Chart.js 学習ガイドブック

30 のサンプルで学ぶデータビジュアライゼーション

河合勝彦

名古屋市立大学大学院経済学研究科

kkawai@econ.nagoya-cu.ac.jp

バージョン 1.0

2026 年 1 月 6 日

実践的なサンプルコードで学ぶ
Chart.js の基礎から応用まで

目次

まえがき 5

第 1 章 Chart.js とは 9
1.1 Chart.js の概要 9
1.2 インストール方法 10
1.3 基本構造 10

第 I 部 初心者レベル（01–10） 13

第 2 章 基本的な折れ線グラフ 15
2.1 学習目標 15
2.2 新しく学ぶ関数・プロパティ 16
2.3 コード例 16
2.4 練習問題 17

第 3 章 基本的な棒グラフ 19
3.1 学習目標 19
3.2 新しく学ぶ関数・プロパティ 20
3.3 コード例 20
3.4 練習問題 21

第 4 章 横棒グラフ 23
4.1 学習目標 23
4.2 新しく学ぶ関数・プロパティ 24
4.3 コード例 24
4.4 練習問題 24

第 5 章 基本的な円グラフ 27
5.1 学習目標 27
5.2 新しく学ぶ関数・プロパティ 28
5.3 コード例 29
5.4 練習問題 29

第 6 章　ドーナツチャート 31
6.1　学習目標 31
6.2　新しく学ぶ関数・プロパティ 32
6.3　コード例 33
6.4　練習問題 33

第 7 章　レーダーチャート 35
7.1　学習目標 35
7.2　新しく学ぶ関数・プロパティ 36
7.3　練習問題 37

第 8 章　複数データセットの折れ線グラフ 39
8.1　学習目標 39
8.2　新しく学ぶ関数・プロパティ 40
8.3　練習問題 40

第 9 章　グループ化棒グラフ 43
9.1　学習目標 43
9.2　コード例 44
9.3　練習問題 44

第 10 章　積み上げ棒グラフ 47
10.1　学習目標 47
10.2　新しく学ぶ関数・プロパティ 48
10.3　練習問題 48

第 11 章　エリアチャート 51
11.1　学習目標 51
11.2　新しく学ぶ関数・プロパティ 52
11.3　練習問題 52

第 II 部　中級レベル（11–20） 55

第 12 章　散布図 57
12.1　学習目標 57
12.2　新しく学ぶ関数・プロパティ 58
12.3　練習問題 58

第 13 章　バブルチャート 61
13.1　学習目標 61
13.2　新しく学ぶ関数・プロパティ 62
13.3　練習問題 62

第 14 章　極座標エリアチャート 65
14.1　学習目標 65
14.2　新しく学ぶ関数・プロパティ 66
14.3　練習問題 67

第 15 章　複合チャート 69
15.1　学習目標 69
15.2　新しく学ぶ関数・プロパティ 70
15.3　練習問題 70

第 16 章　ステップ折れ線グラフ 73
16.1　学習目標 73
16.2　新しく学ぶ関数・プロパティ 74
16.3　練習問題 74

第 17 章　グラデーション背景 77
17.1　学習目標 77
17.2　新しく学ぶ関数・プロパティ 78
17.3　練習問題 78

第 18 章　カスタムツールチップ 81
18.1　学習目標 81
18.2　新しく学ぶ関数・プロパティ 82
18.3　練習問題 82

第 19 章　カスタム凡例 85
19.1　学習目標 85
19.2　新しく学ぶ関数・プロパティ 86
19.3　練習問題 87

第 20 章　カスタム軸 89
20.1　学習目標 89
20.2　新しく学ぶ関数・プロパティ 90
20.3　練習問題 90

第 21 章　データラベル 93
21.1　学習目標 93
21.2　新しく学ぶ関数・プロパティ 94
21.3　練習問題 94

第 III 部　上級レベル（21–30） 97

第 22 章　リアルタイムデータ更新 99
22.1 学習目標 . . . 99
22.2 新しく学ぶ関数・プロパティ . . . 100
22.3 練習問題 . . . 101

第 23 章　ドリルダウン 103
23.1 学習目標 . . . 103
23.2 新しく学ぶ関数・プロパティ . . . 104
23.3 練習問題 . . . 105

第 24 章　カスタムプラグイン 107
24.1 学習目標 . . . 107
24.2 新しく学ぶ関数・プロパティ . . . 108
24.3 練習問題 . . . 108

第 25 章　カスタムアニメーション 111
25.1 学習目標 . . . 111
25.2 新しく学ぶ関数・プロパティ . . . 112
25.3 練習問題 . . . 113

第 26 章　複数 Y 軸 115
26.1 学習目標 . . . 115
26.2 新しく学ぶ関数・プロパティ . . . 116
26.3 練習問題 . . . 116

第 27 章　HTML カスタムツールチップ 119
27.1 学習目標 . . . 119
27.2 新しく学ぶ関数・プロパティ . . . 120
27.3 練習問題 . . . 121

第 28 章　クリックイベント処理 123
28.1 学習目標 . . . 123
28.2 新しく学ぶ関数・プロパティ . . . 124
28.3 練習問題 . . . 124

第 29 章　画像エクスポート 127
29.1 学習目標 . . . 127
29.2 新しく学ぶ関数・プロパティ . . . 128
29.3 練習問題 . . . 129

第 30 章　レスポンシブデザイン 131

30.1 学習目標 . . . 131
30.2 新しく学ぶ関数・プロパティ . . . 132
30.3 練習問題 . . . 133

第 31 章 ダッシュボード 135
31.1 学習目標 . . . 135
31.2 新しく学ぶ関数・プロパティ . . . 136
31.3 練習問題 . . . 137

第 IV 部 LLM 活用学習ガイド 141

第 32 章 最短で上達するプロンプトセット 30 本 143
32.1 本章の目的 . . . 143
32.2 この章の使い方 . . . 143
32.3 共通前置き（毎回コピペ推奨） . . . 144
32.4 Phase 1：まず「確実に描ける」（難易度★ 1〜★ 2） . . . 144
32.5 Phase 2：「業務資料っぽく整う」（難易度★ 2〜★ 3） . . . 147
32.6 Phase 3：操作性で差がつく（難易度★ 3） . . . 150
32.7 Phase 4：時系列をちゃんと扱う（難易度★ 3〜★ 4） . . . 152
32.8 Phase 5：大量データと高速化（難易度★ 4） . . . 154
32.9 Phase 6：実務導入（難易度★ 4〜★ 5） . . . 156
32.10 Phase 7：仕上げ（難易度★ 5） . . . 157
32.11 この 30 本を終えると到達できるレベル . . . 158

参考文献・参考サイト 159

まえがき

本書は、JavaScript のチャートライブラリ「Chart.js」を学ぶための実践的なガイドブックです。30 のサンプルファイルを通じて、基本的なグラフの作成から高度なカスタマイズまでを段階的に学習できるように構成されています。加えて、LLM（大規模言語モデル）を活用した効率的な学習のためのプロンプト集も収録しています。

対象読者

- HTML と JavaScript の基礎知識がある方
- Web ページにグラフを表示したい方
- データの可視化に興味がある方
- Chart.js を体系的に学びたい方

本書の構成

第 I 部（01–10）初心者レベル：基本的なチャートタイプの作成
第 II 部（11–20）中級レベル：カスタマイズとスタイリング
第 III 部（21–30）上級レベル：インタラクティブ機能とダッシュボード
第 IV 部（LLM 活用）プロンプトセットで「描ける→整う→使える→速い/壊れない」まで最短で到達

サンプルファイルの入手

本書で解説するサンプルファイル（01〜30）は、以下の GitHub リポジトリから入手できます。

https://github.com/kkawailab/kklab-chartsJs-samples

リポジトリをクローンするか、ZIP ファイルとしてダウンロードしてご利用ください。

章番号とサンプル番号について

本書の各章は、配布されるサンプルファイル（01〜30）に対応しています。

- **章番号**：本書での解説順序（「第 2 章」「第 15 章」など）
- **サンプル番号**：配布ファイル名（`01-basic-line-chart.html` など）

例えば、第 2 章の内容は `samples/01-basic-line-chart.html` に対応します。課題や実習で特定のサンプルを指定する際は、サンプル番号（01〜30）を使用してください。

基本 HTML テンプレート

各サンプルは以下の構造を共有しています。新しいチャートを作成する際のテンプレートとして活用してください。

Listing 1　基本 HTML テンプレート

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <title>Chart.js サンプル</title>
    <script src="https://cdn.jsdelivr.net/npm/
        chart.js@4.4.1/dist/chart.umd.min.js"></script>
    <style>
        /* maintainAspectRatio: false の場合、親要素に高さが必要 */
        .chart-container {
            width: 80%;
            height: 400px;
            margin: 0 auto;
        }
    </style>
</head>
<body>
    <div class="chart-container">
        <canvas id="myChart"></canvas>
    </div>
    <script>
        const ctx = document.getElementById('myChart').getContext('2d');
        new Chart(ctx, {
            type: 'bar',  // チャートタイプを指定
            data: { /* データ設定 */ },
            options: {
                responsive: true,
                maintainAspectRatio: false
            }
        });
    </script>
</body>
</html>
```

よくあるミス：チャートが表示されない

`maintainAspectRatio: false` を指定した場合、`canvas` の親要素に明示的な高さ（height）を設定する必要があります。高さがないとチャートが表示されません。

第 1 章

Chart.js とは

1.1 Chart.js の概要

Chart.js は、HTML5 の Canvas 要素を使用してグラフを描画するオープンソースの JavaScript ライブラリです。2013 年に初版がリリースされ、執筆時点（2026 年 1 月）ではバージョン 4.x 系が最新です。

1.1.1 特徴

- **簡単**：少ないコードで美しいグラフを作成可能
- **レスポンシブ**：画面サイズに自動適応
- **カスタマイズ性**：色、フォント、アニメーションなど柔軟に設定可能
- **8 種類のチャートタイプ**：折れ線、棒、円、ドーナツ、レーダー、極座標、散布図、バブル
- **プラグイン対応**：機能拡張が容易

1.1.2 対応チャートタイプ

タイプ	type 値	用途
折れ線グラフ	`line`	時系列データ、トレンド表示
棒グラフ	`bar`	カテゴリ比較
円グラフ	`pie`	構成比の表示
ドーナツ	`doughnut`	構成比（中央に情報表示可）
レーダー	`radar`	多軸データの比較
極座標	`polarArea`	放射状のカテゴリ比較
散布図	`scatter`	2 変数の相関関係
バブル	`bubble`	3 変数の関係（大きさで表現）

表 1.1　Chart.js のチャートタイプ一覧

1.2 インストール方法

1.2.1 CDN を使用する方法（推奨）

最も簡単な方法は、CDN（Content Delivery Network）を使用することです。

バージョン固定の重要性

教材や本番環境では、将来のメジャーアップデートによる互換性問題を避けるため、バージョンを固定することを強く推奨します。

```
<!-- バージョンを固定して読み込み（推奨） -->
<script src="https://cdn.jsdelivr.net/npm/
    chart.js@4.4.1/dist/chart.umd.min.js"></script>
```

1.2.2 npm を使用する方法

Node.js 環境（Vite、webpack 等）では、npm でインストールできます。

```
npm install chart.js
```

npm 環境では、CDN とは異なる import 文を使用します。

Listing 1.1 npm 環境での使用例

```
// chart.js/auto を使用すると全機能が自動登録される
import Chart from 'chart.js/auto';

new Chart(document.getElementById('myChart'), {
    type: 'bar',
    data: { /* ... */ }
});
```

1.3 基本構造

Chart.js は、HTML5 の<canvas>要素に描画します。基本的な構造は以下の通りです。

Listing 1.2 HTML 構造

```
<canvas id="myChart"></canvas>
```

Listing 1.3 JavaScript 基本構造

```
const ctx = document.getElementById('myChart').getContext('2d');
const myChart = new Chart(ctx, {
    type: 'line',         // チャートタイプ
    data: { /* データ */ },
    options: { /* オプション */ }
});
```

ポイント

`getContext('2d')` メソッドは、Canvas 要素の 2D 描画コンテキストを取得します。Chart.js はこのコンテキストを使用してグラフを描画します。

第 I 部

初心者レベル（01–10）

第 2 章

基本的な折れ線グラフ

2.1　学習目標

この章を学習すると、以下のことができるようになります。

- Chart.js の基本的なセットアップ方法を理解する
- <canvas>要素と JavaScript の連携方法を習得する
- 折れ線グラフ（type: 'line'）を作成できる
- labels と datasets の基本構造を理解する
- borderColor、backgroundColor、tension などの基本プロパティを設定できる

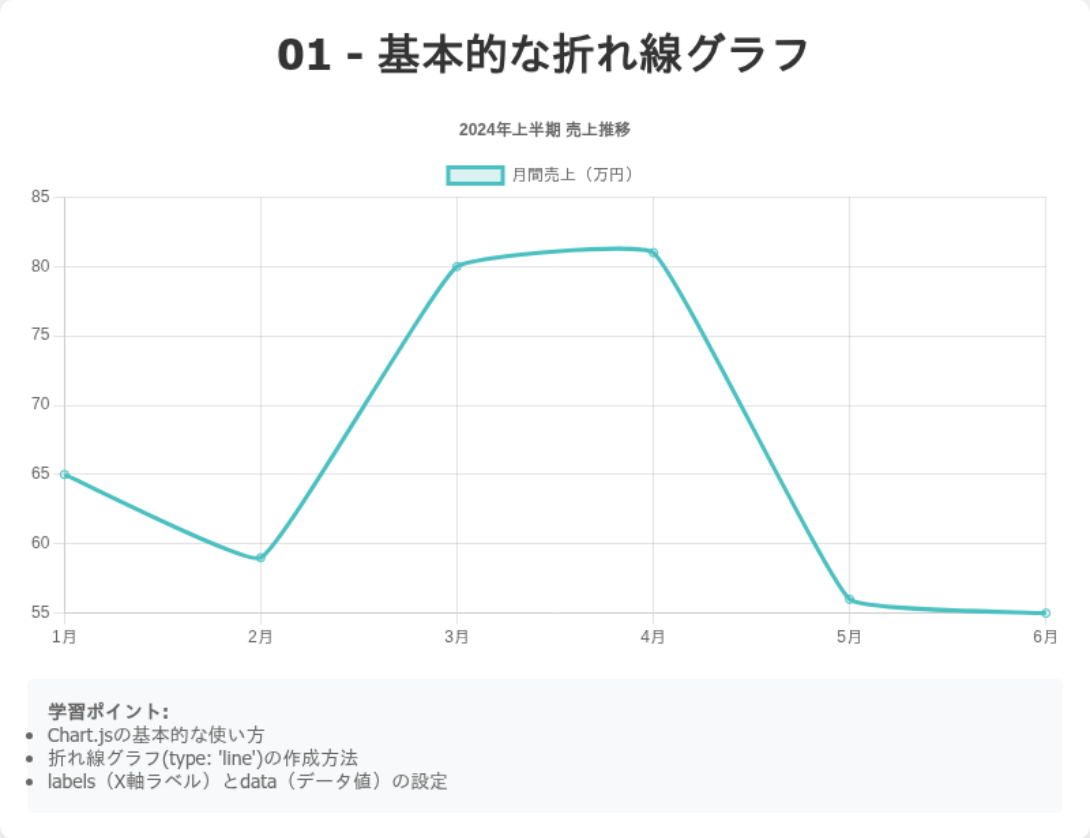

図 2.1　基本的な折れ線グラフの実行結果

2.2 新しく学ぶ関数・プロパティ

2.2.1 Chart コンストラクタ

構文

```
new Chart(context, config)
```

- `context`：Canvas 要素の 2D コンテキスト
- `config`：チャートの設定オブジェクト（type, data, options）

2.2.2 type: 'line'

チャートタイプを折れ線グラフに設定します。

2.2.3 data オブジェクト

- `labels`：X 軸に表示するラベルの配列
- `datasets`：データセットの配列

2.2.4 datasets の主要プロパティ

- `label`：凡例に表示される名前
- `data`：Y 軸の値の配列
- `borderColor`：線の色
- `backgroundColor`：塗りつぶしの色
- `tension`：線の曲がり具合（0=直線、1=最大曲線）

2.2.5 options オブジェクト

- `responsive`：レスポンシブ対応（true/false）
- `maintainAspectRatio`：アスペクト比維持（true/false）
- `plugins.title`：タイトル設定

2.3 コード例

Listing 2.1 基本的な折れ線グラフ

