14章 クラスを理解しよう

プログラミングにおけるクラスとは何かを解説します。

120分 - 読了

14.1 本章の目標

本章では以下を目標にして学習します。

- クラスを理解し、似たようなオブジェクトを大量生産できるようになること
- クラスをより柔軟に使いこなせるようになること

例えばAmazonのようなショッピングサイトにおいて大量の商品データを作るとき、以下のようにゼロから個別のオブジェクトを作るのは極めて大変です（オブジェクトについてはこのあと復習します）。

```
const shampoo = { name: 'シャンプー', price: 500, category: '生活雑貨' };
const coffee = { name: 'コーヒー', price: 1500, category: '飲料' };
const headphone = { name: 'ヘッドホン', price: 6000, category: 'オーディオ家電' };

```

そこで使うのが**クラス**です。クラスを使えば、このような類似するオブジェクトを大量に作れるようになります。

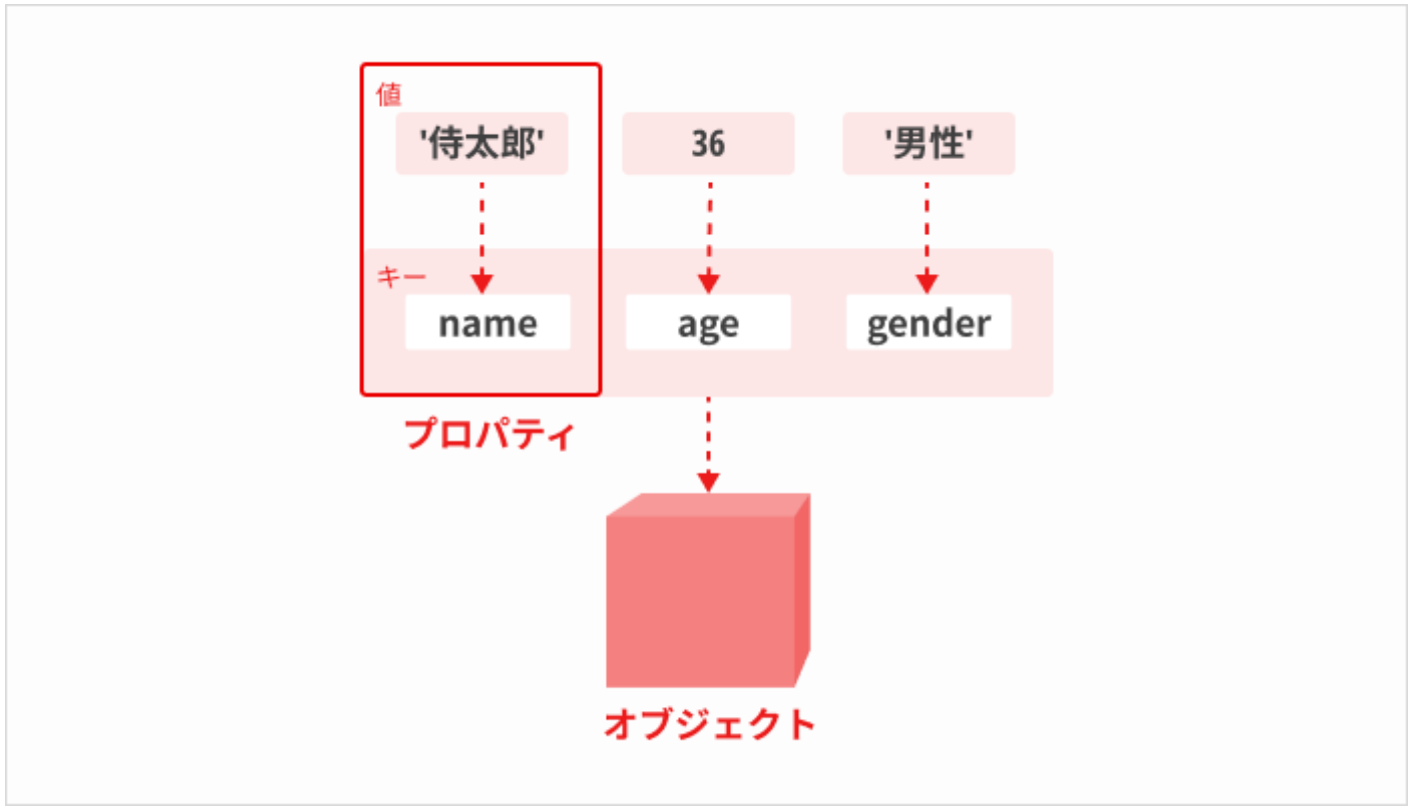
では本章も頑張っていきましょう。

14.2 オブジェクトについて復習しよう

