas, e / ou fornecer uma maneira simples de acessar funcionalidades que de outra forma seriam difíceis ou impossíveis de usar.

Os objetos podem conter dados e códigos relacionados, que representam informações sobre o que você está tentando modelar e a funcionalidade ou

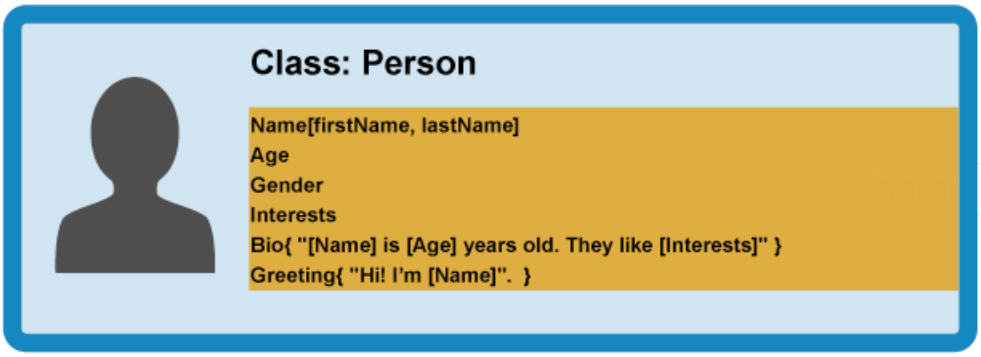

Figura 1: Diagrama de class de Pessoa

o comportamento que você deseja ter. Dados de objeto (e muitas vezes, funções também) podem ser armazenados ordenadamente (a palavra oficial é encapsulados) dentro de um pacote de objetos (que pode ser dado um nome específico para se referir, que é às vezes chamado de namespace), tornando fácil de estruturar e acessar; objetos também são comumente usados como armazenamentos de dados que podem ser facilmente enviados pela rede.

Definindo um modelo de objeto

Vamos considerar um programa simples que exibe informações sobre os alunos e professores de uma escola. Aqui vamos olhar para a teoria OOP em geral, não no contexto de qualquer linguagem de programação específica.

Para começar, poderíamos retornar ao nosso tipo de objeto Person do nosso primeiro artigo de objetos, que define os dados genéricos e a funcionalidade de uma pessoa. Há muitas coisas que você poderia saber sobre uma pessoa (endereço, altura, tamanho do sapato, perfil de DNA, número de passaporte, traços de personalidade significativos ...), mas neste caso estamos interessados apenas em mostrar seu nome, idade, sexo e interesses, e também queremos ser capazes de escrever uma breve introdução sobre eles com base nesses dados e fazê-los dizer oi. Isso é conhecido como **abstração** — criando um modelo simples de uma coisa mais complexa, que representa seus aspectos mais importantes de uma forma que é fácil trabalhar com os objetivos do nosso programa.

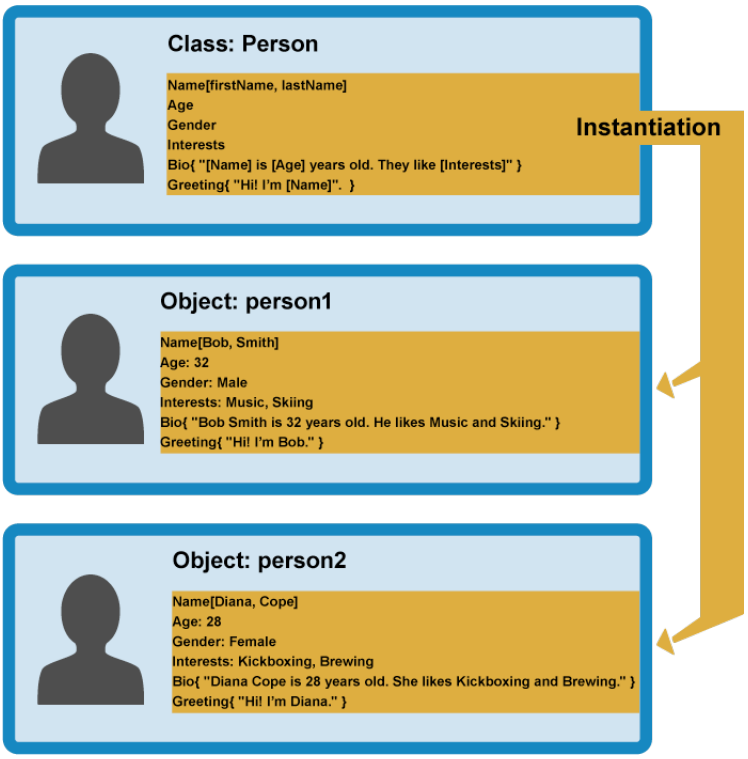

Figura 2: Graficos demostrando a instacia de obejtos da class person

Criando objetos reais

De nossa classe, podemos criar **instâncias de objeto** — objetos que contêm os dados e a funcionalidade definidos na classe. Da nossa classe Person, podemos criar algumas pessoas reais:

Quando uma instância de objeto é criada a partir de uma classe, a **função construtora** da classe é executada para criá-la. Esse processo de criação de uma instância de objeto de uma classe é chamado de **instanciação** — a instância do objeto é **instanciada** a partir da classe.