```
const ctx = document.getElementById('myChart').getContext('2d');
```

```
const data = {
    labels: ['1月', '2月', '3月', '4月', '5月', '6月'],
    datasets: [{
        label: '月間売上（万円）',
        data: [65, 59, 80, 81, 56, 55],
        borderColor: 'rgb(75, 192, 192)',
        backgroundColor: 'rgba(75, 192, 192, 0.2)',
        tension: 0.1
    }]
};

new Chart(ctx, {
    type: 'line',
    data: data,
    options: {
        responsive: true,
        maintainAspectRatio: false
    }
});
```

学習ポイント

- Chart.js は<canvas>要素に描画する
- type: 'line' で折れ線グラフを指定
- labels と datasets 内の data の順番は対応している

2.4 練習問題

問題 1-1

以下の要件を満たす折れ線グラフを作成してください。

- X 軸：「月」「火」「水」「木」「金」
- データ：来客数として 45, 62, 58, 71, 89
- 線の色：青色（rgb(54, 162, 235)）
- タイトル：「週間来客数」

模範解答 1-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'line',
    data: {
        labels: ['月', '火', '水', '木', '金'],
```

```
        datasets: [{
            label: '来客数',
            data: [45, 62, 58, 71, 89],
            borderColor: 'rgb(54, 162, 235)',
            backgroundColor: 'rgba(54, 162, 235, 0.2)',
            tension: 0.1
        }]
    },
    options: {
        plugins: {
            title: { display: true, text: '週間来客数' }
        }
    }
});
```

第 3 章

基本的な棒グラフ

3.1　学習目標

この章を学習すると、以下のことができるようになります。

- 縦棒グラフ（`type: 'bar'`）を作成できる
- 各バーに異なる色を配列で設定できる
- `borderWidth` と `borderRadius` でバーの外観をカスタマイズできる
- RGBA 形式で透明度を含む色を指定できる

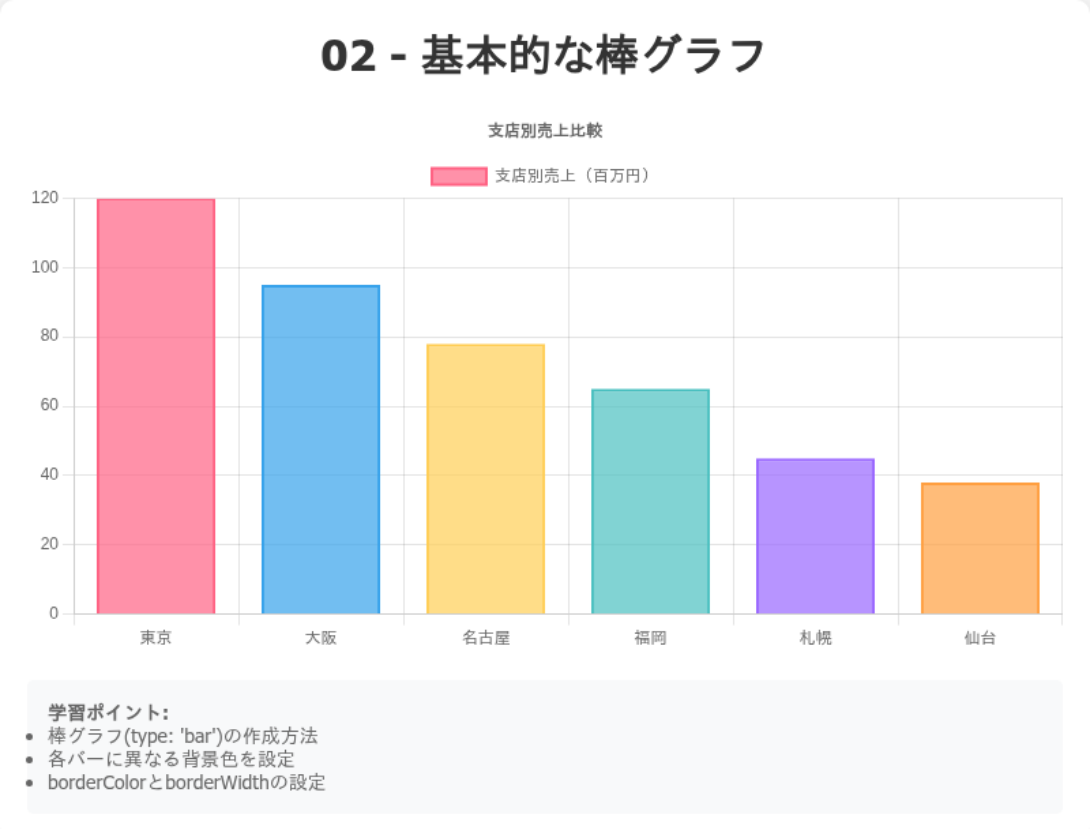

図 3.1　基本的な棒グラフの実行結果

3.2 新しく学ぶ関数・プロパティ

3.2.1 type: 'bar'

チャートタイプを縦棒グラフに設定します。

3.2.2 データセットの色設定

- backgroundColor：各バーの塗りつぶし色（配列で個別指定可能）
- borderColor：各バーの枠線色
- borderWidth：枠線の太さ
- borderRadius：バーの角の丸み

3.3 コード例

Listing 3.1 基本的な棒グラフ

```
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['商品A', '商品B', '商品C', '商品D', '商品E'],
        datasets: [{
            label: '売上個数',
            data: [120, 190, 30, 50, 80],
            backgroundColor: [
                'rgba(255, 99, 132, 0.7)',
                'rgba(54, 162, 235, 0.7)',
                'rgba(255, 206, 86, 0.7)',
                'rgba(75, 192, 192, 0.7)',
                'rgba(153, 102, 255, 0.7)'
            ],
            borderWidth: 1,
            borderRadius: 5
        }]
    }
});
```

学習ポイント

- backgroundColor を配列で指定すると各バーに異なる色を設定可能
- RGBA 形式で透明度を指定できる
- borderRadius でバーの角を丸くできる

3.4 練習問題

問題 2-1

以下の要件を満たす棒グラフを作成してください。

- X 軸：「国語」「数学」「英語」「理科」「社会」
- データ：テストの平均点として 78, 65, 82, 71, 88
- 各バーの色：すべて緑系統（`rgba(75, 192, 192, 0.7)`）
- 枠線の太さ：2px、角の丸み：8px

模範解答 2-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['国語', '数学', '英語', '理科', '社会'],
        datasets: [{
            label: '平均点',
            data: [78, 65, 82, 71, 88],
            backgroundColor: 'rgba(75, 192, 192, 0.7)',
            borderColor: 'rgb(75, 192, 192)',
            borderWidth: 2,
            borderRadius: 8
        }]
    },
    options: {
        plugins: {
            title: { display: true, text: '科目別平均点' }
        }
    }
});
```

第 4 章

横棒グラフ

4.1　学習目標

この章を学習すると、以下のことができるようになります。

- `indexAxis: 'y'` オプションで棒グラフを横向きに変換できる
- 縦棒と横棒の使い分けを判断できる
- ラベルが長いデータに適したレイアウトを選択できる

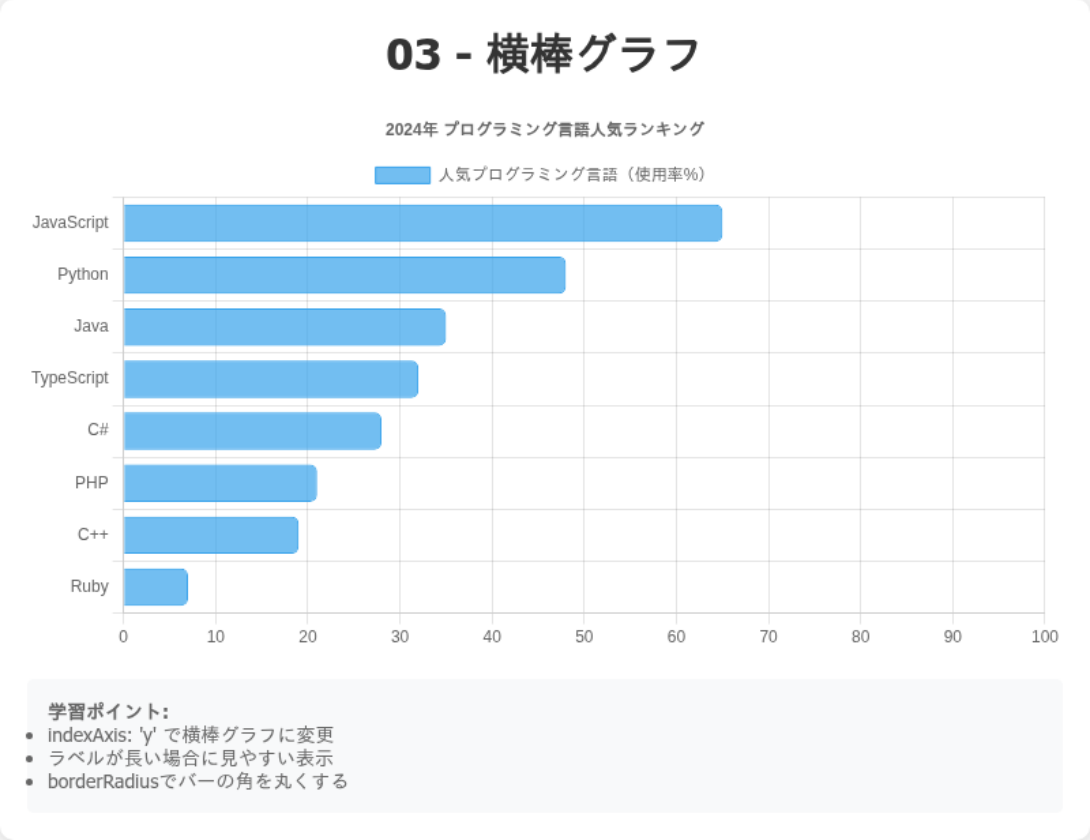

図 4.1　横棒グラフの実行結果

4.2 新しく学ぶ関数・プロパティ

4.2.1 indexAxis

構文

```
options: {
    indexAxis: 'y'  // 'x'（デフォルト）または 'y'
}
```

`indexAxis: 'y'` を設定すると、棒グラフが横向きになります。

4.3 コード例

Listing 4.1 横棒グラフ

```
new Chart(ctx, {
    type: 'bar',
    data: { /* データ */ },
    options: {
        indexAxis: 'y',  // 横棒グラフに変更
        responsive: true,
        maintainAspectRatio: false
    }
});
```

学習ポイント

`indexAxis: 'y'` を指定するだけで縦棒グラフを横棒グラフに変換できます。データの構造は変更不要です。

4.4 練習問題

問題 3-1

以下の要件を満たす横棒グラフを作成してください。

- Y 軸：「JavaScript」「Python」「Java」「C++」「Ruby」（プログラミング言語）
- データ：人気度スコアとして 95, 90, 75, 65, 55
- バーの色：それぞれ異なる色を配列で指定
- タイトル：「プログラミング言語人気ランキング」

模範解答 3-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['JavaScript', 'Python', 'Java', 'C++', 'Ruby'],
        datasets: [{
            label: '人気度スコア',
            data: [95, 90, 75, 65, 55],
            backgroundColor: [
                'rgba(255, 206, 86, 0.7)',
                'rgba(54, 162, 235, 0.7)',
                'rgba(255, 99, 132, 0.7)',
                'rgba(75, 192, 192, 0.7)',
                'rgba(153, 102, 255, 0.7)'
            ]
        }]
    },
    options: {
        indexAxis: 'y',
        plugins: {
            title: { display: true, text:
                'プログラミング言語人気ランキング' }
        }
    }
});
```

第 5 章

基本的な円グラフ

5.1　学習目標

この章を学習すると、以下のことができるようになります。

- 円グラフ（`type: 'pie'`）を作成できる
- 構成比を視覚化するデータ形式を理解する
- `hoverOffset` でホバー時のインタラクションを設定できる
- 各セグメントに色を割り当てることができる

図 5.1　基本的な円グラフの実行結果

5.2　新しく学ぶ関数・プロパティ

5.2.1　type: 'pie'

チャートタイプを円グラフに設定します。

5.2.2　hoverOffset

ホバー時にセグメントが外側に移動する距離を設定します。

```
datasets: [{
    data: [30, 20, 25, 15, 10],
    hoverOffset: 10  // ホバー時に10px外側に
}]
```

5.3 コード例

Listing 5.1 基本的な円グラフ

```
new Chart(ctx, {
    type: 'pie',
    data: {
        labels: ['電子機器', '家電', '日用品', '衣類', 'その他'],
        datasets: [{
            data: [30, 20, 25, 15, 10],
            backgroundColor: [
                '#FF6384', '#36A2EB', '#FFCE56',
                '#4BC0C0', '#9966FF'
            ],
            hoverOffset: 10
        }]
    }
});
```

5.4 練習問題

問題 4-1

以下の要件を満たす円グラフを作成してください。

- ラベル：「朝食」「昼食」「夕食」「間食」
- データ：1 日の食事カロリー割合として 20, 35, 35, 10
- 4 色を使用（暖色系を推奨）
- ホバー時のオフセット：15px
- タイトル：「1 日の食事バランス」

模範解答 4-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'pie',
    data: {
        labels: ['朝食', '昼食', '夕食', '間食'],
        datasets: [{
            data: [20, 35, 35, 10],
            backgroundColor: [
                '#FF6384', '#FF9F40', '#FFCE56', '#FF6B6B'
            ],
```

```
            hoverOffset: 15
        }]
    },
    options: {
        plugins: {
            title: { display: true, text: '1日の食事バランス' }
        }
    }
});
```

第 6 章

ドーナツチャート

6.1　学習目標

この章を学習すると、以下のことができるようになります。

- ドーナツチャート（`type: 'doughnut'`）を作成できる
- `cutout` オプションで中央の穴のサイズを調整できる
- 円グラフとドーナツチャートの使い分けを判断できる
- 凡例の位置を `plugins.legend.position` で変更できる

図 6.1 ドーナツチャートの実行結果

6.2 新しく学ぶ関数・プロパティ

6.2.1 type: 'doughnut'

チャートタイプをドーナツに設定します。

6.2.2 cutout

中央の穴の大きさを設定します。パーセンテージまたはピクセル値で指定可能です。

```
options: {
    cutout: '60%'  // 中央60%を空ける
}
```

6.3 コード例

Listing 6.1 ドーナツチャート

```
new Chart(ctx, {
    type: 'doughnut',
    data: { /* データ */ },
    options: {
        cutout: '60%',
        plugins: {
            legend: {
                position: 'right'
            }
        }
    }
});
```

6.4 練習問題

問題 5-1

以下の要件を満たすドーナツチャートを作成してください。

- ラベル：「完了」「進行中」「未着手」
- データ：タスク進捗として 45, 30, 25
- 中央の穴：70%
- 凡例の位置：下側（bottom）
- タイトル：「プロジェクト進捗状況」

模範解答 5-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'doughnut',
    data: {
        labels: ['完了', '進行中', '未着手'],
        datasets: [{
            data: [45, 30, 25],
            backgroundColor: ['#4BC0C0', '#FFCE56', '#FF6384']
        }]
    },
    options: {
        cutout: '70%',
```

```
        plugins: {
            legend: { position: 'bottom' },
            title: { display: true, text: 'プロジェクト進捗状況' }
        }
    }
});
```

第 7 章

レーダーチャート

7.1　学習目標

この章を学習すると、以下のことができるようになります。

- レーダーチャート（`type: 'radar'`）を作成できる
- 複数の評価軸を持つデータを視覚化できる
- `scales.r` で放射状スケールをカスタマイズできる
- `pointLabels` で軸ラベルのスタイルを設定できる