クラスを理解するにはオブジェクトの知識が不可欠です。そこで、10章で学んだオブジェクトについてもう一度復習しておきましょう。

オブジェクトとは、配列におけるインデックスの代わりに**キー**と呼ばれるラベルをつけて管理するデータの集まりのことでした。

- キー＝値が何を表すのかわかりやすく名前をつけたもの



オブジェクトを作るには以下のように、 {} の中にカンマ区切りでプロパティ（キーと値のセット）を入れていきます。また、キーと値の間にはコロン：を記述します。

JSファイル（見本）

```
const personalData = { name: '侍太郎', age: 36, gender: '男性' };

```

オブジェクトの値を取得・更新・追加する方法は、以下の2通りあることも学びました（本教材ではドット記法を採用）。

1. ブラケット記法： `オブジェクト名['キー名']` （例： `personalData['gender']` ）
2. ドット記法： `オブジェクト名.キー名` （例： `personalData.gender` ）

14.3 クラスとは

クラスとは一言でいえば、**モノ（オブジェクト）の設計図**です。

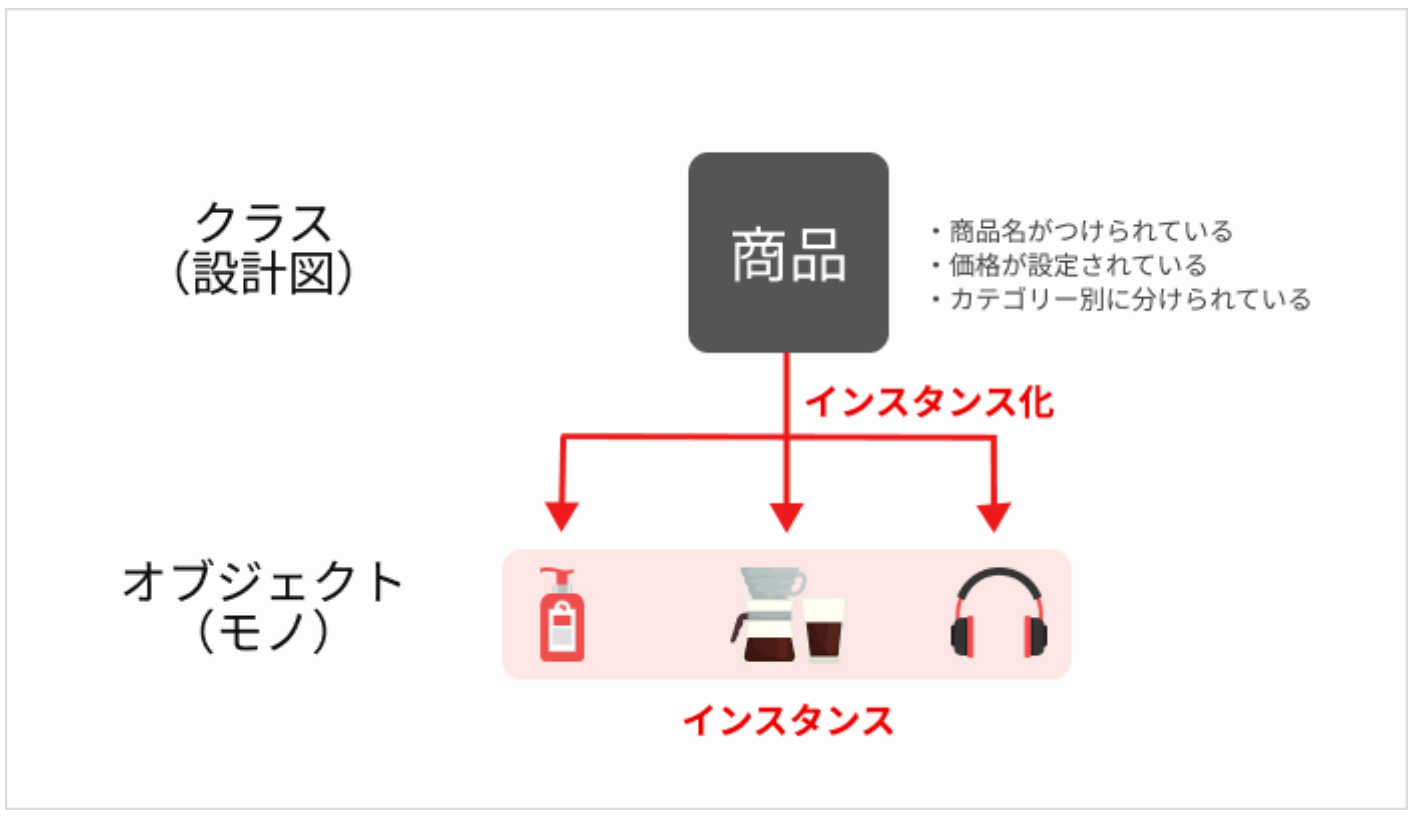
Amazonのようなショッピングサイトで売られている「商品」を例にして考えてみましょう。