Classes especialistas(especializadas)

Neste caso, não queremos pessoas genéricas — queremos professores e alunos, que são tipos mais específicos de pessoas. Em OOP, podemos criar novas classes com base em outras classes — essas novas **classes filhas** podem **herdar** os recursos de dados e código de sua **classe pai**, para que você possa reutilizar a

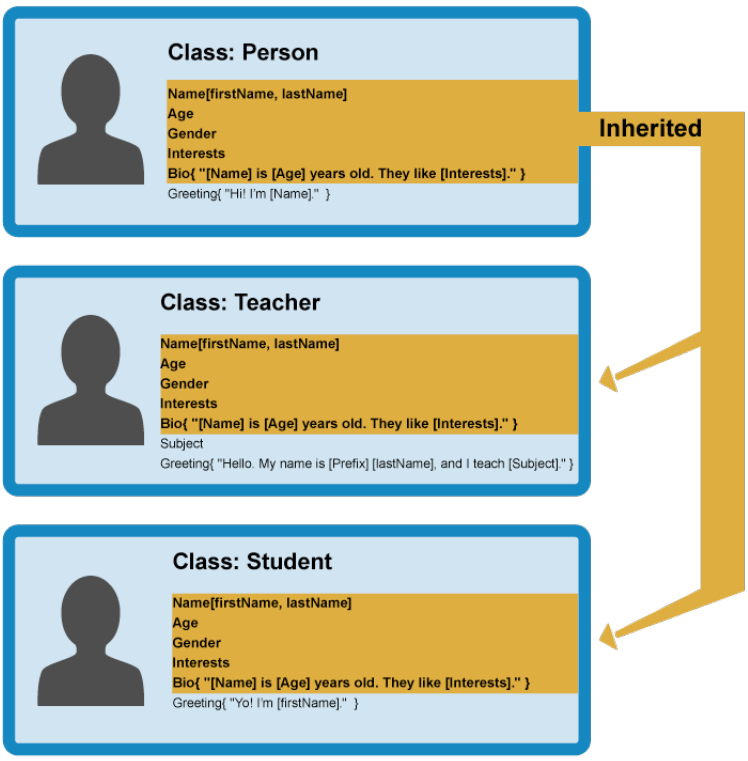

Figura 3: Herança

funcionalidade comum a todos os tipos de objetos em vez de duplicá-los. Onde a funcionalidade difere entre as classes, você pode definir recursos especializados diretamente sobre eles, conforme necessário.

Isso é realmente útil — professores e alunos compartilham muitos recursos comuns, como nome, sexo e idade, por isso é conveniente definir apenas esses recursos uma vez. Você também pode definir o mesmo recurso separadamente em classes diferentes, já que cada definição desse recurso estará em um namespace diferente. Por exemplo, a saudação de um aluno pode estar no formato "Yo, I'm [firstName]"(por exemplo, Yo, I'm Sam), enquanto um professor pode usar algo mais formal, como "Olá, meu nome é [Prefixo [lastName], e eu ensino [Subject]. "(por exemplo Olá, Meu nome é Mr Griffiths, e eu ensino Química).

Nota: A palavra chique para a capacidade de múltiplos tipos de objeto de implementar a mesma funcionalidade é o **polimorfismo**. Apenas no caso de você estar se perguntando.

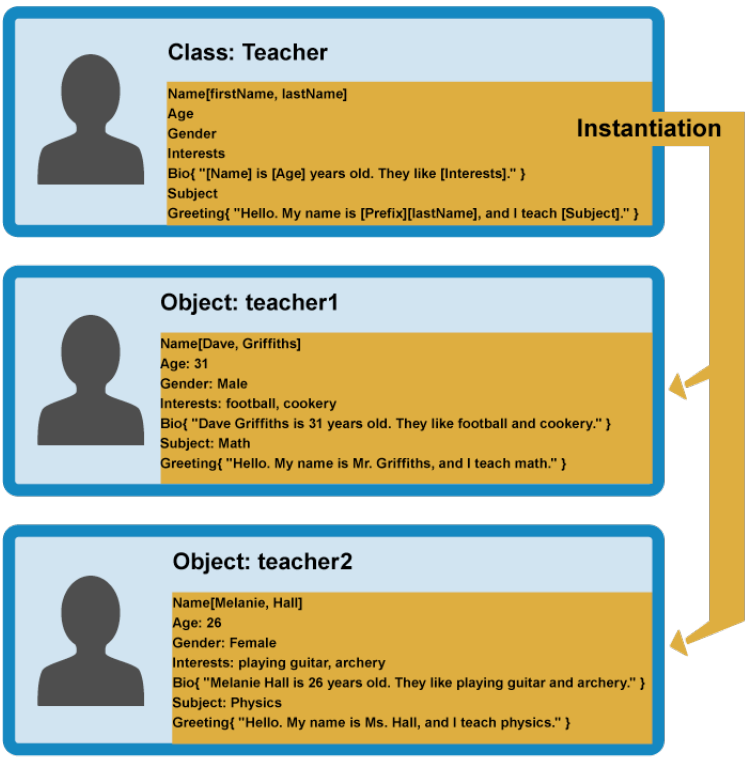

Figura 4: Instancias da classe professor

Agora você pode criar instâncias de objetos de suas classes filhas. Por exemplo:

No restante do artigo, começaremos a analisar como a teoria da POO pode ser colocada em prática no JavaScript.