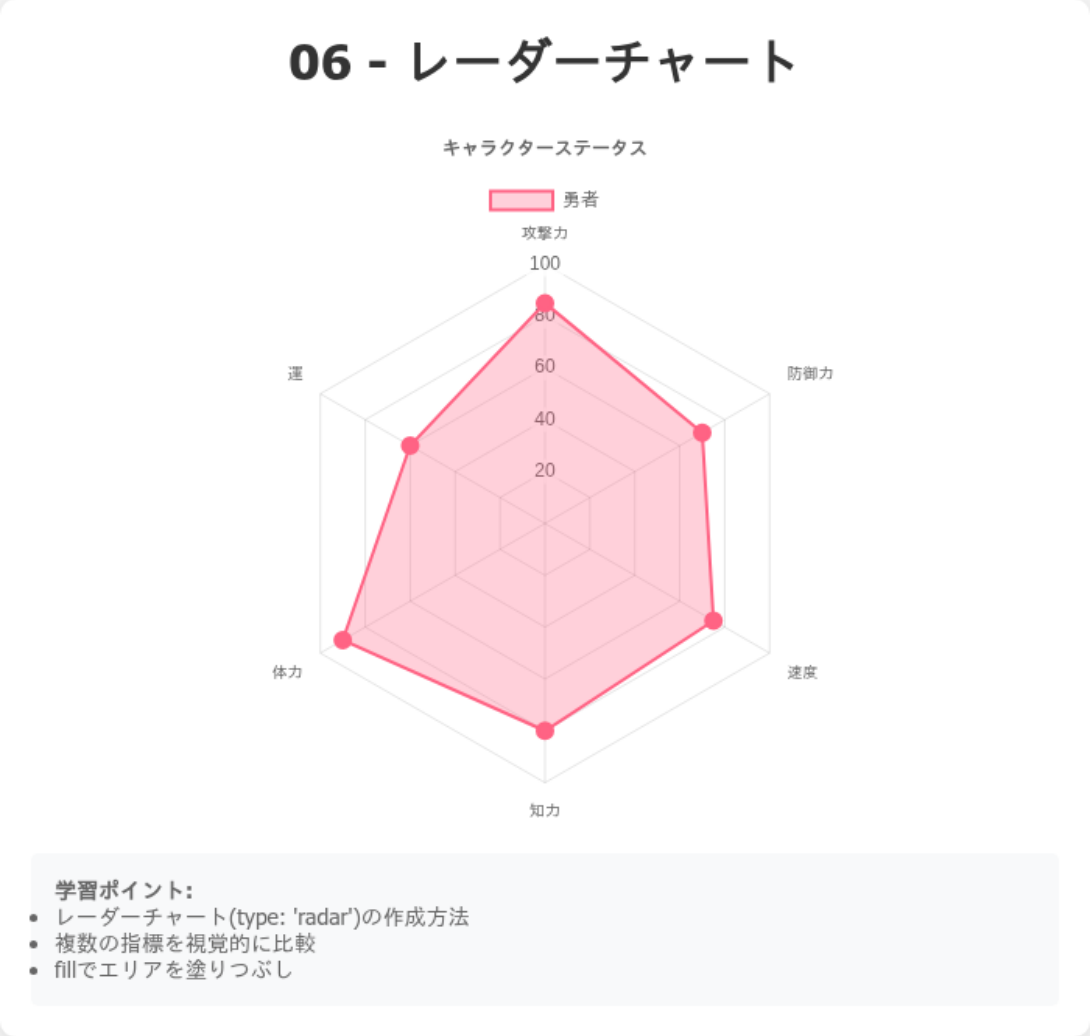

図 7.1　レーダーチャートの実行結果

7.2　新しく学ぶ関数・プロパティ

7.2.1　type: 'radar'

チャートタイプをレーダーに設定します。

7.2.2　scales.r

レーダーチャートの放射状スケールを設定します。

```
options: {
    scales: {
        r: {
            beginAtZero: true,
            max: 100,
            ticks: { stepSize: 20 },
            pointLabels: { font: { size: 14 } }
```

```
        }
    }
}
```

7.3 練習問題

問題 6-1

以下の要件を満たすレーダーチャートを作成してください。

- ラベル（評価軸）:「デザイン」「機能性」「価格」「耐久性」「サポート」
- データ：2 つの製品を比較
  - 製品 A：80, 90, 70, 85, 75
  - 製品 B：70, 75, 90, 70, 85
- スケール：0 から 100、ステップ 20
- タイトル：「製品比較」

模範解答 6-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'radar',
    data: {
        labels: ['デザイン', '機能性', '価格', '耐久性',
            'サポート'],
        datasets: [
            {
                label: '製品A',
                data: [80, 90, 70, 85, 75],
                borderColor: 'rgb(255, 99, 132)',
                backgroundColor: 'rgba(255, 99, 132, 0.2)'
            },
            {
                label: '製品B',
                data: [70, 75, 90, 70, 85],
                borderColor: 'rgb(54, 162, 235)',
                backgroundColor: 'rgba(54, 162, 235, 0.2)'
            }
        ]
    },
    options: {
        scales: {
            r: {
                beginAtZero: true,
                max: 100,
```

```
                ticks: { stepSize: 20 }
            }
        },
        plugins: {
            title: { display: true, text: '製品比較' }
        }
    }
});
```

第 8 章

複数データセットの折れ線グラフ

8.1　学習目標

この章を学習すると、以下のことができるようになります。

- `datasets` 配列に複数のデータセットを追加できる
- 複数の折れ線を 1 つのチャートに表示できる
- 各データセットに異なる色とスタイルを設定できる
- 凡例を使ってデータセットを識別できる

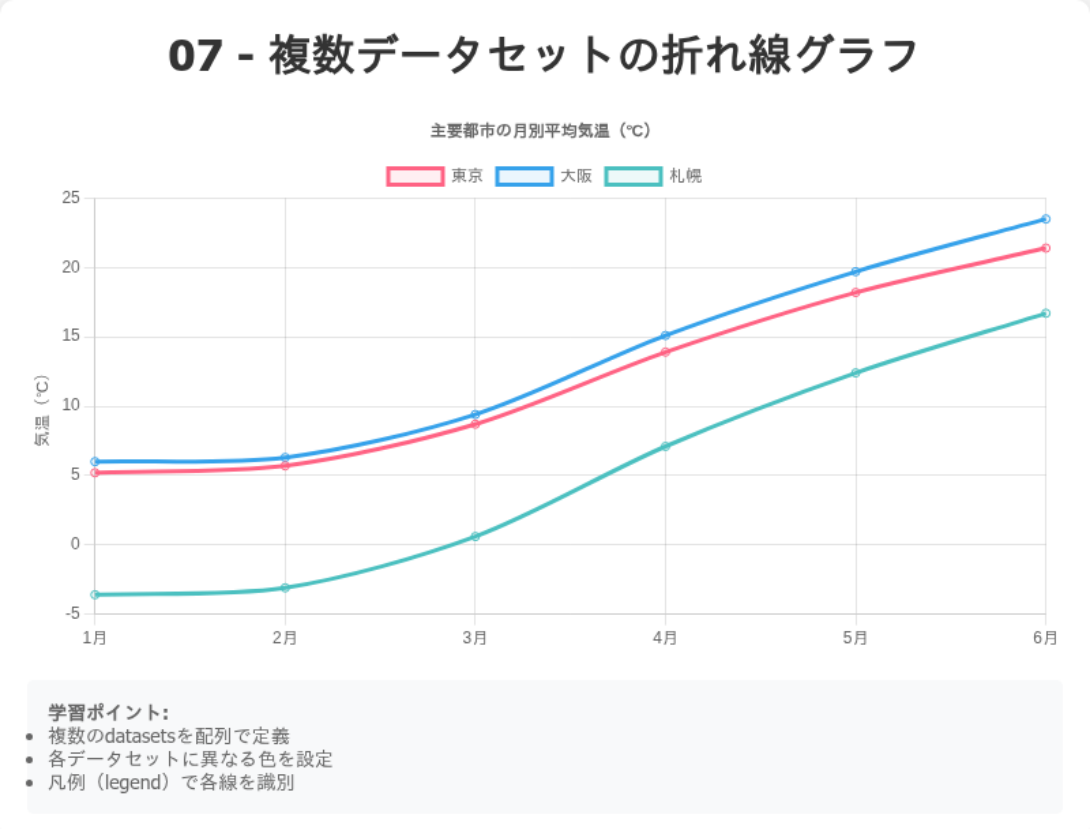

図 8.1　複数データセットの折れ線グラフの実行結果

8.2 新しく学ぶ関数・プロパティ

8.2.1 複数の datasets

datasets 配列に複数のオブジェクトを追加することで、複数の線を表示できます。

Listing 8.1 複数データセット

```
datasets: [
    {
        label: 'データセット1',
        data: [10, 20, 30, 40, 50],
        borderColor: 'red'
    },
    {
        label: 'データセット2',
        data: [15, 25, 35, 45, 55],
        borderColor: 'blue'
    }
]
```

8.3 練習問題

問題 7-1

以下の要件を満たす複数データセットの折れ線グラフを作成してください。

- X 軸：「4 月」「5 月」「6 月」「7 月」「8 月」「9 月」
- データ：
  - 東京の気温：18, 22, 25, 30, 32, 28
  - 札幌の気温：12, 16, 20, 24, 26, 22
- 東京は赤色、札幌は青色の線
- タイトル：「月別平均気温の比較」

模範解答 7-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'line',
    data: {
        labels: ['4月', '5月', '6月', '7月', '8月', '9月'],
        datasets: [
            {
                label: '東京',
```

```
                data: [18, 22, 25, 30, 32, 28],
                borderColor: 'rgb(255, 99, 132)',
                backgroundColor: 'rgba(255, 99, 132, 0.2)',
                tension: 0.3
            },
            {
                label: '札幌',
                data: [12, 16, 20, 24, 26, 22],
                borderColor: 'rgb(54, 162, 235)',
                backgroundColor: 'rgba(54, 162, 235, 0.2)',
                tension: 0.3
            }
        ]
    },
    options: {
        plugins: {
            title: { display: true, text: '月別平均気温の比較' }
        }
    }
});
```

第 9 章

グループ化棒グラフ

9.1　学習目標

この章を学習すると、以下のことができるようになります。

- 複数のデータセットを並べて表示するグループ化棒グラフを作成できる
- 同じカテゴリ内で複数の値を比較できる
- 年度比較や製品比較などのユースケースに適用できる

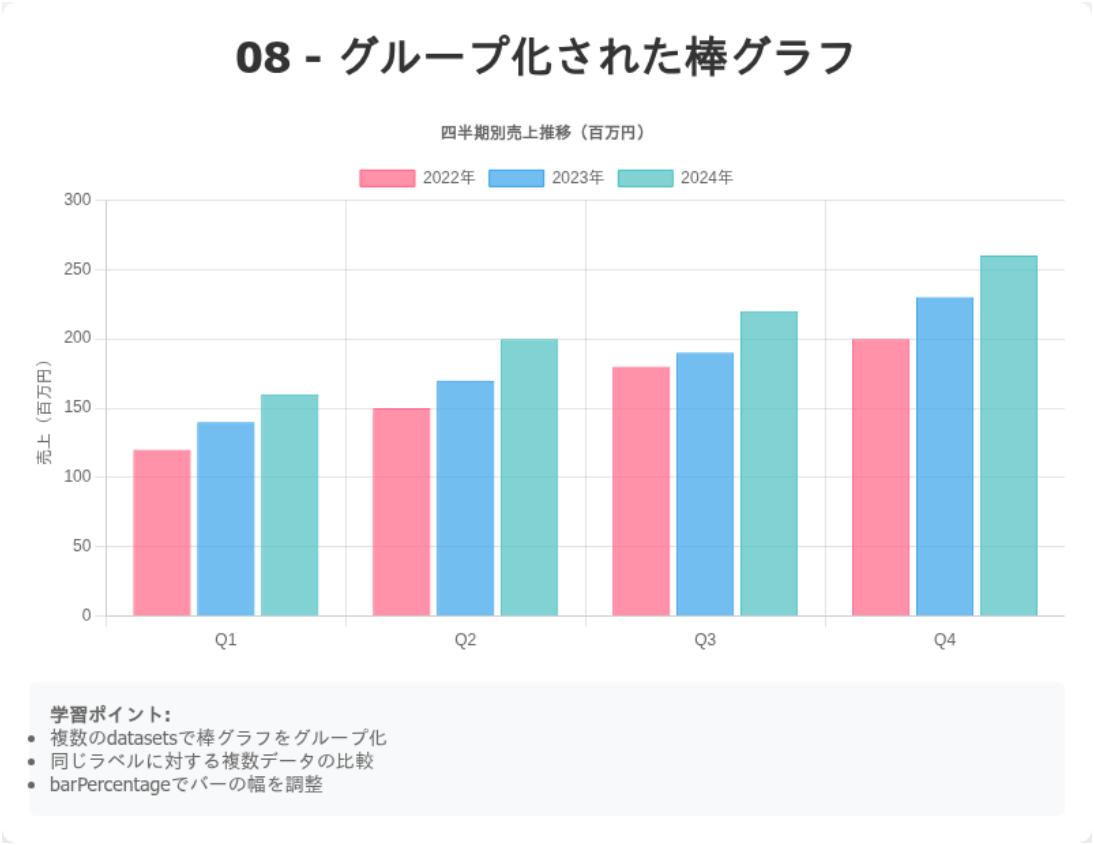

図 9.1　グループ化棒グラフの実行結果

9.2 コード例

Listing 9.1 グループ化棒グラフ

```
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['1月', '2月', '3月'],
        datasets: [
            { label: '2023年', data: [100, 120, 140], backgroundColor:
                'rgba(255, 99, 132, 0.7)' },
            { label: '2024年', data: [120, 140, 160], backgroundColor:
                'rgba(54, 162, 235, 0.7)' }
        ]
    }
});
```

9.3 練習問題

問題 8-1

以下の要件を満たすグループ化棒グラフを作成してください。

- X 軸：「東京店」「大阪店」「名古屋店」「福岡店」
- データ：
  - 上半期売上：320, 280, 190, 150
  - 下半期売上：380, 310, 220, 180
- 上半期は緑系、下半期は紫系の色
- タイトル：「店舗別売上比較」

模範解答 8-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['東京店', '大阪店', '名古屋店', '福岡店'],
        datasets: [
            {
                label: '上半期売上',
                data: [320, 280, 190, 150],
                backgroundColor: 'rgba(75, 192, 192, 0.7)'
            },
```

```
            {
                label: '下半期売上',
                data: [380, 310, 220, 180],
                backgroundColor: 'rgba(153, 102, 255, 0.7)'
            }
        ]
    },
    options: {
        plugins: {
            title: { display: true, text: '店舗別売上比較' }
        }
    }
});
```

第 10 章

積み上げ棒グラフ

10.1　学習目標

この章を学習すると、以下のことができるようになります。

- `scales.x.stacked` と `scales.y.stacked` で積み上げ表示を有効化できる
- カテゴリごとの内訳と合計を同時に表現できる
- グループ化棒グラフとの使い分けを判断できる

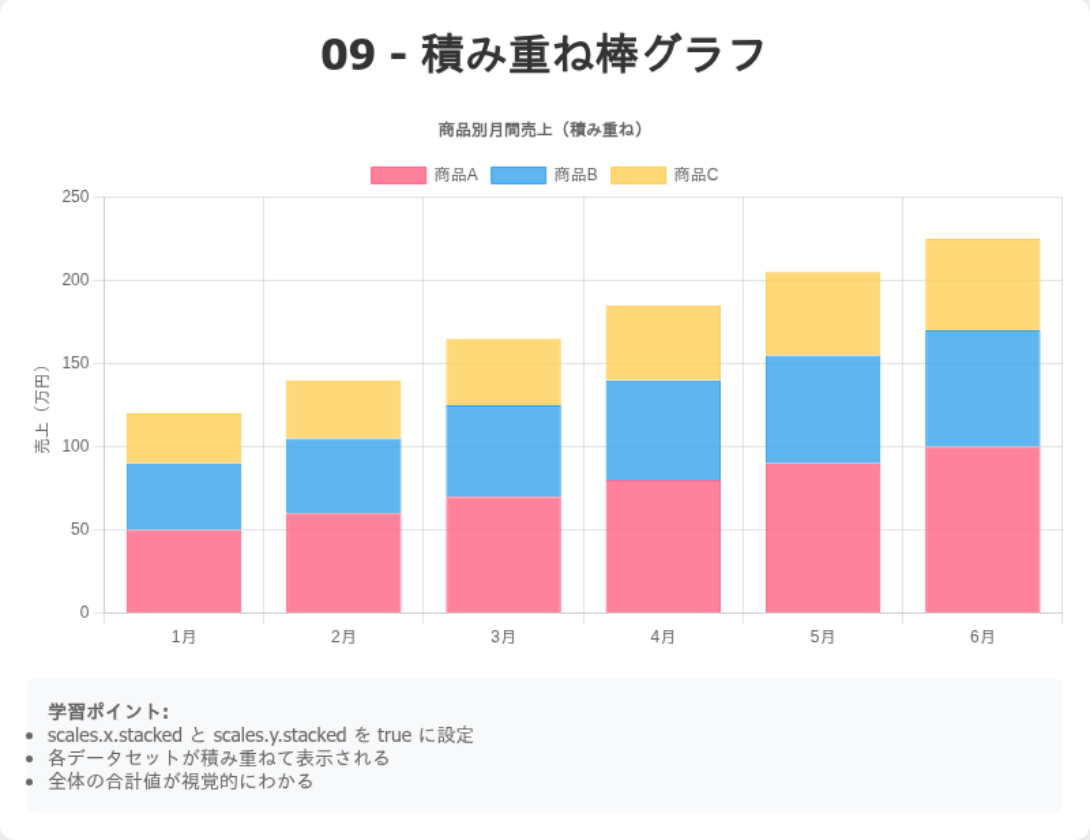

図 10.1　積み上げ棒グラフの実行結果

10.2 新しく学ぶ関数・プロパティ

10.2.1 stacked