商品には「商品名がついている」「価格が設定されている」「カテゴリー別に分けられている」など、どの商品にも共通する特徴があります。このような「**共通する特徴**」を設計図にしたのがクラスです。

一度クラス（設計図）を作ってしまえば、同じ特徴を持った別個のオブジェクト（モノ）を大量生産できるようになるため、とても便利です。

なお、クラスをもとに作られたオブジェクトのことを、特別に**インスタンス**と呼びます。また、クラスをもとにインスタンスを作ることを**インスタンス化**といいます。なお、インスタンスは「実体」という意味です。

- インスタンス＝クラスをもとに作られたオブジェクトのこと（インスタンス＝オブジェクトという認識でOK）



クラスの作り方

クラスの作り方は以下のとおりです。

```
class クラス名 {
  クラスの特徴
}

```

例えば商品（ `Product` ）というクラスであれば、以下のように記述します。なお、クラス名は慣習的に先頭を大文字にします。覚えておきましょう。

JSファイル（見本）

```
class Product {
  クラスの特徴
}

```

クラスの中身の書き方については後ほど学習します。

インスタンス化する方法

クラスをもとにオブジェクトを作る、つまりインスタンス化するには以下のように `new クラス名()` と記述します。 `new` はインスタンス化するときのお作法です。あとで実際にコードを書くので、現時点ではイメージだけつかんでおきましょう。

JSファイル（見本）

```
// クラスを定義する
class Product {
  クラスの特徴
}

// インスタンス化する
const shampoo = new Product();

```

クラスをもとに作られるインスタンスはオブジェクトなので、オブジェクトと同じように `const` を使うのが一般的です（詳細は10章の最後を参照）。

補足：クラスはバージョンES6以降で追加された新機能である

クラスはPHPやRuby、Pythonといったプログラミング言語にはもともと存在している機能ですが、実はJavaScriptにはこのクラスという機能はありませんでした。

そのため、これまでJavaScriptでは、関数を使って「クラスのような機能」を作成していました。

そのような中、バージョンES6以降ではJavaScriptにも新機能としてクラスが追加され、より簡単にクラスを作成できるようになりました。

14.4 コンストラクタとは

コンストラクタとは、クラスをもとにオブジェクトを作る（インスタンス化する）際に処理を行う関数のことです。コンストラクタによって実行される最初の処理のことを、**初期化**といいます。