```
options: {
    scales: {
        x: { stacked: true },
        y: { stacked: true }
    }
}
```

X 軸と Y 軸の両方で `stacked: true` を設定することで、積み上げ表示になります。

10.3 練習問題

問題 9-1

以下の要件を満たす積み上げ棒グラフを作成してください。

- X 軸：「Q1」「Q2」「Q3」「Q4」（四半期）
- データ：3 つの製品カテゴリの売上
  - 電化製品：150, 180, 200, 220
  - 家具：80, 90, 110, 100
  - 雑貨：50, 60, 70, 80
- 積み上げ表示を有効化
- タイトル：「四半期別カテゴリ売上」

模範解答 9-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['Q1', 'Q2', 'Q3', 'Q4'],
        datasets: [
            {
                label: '電化製品',
                data: [150, 180, 200, 220],
                backgroundColor: 'rgba(255, 99, 132, 0.7)'
            },
            {
                label: '家具',
                data: [80, 90, 110, 100],
```

```
                backgroundColor: 'rgba(54, 162, 235, 0.7)'
            },
            {
                label: '雑貨',
                data: [50, 60, 70, 80],
                backgroundColor: 'rgba(255, 206, 86, 0.7)'
            }
        ]
    },
    options: {
        scales: {
            x: { stacked: true },
            y: { stacked: true }
        },
        plugins: {
            title: { display: true, text: '四半期別カテゴリ売上' }
        }
    }
});
```

第 11 章

エリアチャート

11.1　学習目標

この章を学習すると、以下のことができるようになります。

- `fill: true` オプションで線の下を塗りつぶせる
- エリアチャートで累積値や面積を強調できる
- 複数のエリアを重ねて表示できる
- 折れ線グラフとエリアチャートの使い分けを判断できる

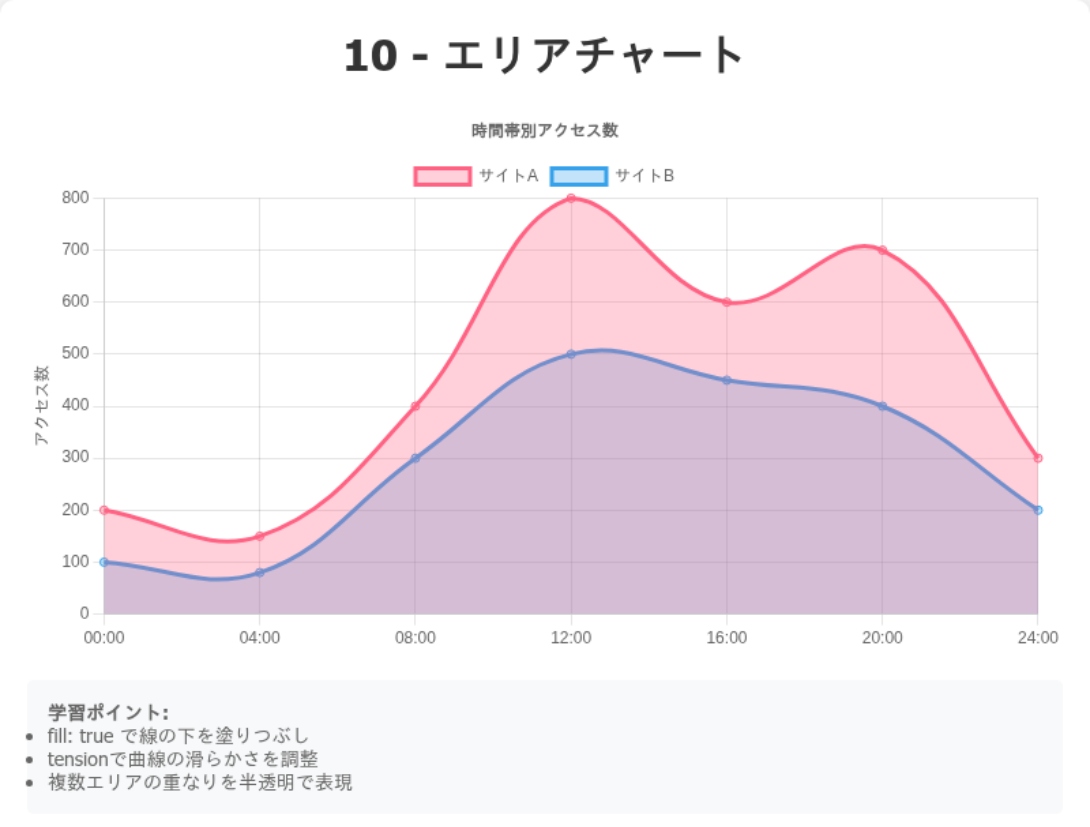

図 11.1　エリアチャートの実行結果

11.2 新しく学ぶ関数・プロパティ

11.2.1 fill

```
datasets: [{
    data: [10, 20, 30],
    fill: true  // 線の下を塗りつぶす
}]
```

fill: true で線の下を背景色で塗りつぶします。

11.3 練習問題

問題 10-1

以下の要件を満たすエリアチャートを作成してください。

- X 軸：「1 月」「2 月」「3 月」「4 月」「5 月」「6 月」
- データ：Web サイトの月間 PV（ページビュー）として 12000, 15000, 18000, 22000, 25000, 28000
- 塗りつぶしを有効化（fill: true）
- 線と塗りつぶしの色：オレンジ系
- 線の曲がり具合：0.4
- タイトル：「月間 PV 推移」

模範解答 10-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'line',
    data: {
        labels: ['1月', '2月', '3月', '4月', '5月', '6月'],
        datasets: [{
            label: 'ページビュー',
            data: [12000, 15000, 18000, 22000, 25000, 28000],
            fill: true,
            borderColor: 'rgb(255, 159, 64)',
            backgroundColor: 'rgba(255, 159, 64, 0.3)',
            tension: 0.4
        }]
    },
    options: {
        plugins: {
```

```
            title: { display: true, text: '月間PV推移' }
        }
    }
});
```

第 II 部

中級レベル（11–20）

第 12 章

散布図

12.1　学習目標

この章を学習すると、以下のことができるようになります。

- 散布図（`type: 'scatter'`）を作成できる
- `{x, y}`オブジェクト形式でデータを指定できる
- 2 つの変数間の相関関係を視覚化できる
- `pointRadius` と `pointHoverRadius` でポイントサイズを設定できる

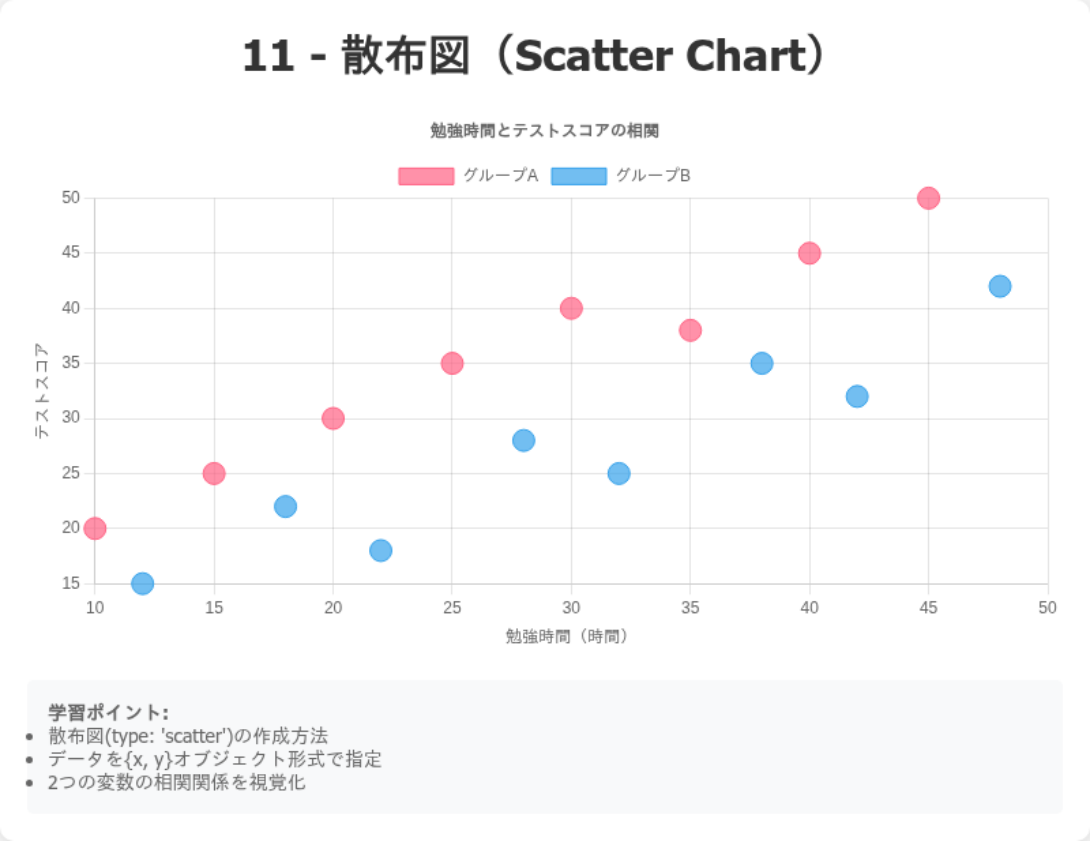

図 12.1　散布図の実行結果

12.2 新しく学ぶ関数・プロパティ

12.2.1 type: 'scatter'

チャートタイプを散布図に設定します。

12.2.2 データ形式

散布図では、データを`{x, y}`オブジェクトの形式で指定します。

Listing 12.1 散布図のデータ形式

```
data: [
    { x: 10, y: 20 },
    { x: 15, y: 25 },
    { x: 20, y: 30 }
]
```

12.2.3 pointRadius / pointHoverRadius

- `pointRadius`：データポイントのサイズ
- `pointHoverRadius`：ホバー時のポイントサイズ

12.3 練習問題

問題 11-1

以下の要件を満たす散布図を作成してください。

- データ：勉強時間と試験点数の関係
  - {x: 2, y: 55}, {x: 3, y: 62}, {x: 4, y: 70}, {x: 5, y: 75}, {x: 6, y: 82}, {x: 7, y: 88}, {x: 8, y: 92}
- ポイントの半径：8px、ホバー時：12px
- 色：青系統
- タイトル：「勉強時間と試験点数の相関」

模範解答 11-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'scatter',
    data: {
```

```
        datasets: [{
            label: '学生データ',
            data: [
                {x: 2, y: 55}, {x: 3, y: 62}, {x: 4, y: 70},
                {x: 5, y: 75}, {x: 6, y: 82}, {x: 7, y: 88}, {x: 8, y: 92}
            ],
            backgroundColor: 'rgba(54, 162, 235, 0.7)',
            pointRadius: 8,
            pointHoverRadius: 12
        }]
    },
    options: {
        scales: {
            x: { title: { display: true, text: '勉強時間（時間）' } },
            y: { title: { display: true, text: '試験点数' } }
        },
        plugins: {
            title: { display: true, text: '勉強時間と試験点数の相関' }
        }
    }
});
```

第 13 章

バブルチャート

13.1　学習目標

この章を学習すると、以下のことができるようになります。

- バブルチャート（type: 'bubble'）を作成できる
- {x, y, r}オブジェクト形式で 3 変数のデータを指定できる
- バブルの大きさで第 3 の変数を表現できる
- 散布図とバブルチャートの使い分けを判断できる

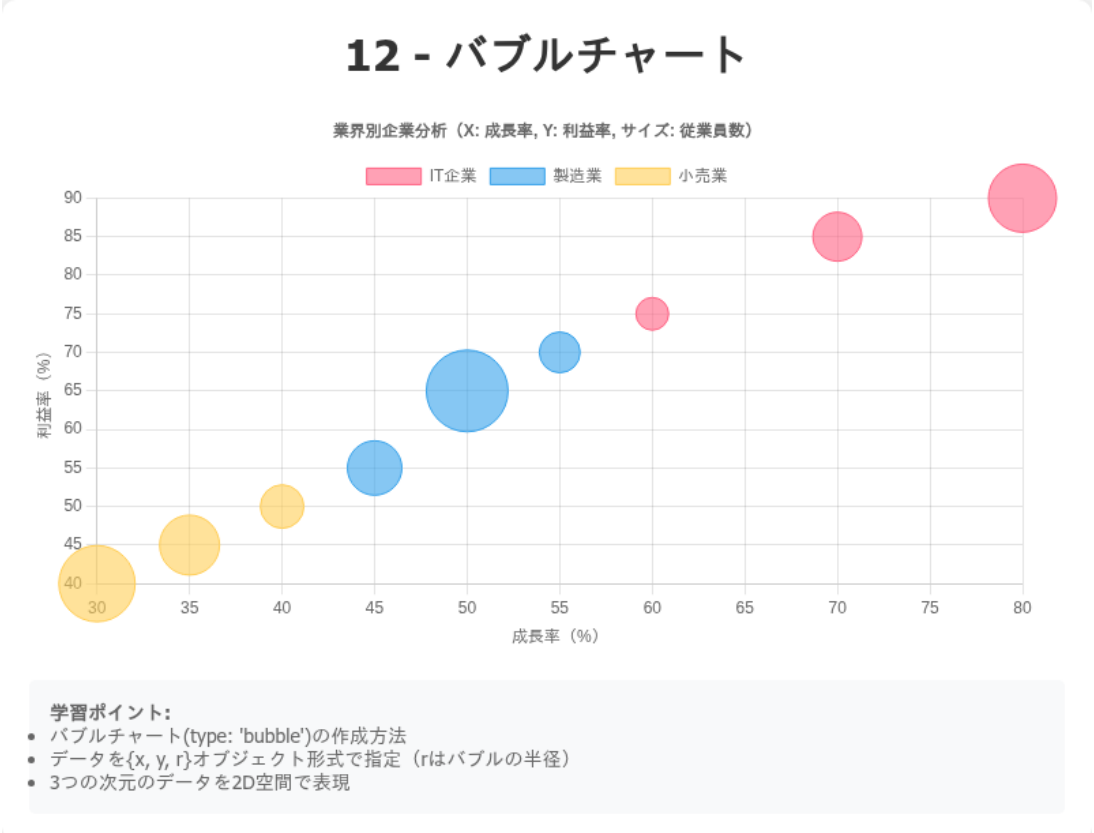

図 13.1　バブルチャートの実行結果

13.2 新しく学ぶ関数・プロパティ

13.2.1 type: 'bubble'

チャートタイプをバブルに設定します。

13.2.2 データ形式

バブルチャートでは、{x, y, r}オブジェクトでデータを指定します。

Listing 13.1 バブルチャートのデータ形式

```
data: [
    { x: 10, y: 20, r: 15 },  // r は半径
    { x: 15, y: 25, r: 10 },
    { x: 20, y: 30, r: 20 }
]
```

13.3 練習問題

問題 12-1

以下の要件を満たすバブルチャートを作成してください。

- データ：都市の人口、平均年収、面積の関係
  - 東京：{x: 950, y: 520, r: 25}（年収 520 万、人口 950 万、面積大）
  - 大阪：{x: 880, y: 480, r: 18}
  - 名古屋：{x: 230, y: 460, r: 12}
  - 福岡：{x: 160, y: 440, r: 10}
- 各バブルに異なる色を設定
- X 軸：人口（万人）、Y 軸：平均年収（万円）
- タイトル：「主要都市の比較」