なお、コンストラクタを英語表記にするとconstructorで、「建設者」という意味があります。

コンストラクタを使うことで、例えば「商品を作ると同時に出品する」といった最初の処理を設定できます。

コンストラクタを使うには、以下のように記述します。

JSファイル（見本）

```
class Product {
  constructor() {
    初期化の内容
  }
}

```

コンストラクタで処理を行う

例えば以下のように記述すると、インスタンス化したときに `constructor()` 内の処理が実行されます。

JSファイル（見本）

```
class Product {
  constructor() {
    console.log('敏感肌にも優しい100%天然由来のシャンプーです。');
  }
}

```

実際にやってみましょう。まずはVisual Studio Codeを開き、 `js` フォルダ内に新しく `class.js` というファイルを作成してください。

続いて、 `class.js` を以下のように編集しましょう。

class.js

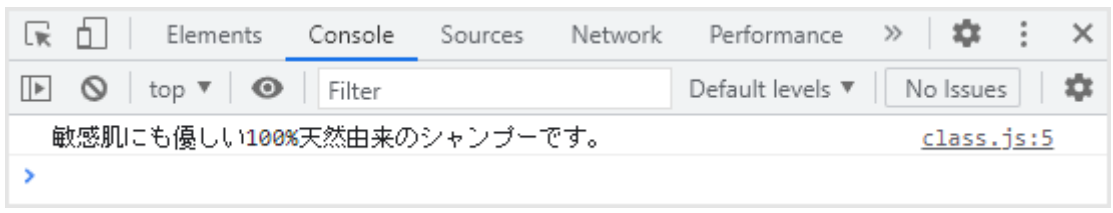
```
// クラスを定義する
class Product {
  // インスタンス化すると同時に処理を実行（初期化）する
  constructor() {
    console.log('敏感肌にも優しい100%天然由来のシャンプーです。');
  }
}

// インスタンス化する
const shampoo = new Product();

```

次に `index.html` を以下のように編集し、読み込むJSファイルを `class.js` に変更してください。

index.html

```
<!DOCTYPE html>
<html lang="ja">

<head>
  <meta charset="UTF-8">
  <title>JavaScript基礎編</title>
</head>

<body>
-   <script src="js/function.js"></script>
+   <script src="js/class.js"></script>
</body>

</html>

```

では `index.html` をブラウザで開き、デベロッパーツールのコンソールを確認してみましょう。以下のように、インスタンス化と同時に処理が実行されていればOKです。



コンストラクタでプロパティを持たせる（引数なしバージョン）

クラスをもとに作られるのはオブジェクトなので、当然プロパティ（キーと値のセット）を持つことができます。

例として、インスタンス化するときに以下のプロパティを持たせてみましょう。

```
{ name: 'シャンプー', price: 500, category: '生活雑貨' }

```

インスタンス化するときに上記のプロパティを持たせるには、コンストラクタ内に以下のように記述します。

JSファイル（見本）

```
class Product {
  constructor() {
    this.name = 'シャンプー';
    this.price = 500;
    this.category = '生活雑貨';
  }
}

```

なお、 `this` はインスタンス化されたオブジェクト自身を意味します。ドット記法では `オブジェクト名.キー名` （例： `personalData.gender` ）で値の取得・更新・追加ができますが、この「オブジェクト名」の部分を `this` に置き換えているイメージです。

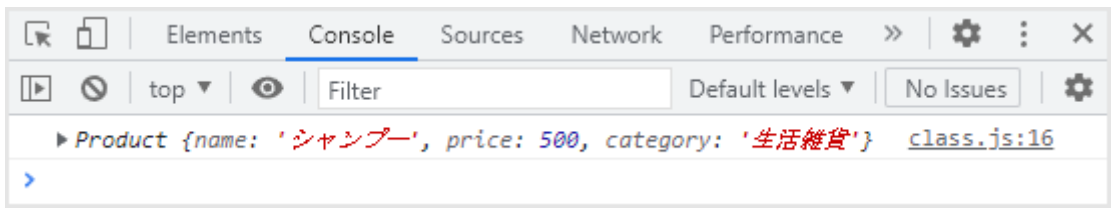
クラスという設計図を作っている時点ではオブジェクト名は決まっていませんが、 `this` と書くことでどのようなオブジェクト名にも対応できます。

では実際にやってみましょう。 `class.js` を以下のように編集してください。

class.js

```
// クラスを定義する
class Product {
  // コンストラクタを使ってインスタンス化するときに初期化する
  constructor() {
-     console.log('敏感肌にも優しい100%天然由来のシャンプーです。')
+     // インスタンス（オブジェクト）にプロパティを持たせる
+     this.name = 'シャンプー';
+     this.price = 500;
+     this.category = '生活雑貨';
  }
}

// インスタンス化する
const shampoo = new Product();

+ // インスタンス（オブジェクト）の値を出力する
+ console.log(shampoo);

```

続いて index.html をブラウザで開き、デベロッパーツールのコンソールを確認してみましょう。以下のように、コンストラクタ内で持たせたプロパティが出力されていればOKです。



インスタンス化によってオブジェクトが作られることが、これでおわかりいただけたかと思います。

コンストラクタでプロパティを持たせる（引数ありバージョン）

先ほどの方法では、どのインスタンス（オブジェクト）も同じ値になってしまいます。

- インスタンス1： { name: 'シャンプー', price: 500, category: '生活雑貨' }
- インスタンス2： { name: 'シャンプー', price: 500, category: '生活雑貨' }
- インスタンス3： { name: 'シャンプー', price: 500, category: '生活雑貨' }

インスタンスごとに値を変えたい場合は、関数でも登場した引数を使います。以下のように記述することで、インスタンス化するときに引数として渡した値をそれぞれのキーに代入できます。

JSファイル（見本）

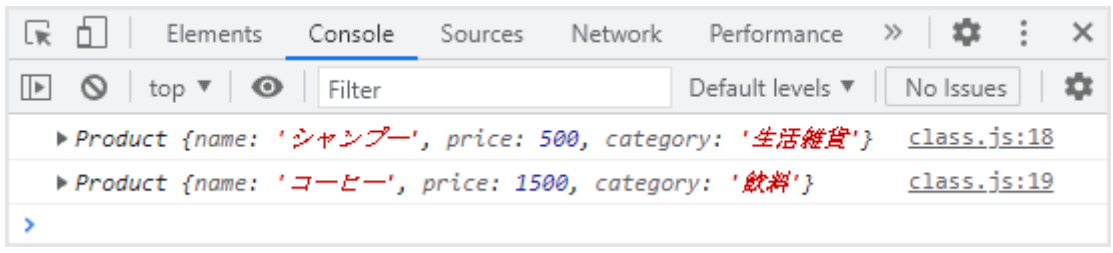
```
class Product {
  constructor(name, price, category) {
    this.name = name;
    this.price = price;
    this.category = category;
  }
}

// インスタンス化するときに引数を渡す
const shampoo = new Product('シャンプー', 500, '生活雑貨');

```

実際にやってみましょう。 `class.js` を以下のように編集してください。

class.js

```
// クラスを定義する
class Product {
  // コンストラクタを使ってインスタンス化するときに初期化する
-  constructor() {
+  constructor(name, price, category) {
    // インスタンス（オブジェクト）にプロパティを持たせる
-    this.name = 'シャンプー';
-    this.price = 500;
-    this.category = '生活雑貨';
+    this.name = name;
+    this.price = price;
+    this.category = category;
  }
}

// インスタンス化する
- const shampoo = new Product();
+ const shampoo = new Product('シャンプー', 500, '生活雑貨');
+ const coffee = new Product('コーヒー', 1500, '飲料');

// インスタンス（オブジェクト）の値を出力する
 console.log(shampoo);
+ console.log(coffee);
```

続いて `index.html` をブラウザで開き、デベロッパーツールのコンソールを確認してみましょう。以下のように、引数として渡した値が出力されていればOKです。