模範解答 12-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bubble',
    data: {
        datasets: [{
            label: '都市データ',
            data: [
                {x: 950, y: 520, r: 25},
```

```
                {x: 880, y: 480, r: 18},
                {x: 230, y: 460, r: 12},
                {x: 160, y: 440, r: 10}
            ],
            backgroundColor: [
                'rgba(255, 99, 132, 0.6)',
                'rgba(54, 162, 235, 0.6)',
                'rgba(255, 206, 86, 0.6)',
                'rgba(75, 192, 192, 0.6)'
            ]
        }]
    },
    options: {
        scales: {
            x: { title: { display: true, text: '人口（万人）' } },
            y: { title: { display: true, text: '平均年収（万円）' }
                }
        },
        plugins: {
            title: { display: true, text: '主要都市の比較' }
        }
    }
});
```

第 14 章

極座標エリアチャート

14.1　学習目標

この章を学習すると、以下のことができるようになります。

- 極座標エリアチャート（`type: 'polarArea'`）を作成できる
- 各セグメントの半径で値を表現できる
- 円グラフと極座標エリアチャートの違いを理解する
- 放射状スケールの設定方法を理解する

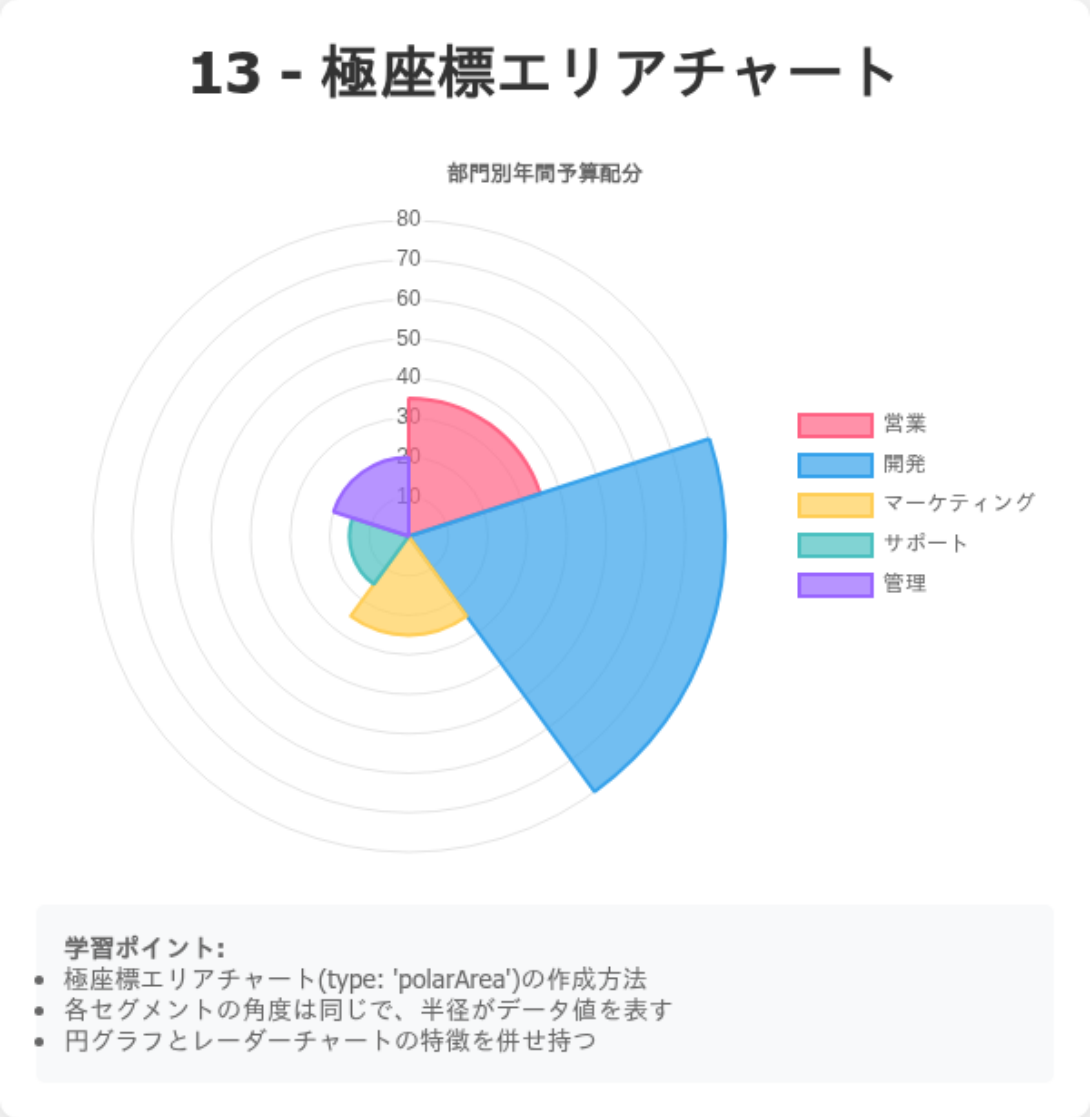

図 14.1　極座標エリアチャートの実行結果

14.2　新しく学ぶ関数・プロパティ

14.2.1　type: 'polarArea'

チャートタイプを極座標エリアに設定します。

14.3 練習問題

問題 13-1

以下の要件を満たす極座標エリアチャートを作成してください。

- ラベル：「スピード」「パワー」「テクニック」「スタミナ」「メンタル」
- データ：選手の能力値として 85, 70, 90, 75, 80
- 5 色を使用
- タイトル：「選手能力分析」

模範解答 13-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'polarArea',
    data: {
        labels: ['スピード', 'パワー', 'テクニック', 'スタミナ',
            'メンタル'],
        datasets: [{
            data: [85, 70, 90, 75, 80],
            backgroundColor: [
                'rgba(255, 99, 132, 0.7)',
                'rgba(54, 162, 235, 0.7)',
                'rgba(255, 206, 86, 0.7)',
                'rgba(75, 192, 192, 0.7)',
                'rgba(153, 102, 255, 0.7)'
            ]
        }]
    },
    options: {
        plugins: {
            title: { display: true, text: '選手能力分析' }
        }
    }
});
```

第 15 章

複合チャート

15.1　学習目標

この章を学習すると、以下のことができるようになります。

- 複数のチャートタイプを 1 つのグラフに組み合わせられる
- データセットごとに異なる `type` を指定できる
- `yAxisID` で複数の Y 軸を使い分けられる
- 棒グラフと折れ線グラフを効果的に組み合わせられる

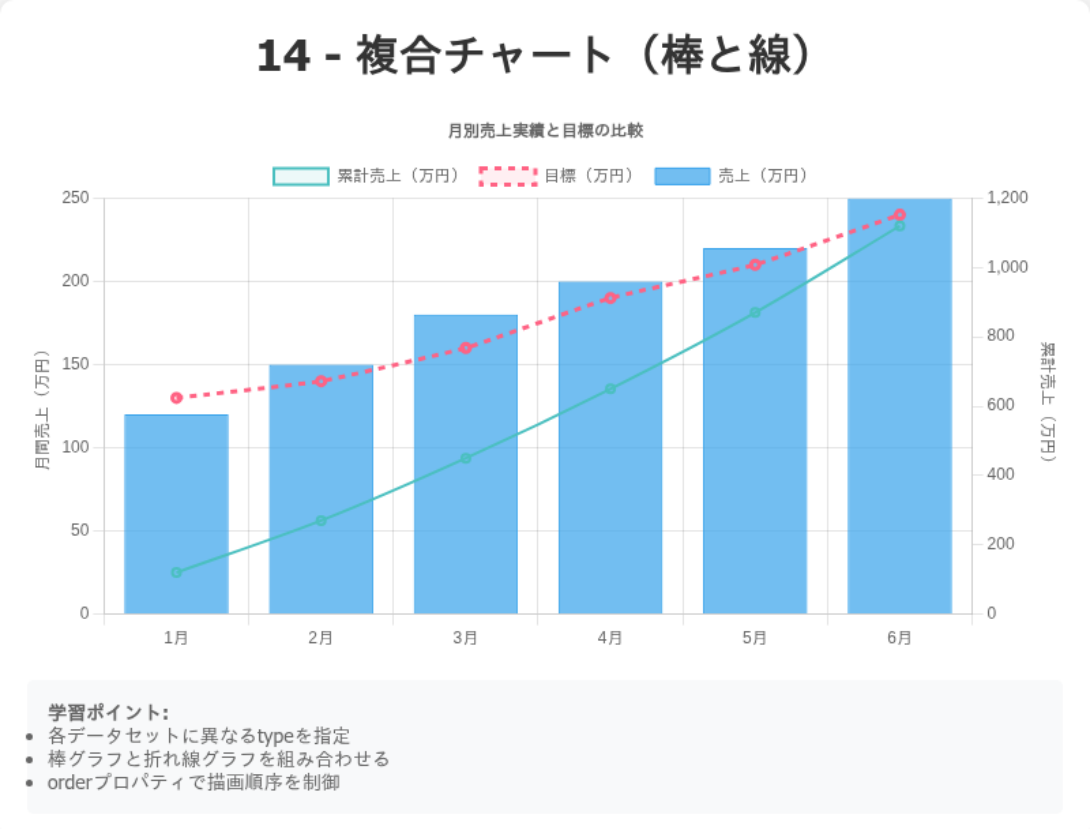

図 15.1　複合チャートの実行結果

15.2 新しく学ぶ関数・プロパティ

15.2.1 データセットごとの type 指定

各データセットに個別の type を指定できます。

Listing 15.1 複合チャート

```
datasets: [
    {
        type: 'bar',
        label: '売上',
        data: [100, 120, 140]
    },
    {
        type: 'line',
        label: 'トレンド',
        data: [90, 130, 150]
    }
]
```

15.2.2 yAxisID

データセットを特定の Y 軸に関連付けます。

```
datasets: [
    { yAxisID: 'y', data: [...] },   // 左の Y軸
    { yAxisID: 'y1', data: [...] }   // 右の Y軸
],
options: {
    scales: {
        y: { position: 'left' },
        y1: { position: 'right' }
    }
}
```

15.3 練習問題

問題 14-1

以下の要件を満たす複合チャートを作成してください。

- X 軸：「1 月」「2 月」「3 月」「4 月」「5 月」「6 月」
- データ：

  - 売上（棒グラフ、左 Y 軸）：120, 150, 180, 160, 200, 220
  - 利益率（折れ線、右 Y 軸）：15, 18, 22, 20, 25, 28
- 左 Y 軸：売上（万円）、右 Y 軸：利益率（%）
- タイトル：「売上と利益率の推移」

模範解答 14-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['1月', '2月', '3月', '4月', '5月', '6月'],
        datasets: [
            {
                type: 'bar',
                label: '売上',
                data: [120, 150, 180, 160, 200, 220],
                backgroundColor: 'rgba(54, 162, 235, 0.7)',
                yAxisID: 'y'
            },
            {
                type: 'line',
                label: '利益率',
                data: [15, 18, 22, 20, 25, 28],
                borderColor: 'rgb(255, 99, 132)',
                yAxisID: 'y1'
            }
        ]
    },
    options: {
        scales: {
            y: { position: 'left', title: { display: true, text: '売上（万円）' } },
            y1: { position: 'right', title: { display: true, text: '利益率（%）' }, grid: { drawOnChartArea: false } }
        },
        plugins: {
            title: { display: true, text: '売上と利益率の推移' }
        }
    }
});
```

第 16 章

ステップ折れ線グラフ

16.1　学習目標

この章を学習すると、以下のことができるようになります。

- `stepped` オプションでステップ折れ線を作成できる
- `'before'`、`'after'`、`'middle'` の違いを理解する
- 離散的なデータや段階的な変化を適切に表現できる
- 通常の折れ線とステップ折れ線の使い分けを判断できる

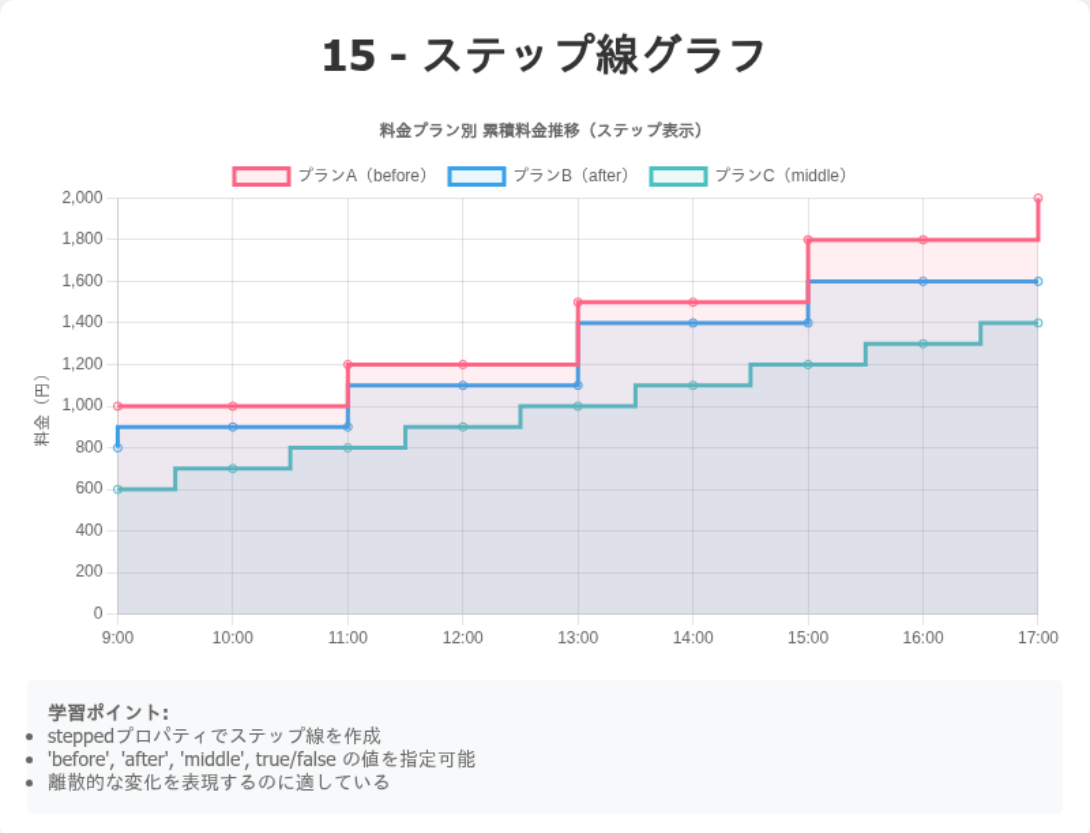

図 16.1　ステップ折れ線グラフの実行結果

16.2 新しく学ぶ関数・プロパティ

16.2.1 stepped

```
datasets: [{
    stepped: 'before'  // 'before', 'after', 'middle', true
}]
```

- 'before'：点の前でステップ
- 'after'：点の後でステップ
- 'middle'：点間の中央でステップ
- true：'before' と同等

16.3 練習問題

問題 15-1

以下の要件を満たすステップ折れ線グラフを作成してください。

- X 軸：「1 月」「4 月」「7 月」「10 月」「1 月」（次年度）
- データ：料金プランの価格変更として 980, 980, 1280, 1280, 1480
- ステップの種類：'after'（値が変わった後に上昇）
- 線の色：赤系統
- タイトル：「サブスクリプション料金の推移」

模範解答 15-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'line',
    data: {
        labels: ['1月', '4月', '7月', '10月', '1月'],
        datasets: [{
            label: '月額料金（円）',
            data: [980, 980, 1280, 1280, 1480],
            stepped: 'after',
            borderColor: 'rgb(255, 99, 132)',
            backgroundColor: 'rgba(255, 99, 132, 0.2)',
            fill: true
        }]
    },
```

```
    options: {
        plugins: {
            title: { display: true, text:
                'サブスクリプション料金の推移' }
        }
    }
});
```

第 17 章

グラデーション背景

17.1　学習目標

この章を学習すると、以下のことができるようになります。

- Canvas API の `createLinearGradient()` でグラデーションを作成できる
- `addColorStop()` でグラデーションの色を設定できる
- チャートの背景にグラデーションを適用できる
- Canvas 2D コンテキストの基本操作を理解する