本章もあともう一息です。ここで一度休憩しておいてください。クラスは難しい概念なので大変だと思いますが、もう少しだけ頑張りましょう。

14.5 メソッドとは

コンストラクタを使うことで、例えば「商品を作ると同時に出品する」といった**最初の処理**を設定できることを学びました。

では例えば、「商品を出品したあと、商品の個別ページに説明文を表示する」というように、**任意のタイミングで処理を行いたい**場合はどうすればよいでしょうか。

そこで役立つのが**メソッド**です。

実は、オブジェクトには `'シャンプー'` 、 `500` 、 `'生活雑貨'` といった値の他に、関数を持たせることもできます。このオブジェクトが持つ関数のことを、メソッドといいます。

インスタンス（オブジェクト）のメソッドは、クラス内で以下のように定義します。

JSファイル（見本）

```
class Product {
  constructor() {
    初期化の内容
  }

  // メソッドを定義する
  メソッド名() {
    一連の処理
  }
}

```

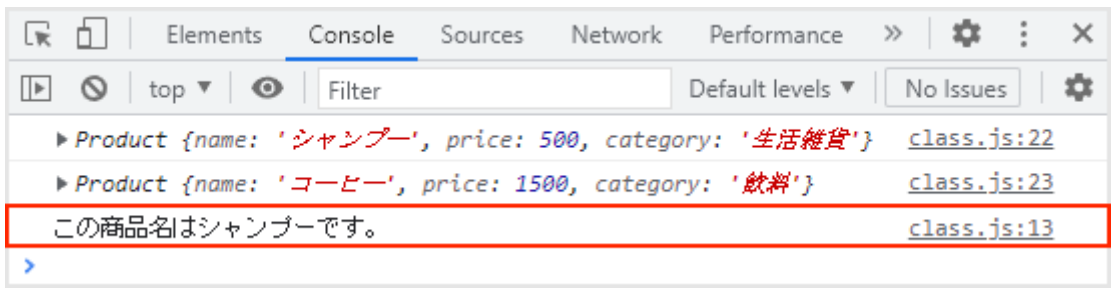
なお、メソッドを呼び出すときは、 `オブジェクト名.メソッド名()` （例： `shampoo.describe()` ）と記述します。

実際にメソッドを定義し、呼び出してみましょう。 `class.js` を以下のように編集してください。

class.js

```
// クラスを定義する
class Product {
  // コンストラクタを使ってインスタンス化するときに初期化する
  constructor(name, price, category) {
    // インスタンス（オブジェクト）にプロパティを持たせる
    this.name = name;
    this.price = price;
    this.category = category;
  }

+   // メソッドを定義する
+   describe() {
+     console.log('この商品名は' + this.name + 'です。');
+   }
}

// インスタンス化する
const shampoo = new Product('シャンプー', 500, '生活雑貨');
const coffee = new Product('コーヒー', 1500, '飲料');

// インスタンス（オブジェクト）の値を出力する
console.log(shampoo);
console.log(coffee);

+ // メソッドを呼び出す（実行する）
+ shampoo.describe();
```

続いて index.html をブラウザで開き、デベロッパーツールのコンソールを確認してみましょう。以下のように、クラス内で定義したメソッドが実行されていればOKです。



通常のオブジェクトにメソッドを定義する方法

クラスから生成されるインスタンスではなく、通常のオブジェクトにメソッドを定義することもできます。

なお、ここでいう「通常のオブジェクト」とは、10章で学んだ以下のようなオブジェクトのことです。

JSファイル（見本）

```
const personalData = { name: '侍太郎', age: 36, gender: '男性' };

```

通常のオブジェクトにメソッドを定義する方法は以下のとおりです。関数を定義するときと似たような書き方になります。

```
const 定数名 = {
  キー名（メソッド名） : () => {
    一連の処理
  }
}
```

インスタンスのときと同様、メソッドを呼び出すときは `オブジェクト名.メソッド名()` （例： `shampoo.describe()` ）と記述します。

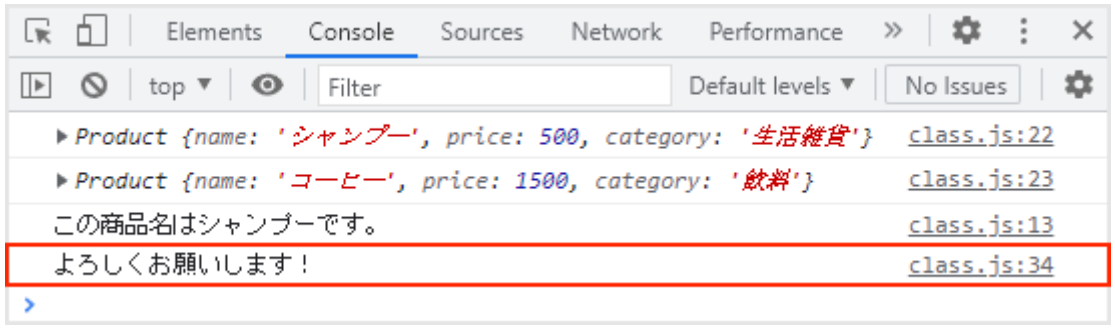
実際にやってみましょう。 `class.js` を以下のように編集してください。

class.js

```
// クラスを定義する
class Product {
  // コンストラクタを使ってインスタンス化するときに初期化する
  constructor(name, price, category) {
    // インスタンス（オブジェクト）にプロパティを持たせる
    this.name = name;
    this.price = price;
    this.category = category;
  }

  // メソッドを定義する
  describe() {
    console.log('この商品名は' + this.name + 'です。');
  }
}

// インスタンス化する
const shampoo = new Product('シャンプー', 500, '生活雑貨');
const coffee = new Product('コーヒー', 1500, '飲料');

// インスタンス（オブジェクト）の値を出力する
console.log(shampoo);
console.log(coffee);

// メソッドを呼び出す（実行する）
shampoo.describe();

+ // 通常のオブジェクトにメソッドを定義する
+ const user = {
+   name: '侍太郎',
+   age: 36,
+   gender: '男性',
+   greet: () => {
+     console.log('よろしくお願いします！');
+   }
+ }
+
+ // メソッドを呼び出す（実行する）
+ user.greet();
```

オブジェクトにメソッドを定義するときやプロパティが多いときは、上記のようにプロパティごとに改行すると見やすくなります。

では `index.html` をブラウザで開き、デベロッパーツールのコンソールを確認してみましょう。以下のように、通常のオブジェクトに定義したメソッドが実行されていればOKです。



まとめ

本章では以下の内容を学習しました。

- クラスとは一言でいえば、**モノ（オブジェクト）の設計図**である
- クラスを使えば、似たようなオブジェクトを大量に作れる
- クラスをもとに作られたオブジェクトのことを、**インスタンス**という
- クラスをもとにオブジェクトを作ることを、**インスタンス化**という
- コンストラクタとは、インスタンス化するときに**初期化を行う関数**のことである
- コンストラクタを使えば、インスタンス化するときに自動で処理を実行したり、インスタンスにプロパティを持たせたりできる
- インスタンスや通常のオブジェクトには関数を持たせることができる
- インスタンスや通常のオブジェクトが持つ関数のことを、**メソッド**という

```
// クラスの定義
class クラス名 {
  // コンストラクタの定義
  constructor() {
    初期化の内容
  }

  // メソッドの定義
  メソッド名() {
    一連の処理
  }
}

// インスタンス化
const 定数名 = new クラス名();

// メソッドの呼び出し
定数名.メソッド名();

// メソッドの定義（通常のオブジェクト）
const 定数名 = {
  キー名（メソッド名） : () => {
    一連の処理
  }
}
```

次章では、DOM操作の基本について学びます。

理解度を選択して次に進みましょう

ボタンを押していただくと次の章に進むことができます

～50%　50～80%　80～100%

最後に確認テストを行いましょう

下のボタンを押すとテストが始まります。

教材をみなおす　テストをはじめる