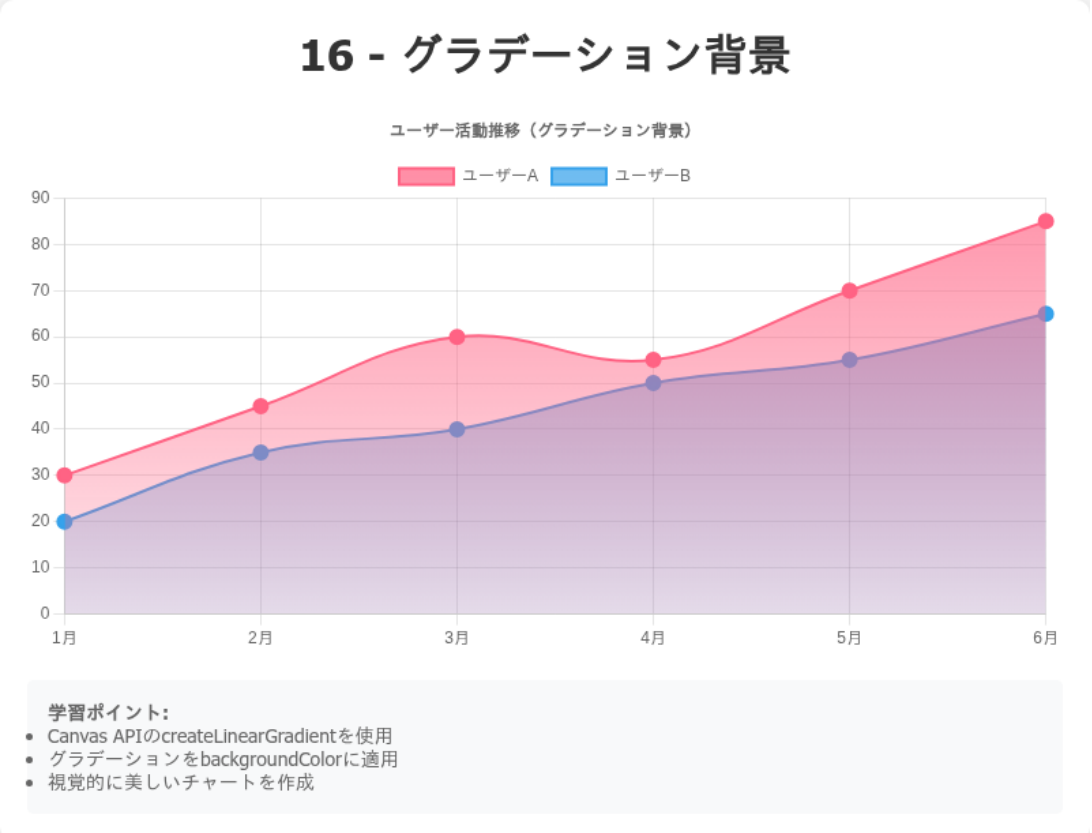

図 17.1　グラデーション背景の実行結果

17.2 新しく学ぶ関数・プロパティ

17.2.1 createLinearGradient

Canvas 2D コンテキストの `createLinearGradient()` メソッドで線形グラデーションを作成します。

Listing 17.1 グラデーション背景

```
const ctx = document.getElementById('myChart').getContext('2d');
const gradient = ctx.createLinearGradient(0, 0, 0, 400);
gradient.addColorStop(0, 'rgba(75, 192, 192, 0.8)');
gradient.addColorStop(1, 'rgba(75, 192, 192, 0.1)');

new Chart(ctx, {
    type: 'line',
    data: {
        datasets: [{
            backgroundColor: gradient,
            fill: true
        }]
    }
});
```

17.2.2 addColorStop

グラデーションに色を追加します。第 1 引数は位置（0–1）、第 2 引数は色です。

17.3 練習問題

問題 16-1

以下の要件を満たすグラデーション背景の折れ線グラフを作成してください。

- X 軸：「月」「火」「水」「木」「金」「土」「日」
- データ：アクセス数として 1200, 1500, 1800, 2200, 2000, 3500, 3000
- グラデーション：上から下へ紫からピンクに変化
- 塗りつぶしを有効化
- タイトル：「週間アクセス数」

模範解答 16-1

```
const ctx = document.getElementById('myChart').getContext('2d');
```

```
const gradient = ctx.createLinearGradient(0, 0, 0, 400);
gradient.addColorStop(0, 'rgba(153, 102, 255, 0.8)');
gradient.addColorStop(1, 'rgba(255, 99, 132, 0.2)');

new Chart(ctx, {
    type: 'line',
    data: {
        labels: ['月', '火', '水', '木', '金', '土', '日'],
        datasets: [{
            label: 'アクセス数',
            data: [1200, 1500, 1800, 2200, 2000, 3500, 3000],
            borderColor: 'rgb(153, 102, 255)',
            backgroundColor: gradient,
            fill: true,
            tension: 0.4
        }]
    },
    options: {
        plugins: {
            title: { display: true, text: '週間アクセス数' }
        }
    }
});
```

第 18 章

カスタムツールチップ

18.1　学習目標

この章を学習すると、以下のことができるようになります。

- `tooltip.callbacks` でツールチップの内容をカスタマイズできる
- `title`、`label`、`footer` コールバックを実装できる
- データに応じた動的なツールチップを作成できる
- 数値フォーマットや単位の追加ができる

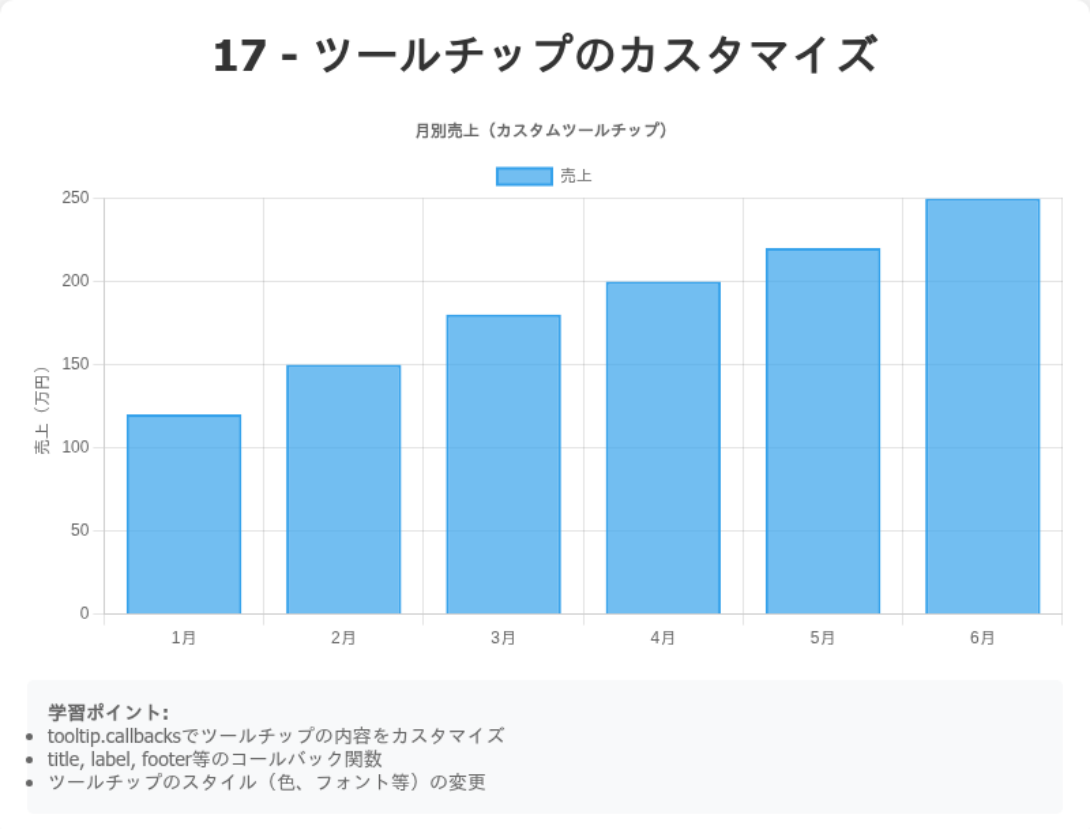

図 18.1　カスタムツールチップの実行結果

18.2 新しく学ぶ関数・プロパティ

18.2.1 tooltip.callbacks

Listing 18.1 ツールチップのカスタマイズ

```
options: {
    plugins: {
        tooltip: {
            callbacks: {
                title: (items) => `詳細: ${items[0].label}`,
                label: (context) => {
                    const value = context.raw;
                    return `値: ${value}円`;
                },
                footer: () => '※ 税込価格'
            }
        }
    }
}
```

主なコールバック：

- `title`：ツールチップのタイトル
- `label`：各データセットのラベル
- `footer`：フッターテキスト
- `beforeBody`, `afterBody`：本文の前後

18.3 練習問題

問題 17-1

以下の要件を満たすカスタムツールチップを実装してください。

- 棒グラフで商品売上を表示
- ツールチップのタイトル：「売上詳細」
- ツールチップのラベル：値に「個」の単位を追加
- ツールチップのフッター：「※ 税抜価格」

模範解答 17-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
```

```
    type: 'bar',
    data: {
        labels: ['商品A', '商品B', '商品C', '商品D'],
        datasets: [{
            label: '売上個数',
            data: [150, 230, 180, 290],
            backgroundColor: 'rgba(54, 162, 235, 0.7)'
        }]
    },
    options: {
        plugins: {
            tooltip: {
                callbacks: {
                    title: () => '売上詳細',
                    label: (context) => `${context.dataset.label}:
                        ${context.raw}個`,
                    footer: () => '※ 税抜価格'
                }
            }
        }
    }
});
```

第 19 章

カスタム凡例

19.1　学習目標

この章を学習すると、以下のことができるようになります。

- `legend.display: false` でデフォルト凡例を非表示にできる
- HTML/CSS で独自の凡例を作成できる
- `toggleDataVisibility()` でデータの表示/非表示を切り替えられる
- `chart.update()` でチャートを再描画できる

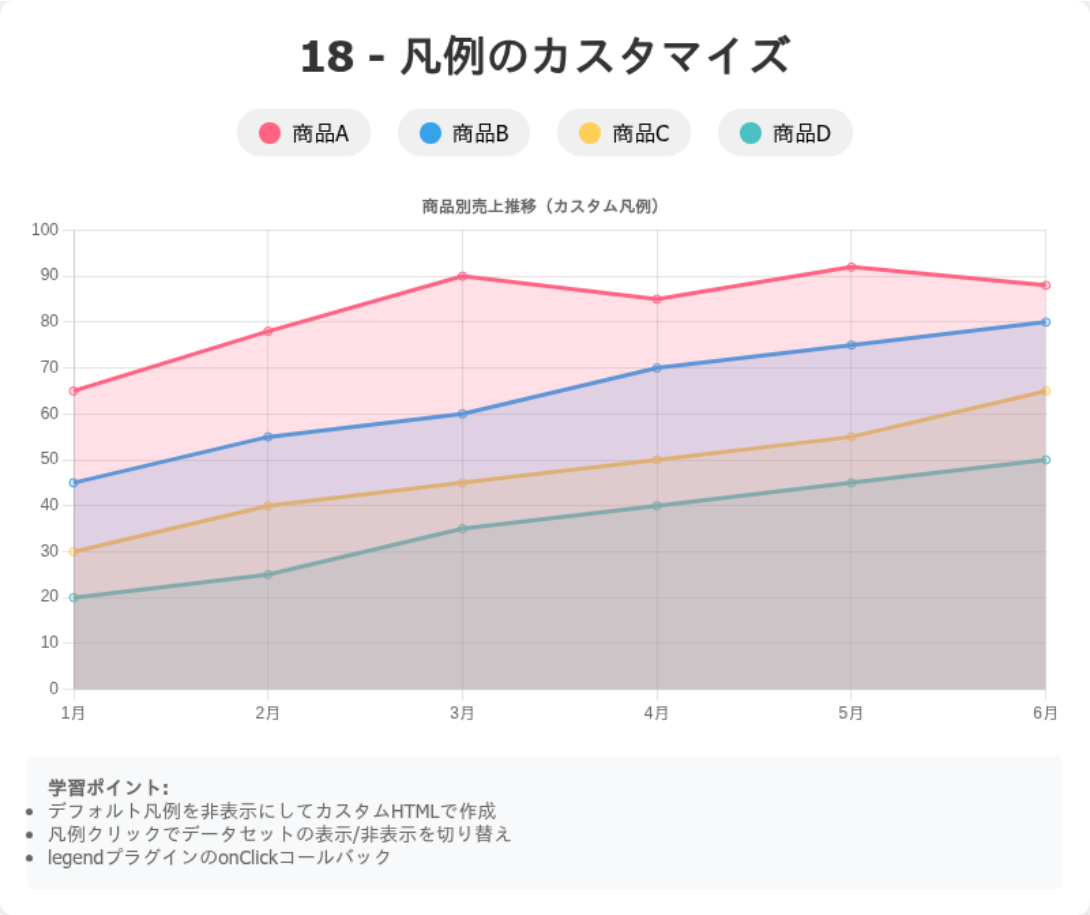

図 19.1　カスタム凡例の実行結果

19.2　新しく学ぶ関数・プロパティ

19.2.1　legend.display

`legend.display: false` でデフォルト凡例を非表示にします。

19.2.2　generateLabels

Listing 19.1　カスタム凡例生成

```
options: {
    plugins: {
        legend: {
            display: false  // デフォルト凡例を非表示
        }
    }
}

// HTMLで独自の凡例を作成
const legendHTML = chart.data.labels.map((label, i) => {
```

```
    const color = chart.data.datasets[0].backgroundColor[i];
    return `<div onclick="toggleData(${i})">${label}</div>`;
}).join('');
```

19.2.3 toggleDataVisibility / update

```
function toggleData(index) {
    chart.toggleDataVisibility(index);  // データ表示切替
    chart.update();  // チャート更新
}
```

19.3 練習問題

問題 18-1

以下の要件を満たすカスタム凡例を実装してください。

- ドーナツチャートで部門別売上を表示
- デフォルトの凡例を非表示にする
- HTML でカスタム凡例を作成（各項目にクリックイベント）
- クリックで該当データの表示/非表示を切り替え

模範解答 18-1

```
const ctx = document.getElementById('myChart').getContext('2d');
const chart = new Chart(ctx, {
    type: 'doughnut',
    data: {
        labels: ['営業部', '開発部', '管理部', 'マーケティング部'],
        datasets: [{
            data: [35, 40, 15, 10],
            backgroundColor: ['#FF6384', '#36A2EB', '#FFCE56',
                '#4BC0C0']
        }]
    },
    options: {
        plugins: {
            legend: { display: false }
        }
    }
});

// カスタム凡例を生成
```

```
const legendContainer = document.getElementById('legend');
chart.data.labels.forEach((label, i) => {
    const color = chart.data.datasets[0].backgroundColor[i];
    const div = document.createElement('div');
    div.innerHTML = `<span
        style="background:${color}"></span>${label}`;
    div.onclick = () => {
        chart.toggleDataVisibility(i);
        chart.update();
    };
    legendContainer.appendChild(div);
});
```

第 20 章

カスタム軸

20.1　学習目標

この章を学習すると、以下のことができるようになります。

- `ticks.callback` で軸ラベルをカスタマイズできる
- 通貨記号やパーセント表示などのフォーマットができる
- `grid` オプションでグリッド線のスタイルを設定できる
- 軸タイトルの表示と位置を制御できる

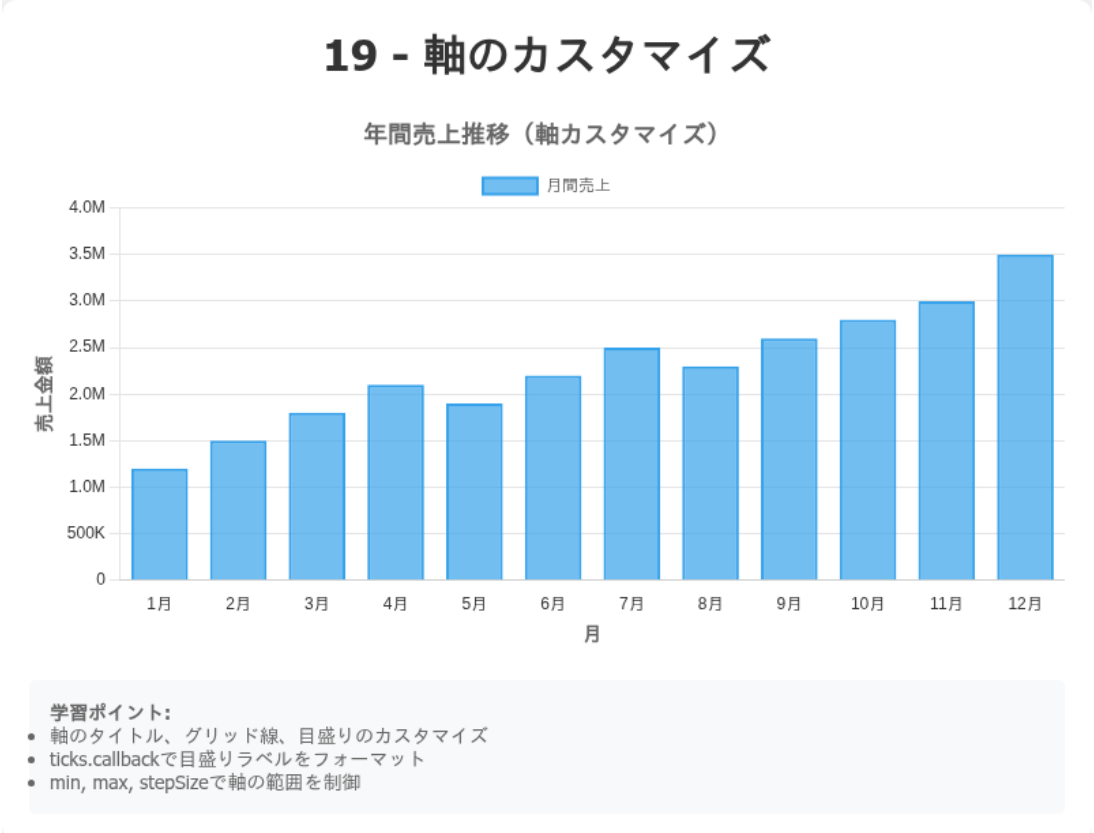

図 20.1　カスタム軸の実行結果

20.2 新しく学ぶ関数・プロパティ

20.2.1 ticks.callback

Listing 20.1 軸ラベルのカスタマイズ

```
options: {
    scales: {
        y: {
            ticks: {
                callback: function(value) {
                    return '¥' + value.toLocaleString();
                }
            }
        }
    }
}
```

20.2.2 grid

```
scales: {
    y: {
        grid: {
            color: 'rgba(0, 0, 0, 0.1)',
            lineWidth: 1,
            drawBorder: false
        }
    }
}
```

20.3 練習問題

問題 19-1

以下の要件を満たすカスタム軸の棒グラフを作成してください。

- X 軸：商品カテゴリ 5 つ
- データ：売上金額（万円単位）
- Y 軸のラベルに「万円」を追加表示
- グリッド線を点線スタイル（`borderDash: [5, 5]`）に設定
- Y 軸の軸タイトル：「売上金額」

模範解答 19-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['食品', '家電', '衣類', '雑貨', '書籍'],
        datasets: [{
            label: '売上',
            data: [250, 180, 120, 90, 60],
            backgroundColor: 'rgba(75, 192, 192, 0.7)'
        }]
    },
    options: {
        scales: {
            y: {
                title: { display: true, text: '売上金額' },
                ticks: {
                    callback: (value) => value + '万円'
                },
                grid: {
                    color: 'rgba(0, 0, 0, 0.1)',
                    borderDash: [5, 5]
                }
            }
        }
    }
});
```

第 21 章

データラベル

21.1　学習目標

この章を学習すると、以下のことができるようになります。

- chartjs-plugin-datalabels プラグインを導入できる
- データポイントに直接ラベルを表示できる
- `formatter` でラベルの表示形式をカスタマイズできる
- `anchor` と `align` でラベルの位置を調整できる

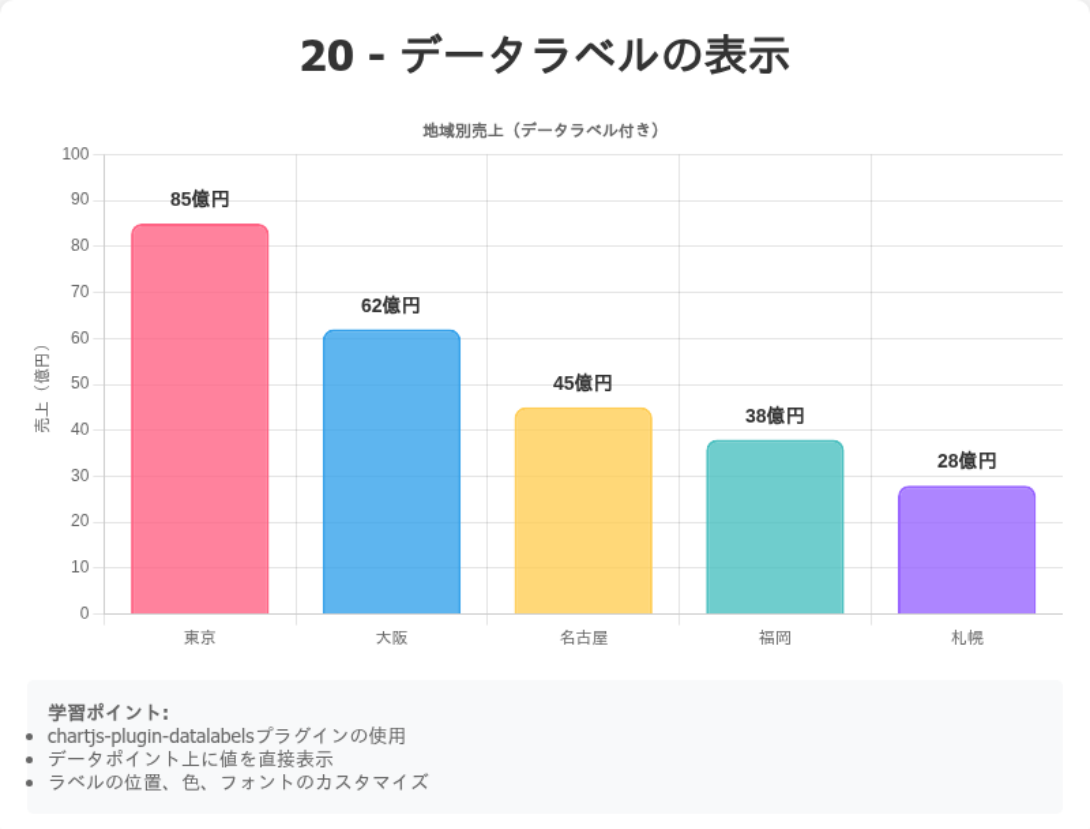

図 21.1　データラベルの実行結果

21.2 新しく学ぶ関数・プロパティ

21.2.1 プラグインの読み込み

Chart.js 本体の後にプラグインを読み込みます。バージョンは固定を推奨します。

```
<!-- Chart.js本体の後に読み込む -->
<script
    src="https://cdn.jsdelivr.net/npm/chartjs-plugin-datalabels@2.2.0"></script>
```

21.2.2 datalabels オプション

Listing 21.1 データラベル設定

```
options: {
    plugins: {
        datalabels: {
            color: '#fff',
            font: { weight: 'bold' },
            formatter: (value) => value + '%',
            anchor: 'end',
            align: 'top'
        }
    }
}
```

21.3 練習問題

問題 20-1

以下の要件を満たすデータラベル付き棒グラフを作成してください。

- chartjs-plugin-datalabels を使用
- X 軸：月名（1 月〜6 月）
- データ：達成率として 85, 92, 78, 95, 88, 100
- データラベルに「%」を追加
- ラベルの位置：バーの中央（`anchor: 'center'`）
- ラベルの色：白

模範解答 20-1

```
// プラグインを登録
Chart.register(ChartDataLabels);

const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['1月', '2月', '3月', '4月', '5月', '6月'],
        datasets: [{
            label: '達成率',
            data: [85, 92, 78, 95, 88, 100],
            backgroundColor: 'rgba(54, 162, 235, 0.8)'
        }]
    },
    options: {
        plugins: {
            datalabels: {
                color: '#fff',
                font: { weight: 'bold', size: 14 },
                formatter: (value) => value + '%',
                anchor: 'center',
                align: 'center'
            },
            title: { display: true, text: '月別目標達成率' }
        }
    }
});
```

第 III 部

上級レベル（21–30）

第 22 章

リアルタイムデータ更新

22.1　学習目標

この章を学習すると、以下のことができるようになります。

- `setInterval()` で定期的にデータを更新できる
- `push()` と `shift()` でデータを追加・削除できる
- `chart.update()` でチャートを再描画できる
- リアルタイムモニタリング画面を実装できる

図 22.1 リアルタイムデータ更新の実行結果

22.2 新しく学ぶ関数・プロパティ

22.2.1 データの追加

Listing 22.1 リアルタイム更新

```
function addData() {
    chart.data.labels.push(newLabel);
    chart.data.datasets[0].data.push(newValue);

    // データ数が上限を超えたら古いデータを削除
    if (chart.data.labels.length > maxPoints) {
        chart.data.labels.shift();
        chart.data.datasets[0].data.shift();
    }

```

```
    chart.update('none');  // アニメーションなしで更新
}

setInterval(addData, 1000);  // 1秒ごとに実行
```

22.2.2 chart.update()

チャートを再描画します。引数に'none' を渡すとアニメーションなしで即座に更新されます。

22.2.3 setInterval / clearInterval

```
const intervalId = setInterval(func, 1000);  // 開始
clearInterval(intervalId);  // 停止
```

22.3 練習問題

問題 21-1

以下の要件を満たすリアルタイム更新チャートを作成してください。

- 折れ線グラフで CPU 使用率をシミュレーション表示
- 1 秒ごとに新しいデータ（0〜100 のランダム値）を追加
- 表示するデータポイントは最大 20 個
- 古いデータは自動的に削除
- 開始/停止ボタンを実装

模範解答 21-1

```
const ctx = document.getElementById('myChart').getContext('2d');
const chart = new Chart(ctx, {
    type: 'line',
    data: {
        labels: [],
        datasets: [{
            label: 'CPU使用率 (%)',
            data: [],
            borderColor: 'rgb(75, 192, 192)',
            tension: 0.1
        }]
    },
    options: { animation: false }
});
```

```
15
16 let intervalId = null;
17 const maxPoints = 20;
18
19 function addData() {
20     const now = new Date().toLocaleTimeString();
21     const value = Math.floor(Math.random() * 100);
22
23     chart.data.labels.push(now);
24     chart.data.datasets[0].data.push(value);
25
26     if (chart.data.labels.length > maxPoints) {
27         chart.data.labels.shift();
28         chart.data.datasets[0].data.shift();
29     }
30     chart.update('none');
31 }
32
33 // 多重起動防止のガードを追加
34 function start() {
35     if (intervalId !== null) return; // 既に動作中なら何もしない
36     intervalId = setInterval(addData, 1000);
37 }
38
39 function stop() {
40     if (intervalId === null) return;
41     clearInterval(intervalId);
42     intervalId = null; // 停止後はnullに戻す
43 }
```

よくあるミス：多重起動

開始ボタンを複数回押すと `setInterval` が重複して実行され、更新速度が加速してしまいます。必ず多重起動防止のガードを入れましょう。

第 23 章

ドリルダウン

23.1　学習目標

この章を学習すると、以下のことができるようになります。

- `onClick` イベントでクリックされた要素を検出できる
- `chart.destroy()` で既存のチャートを破棄できる
- クリックに応じて詳細チャートに遷移するドリルダウンを実装できる
- 階層的なデータナビゲーションを構築できる

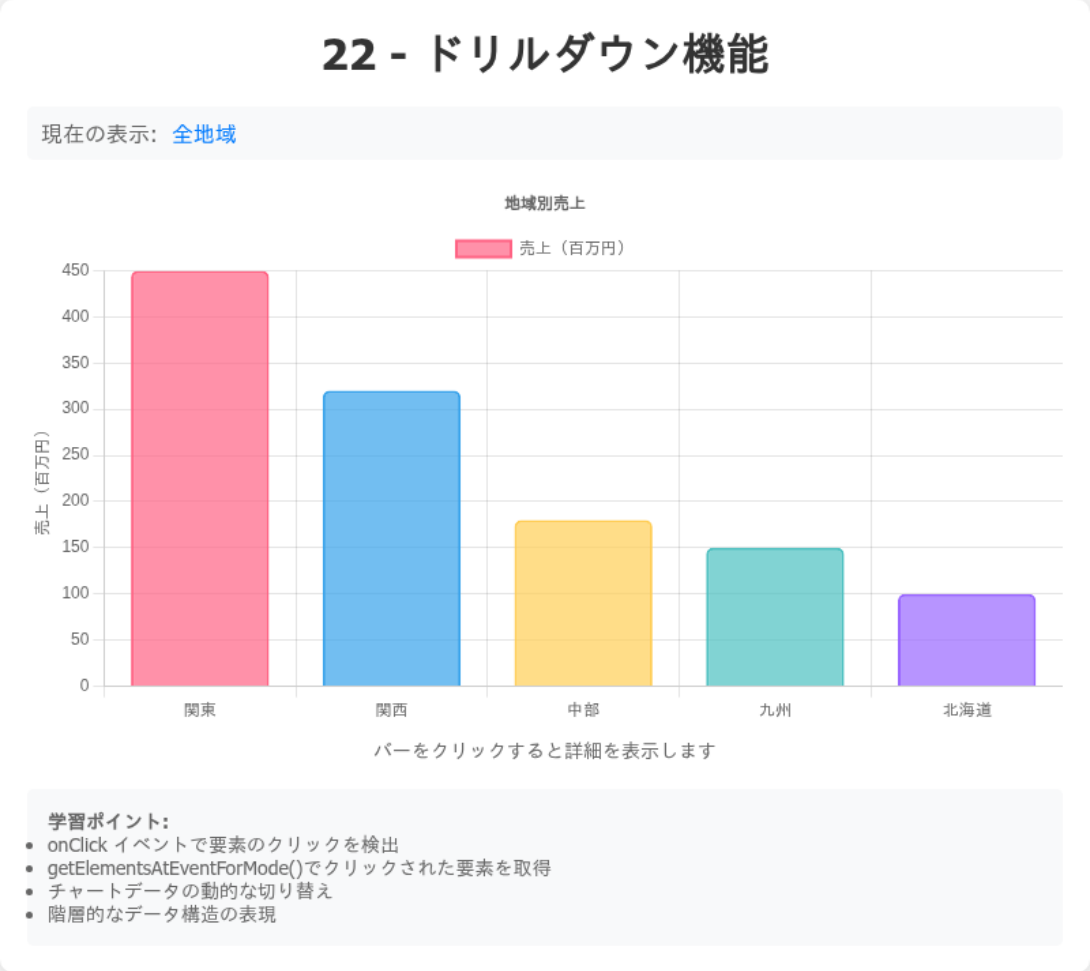

図 23.1 ドリルダウンの実行結果

23.2 新しく学ぶ関数・プロパティ

23.2.1 onClick

Listing 23.1 クリックイベント

```
options: {
    onClick: (event, elements, chart) => {
        if (elements.length > 0) {
            const index = elements[0].index;
            showDetailChart(index);
        }
    }
}
```

23.2.2 chart.destroy()

既存のチャートを破棄してから新しいチャートを作成します。

```
if (currentChart) {
    currentChart.destroy();
}
currentChart = new Chart(ctx, newConfig);
```

23.3 練習問題

問題 22-1

以下の要件を満たすドリルダウンチャートを作成してください。

- 初期表示：年間売上（円グラフ：Q1〜Q4）
- 各四半期をクリックすると月別詳細（棒グラフ）に切り替え
- 「戻る」ボタンで年間表示に戻る
- chart.destroy() で既存チャートを破棄してから新規作成

模範解答 22-1

```
const ctx = document.getElementById('myChart').getContext('2d');
let currentChart = null;

const yearlyData = { labels: ['Q1', 'Q2', 'Q3', 'Q4'], data: [280, 320, 350, 400] };
const quarterlyDetail = {
    Q1: { labels: ['1月', '2月', '3月'], data: [90, 95, 95] },
    Q2: { labels: ['4月', '5月', '6月'], data: [100, 110, 110] },
    Q3: { labels: ['7月', '8月', '9月'], data: [120, 115, 115] },
    Q4: { labels: ['10月', '11月', '12月'], data: [130, 135, 135] }
};

function showYearlyChart() {
    if (currentChart) currentChart.destroy();
    currentChart = new Chart(ctx, {
        type: 'pie',
        data: { labels: yearlyData.labels, datasets: [{ data: yearlyData.data }] },
        options: {
            onClick: (e, elements) => {
                if (elements.length > 0) {
```

```
                    const quarter =
                        yearlyData.labels[elements[0].index];
                    showQuarterlyChart(quarter);
                }
            }
        }
    });
}

function showQuarterlyChart(quarter) {
    if (currentChart) currentChart.destroy();
    const detail = quarterlyDetail[quarter];
    currentChart = new Chart(ctx, {
        type: 'bar',
        data: { labels: detail.labels, datasets: [{ label: quarter,
            data: detail.data }] }
    });
}

showYearlyChart();
```

第 24 章

カスタムプラグイン

24.1　学習目標

この章を学習すると、以下のことができるようになります。

- Chart.js プラグインの基本構造を理解する
- `beforeDraw` と `afterDraw` フックを実装できる
- `Chart.register()` でプラグインをグローバル登録できる
- 中央テキストや背景パターンなど独自の描画を追加できる

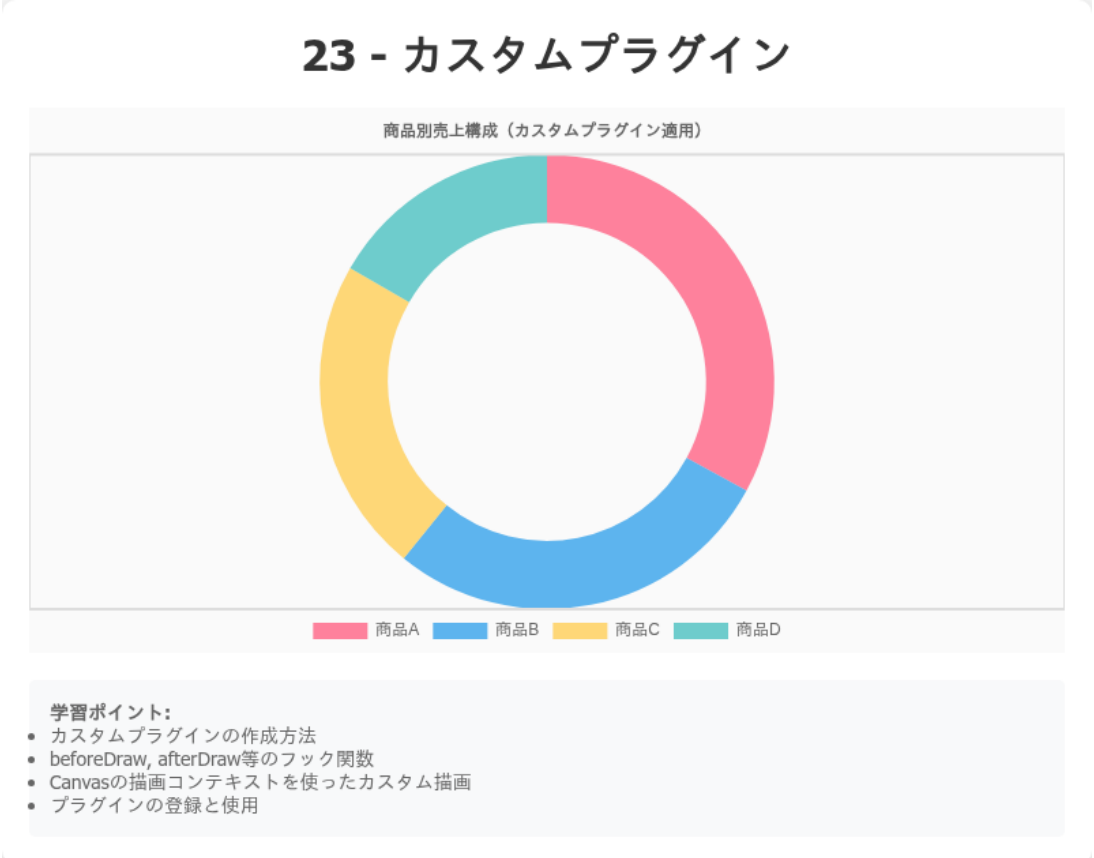

図 24.1　カスタムプラグインの実行結果

24.2 新しく学ぶ関数・プロパティ

24.2.1 プラグインの構造

Listing 24.1 カスタムプラグイン

```
const myPlugin = {
    id: 'myPlugin',
    beforeDraw: (chart) => {
        // チャート描画前に実行
    },
    afterDraw: (chart) => {
        // チャート描画後に実行
    }
};

Chart.register(myPlugin);  // グローバル登録
// または
new Chart(ctx, { plugins: [myPlugin] });  // 個別登録
```

24.2.2 主なフック

- `beforeDraw`：描画前
- `afterDraw`：描画後
- `beforeDatasetsDraw`：データセット描画前
- `afterDatasetsDraw`：データセット描画後

24.3 練習問題

問題 23-1

以下の要件を満たすカスタムプラグインを作成してください。

- ドーナツチャートの中央にテキストを表示するプラグイン
- 表示内容：「合計」とデータの合計値
- フォントサイズ：20px
- `afterDraw` フックを使用

模範解答 23-1

```
const centerTextPlugin = {
    id: 'centerText',
```

```
    afterDraw: (chart) => {
        const { ctx, chartArea } = chart;
        const total = chart.data.datasets[0].data.reduce((a, b) =>
            a + b, 0);

        // chartAreaはキャンバス左上(0,0)基準ではないため、
        // left/right, top/bottomから中心を計算する
        const centerX = (chartArea.left + chartArea.right) / 2;
        const centerY = (chartArea.top + chartArea.bottom) / 2;

        ctx.save();
        ctx.font = 'bold 20px Arial';
        ctx.textAlign = 'center';
        ctx.textBaseline = 'middle';
        ctx.fillStyle = '#333';

        ctx.fillText('合計', centerX, centerY - 15);
        ctx.fillText(total.toLocaleString(), centerX, centerY + 15);
        ctx.restore();
    }
};

const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'doughnut',
    data: {
        labels: ['A', 'B', 'C'],
        datasets: [{ data: [300, 200, 100] }]
    },
    plugins: [centerTextPlugin]
});
```

第 25 章

カスタムアニメーション

25.1　学習目標

この章を学習すると、以下のことができるようになります。

- `animation.duration` でアニメーション時間を設定できる
- `animation.easing` でイージング関数を選択できる
- `animation.delay` でデータごとの遅延を設定できる
- 効果的なアニメーションでユーザー体験を向上できる

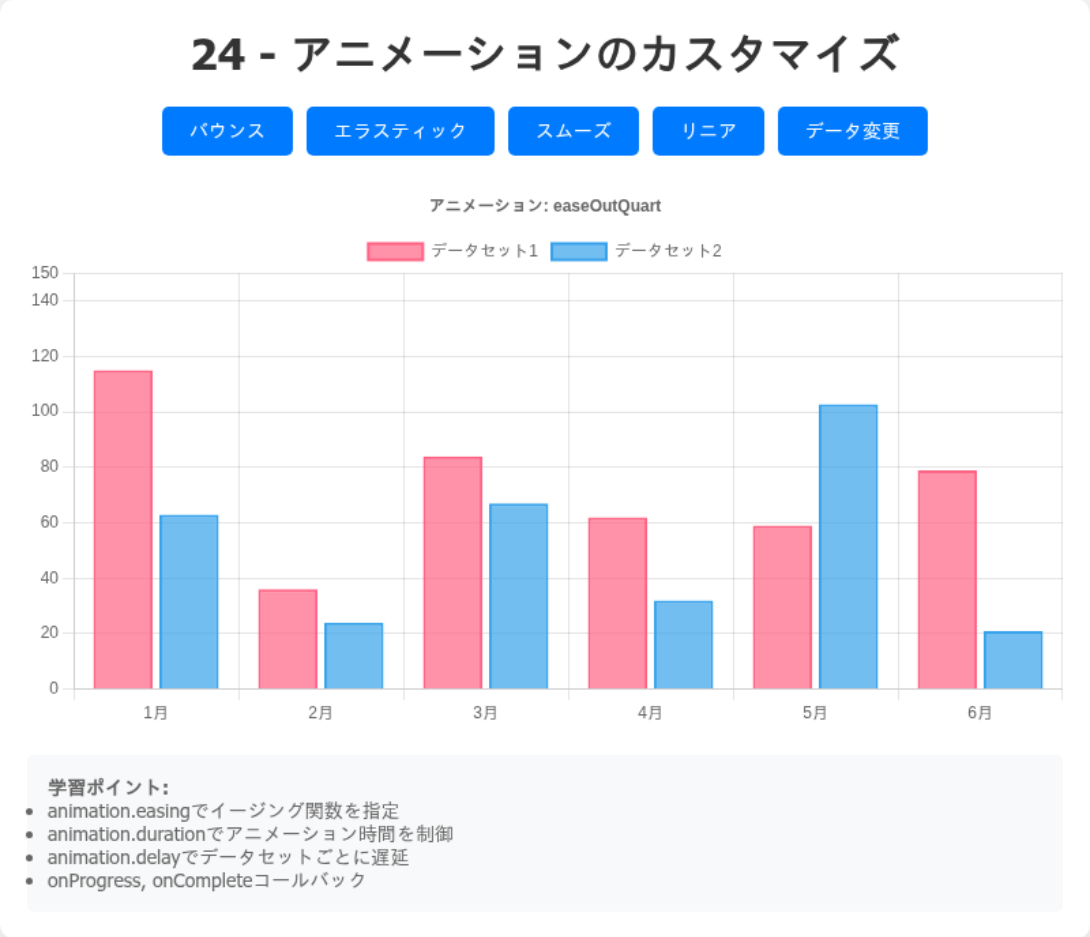

図 25.1　カスタムアニメーションの実行結果

25.2　新しく学ぶ関数・プロパティ

25.2.1　animation オプション

Listing 25.1　アニメーション設定

```
options: {
    animation: {
        duration: 2000,       // アニメーション時間（ミリ秒）
        easing: 'easeOutBounce',  // イージング関数
        delay: (context) => context.dataIndex * 100  // 遅延
    }
}
```

25.2.2　easing

利用可能なイージング関数：

- linear, easeInQuad, easeOutQuad, easeInOutQuad
- easeInCubic, easeOutCubic, easeInOutCubic
- easeInBounce, easeOutBounce, easeInOutBounce
- その他多数

25.3 練習問題

問題 24-1

以下の要件を満たすカスタムアニメーションを作成してください。

- 棒グラフで売上データを表示
- アニメーション時間：1500ms
- イージング：easeOutBounce
- 各バーが順番にアニメーション（遅延を追加）

模範解答 24-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['1月', '2月', '3月', '4月', '5月', '6月'],
        datasets: [{
            label: '売上',
            data: [120, 150, 180, 200, 175, 220],
            backgroundColor: 'rgba(54, 162, 235, 0.7)'
        }]
    },
    options: {
        animation: {
            duration: 1500,
            easing: 'easeOutBounce',
            delay: (context) => {
                return context.dataIndex * 200;
            }
        },
        plugins: {
            title: { display: true, text:
                '月別売上（バウンスアニメーション）' }
        }
    }
});
```

第 26 章

複数 Y 軸

26.1　学習目標

この章を学習すると、以下のことができるようになります。

- 複数の Y 軸（y、y1 など）を定義できる
- `yAxisID` でデータセットを特定の軸に関連付けられる
- 左右に異なるスケールの軸を配置できる
- 単位の異なるデータを 1 つのチャートで比較できる

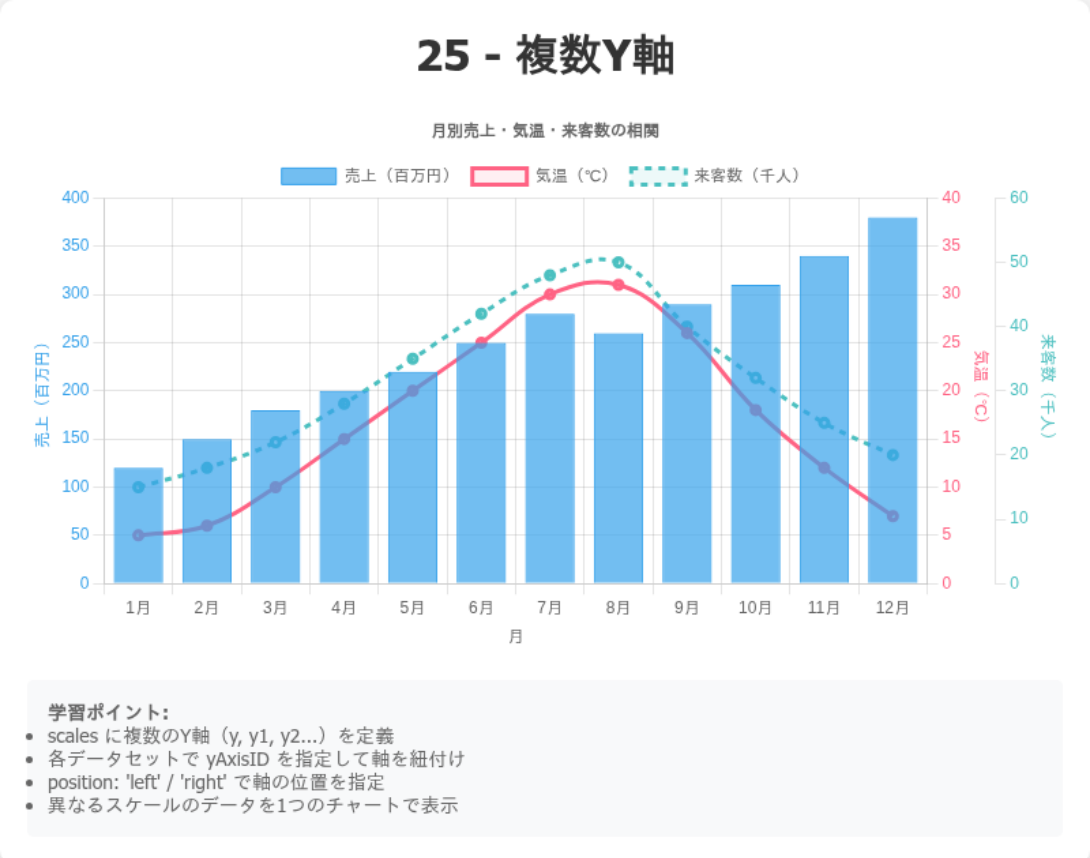

図 26.1　複数 Y 軸の実行結果

26.2 新しく学ぶ関数・プロパティ

26.2.1 複数軸の定義

Listing 26.1 複数 Y 軸

```
options: {
    scales: {
        y: {
            type: 'linear',
            position: 'left',
            title: { display: true, text: '売上（万円）' }
        },
        y1: {
            type: 'linear',
            position: 'right',
            title: { display: true, text: '気温（℃）' },
            grid: { drawOnChartArea: false }
        }
    }
},
datasets: [
    { yAxisID: 'y', data: [...] },
    { yAxisID: 'y1', data: [...] }
]
```

26.3 練習問題

問題 25-1

以下の要件を満たす複数 Y 軸のチャートを作成してください。

- X 軸：1 月〜12 月
- 左 Y 軸：販売数量（棒グラフ）
- 右 Y 軸：平均単価（折れ線グラフ）
- 左軸に「個」、右軸に「円」の単位表示
- 右軸のグリッド線を非表示

模範解答 25-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
```

```
    data: {
        labels: ['1月', '2月', '3月', '4月', '5月', '6月'],
        datasets: [
            {
                type: 'bar',
                label: '販売数量',
                data: [120, 150, 180, 160, 200, 220],
                backgroundColor: 'rgba(54, 162, 235, 0.7)',
                yAxisID: 'y'
            },
            {
                type: 'line',
                label: '平均単価',
                data: [2500, 2400, 2600, 2550, 2700, 2650],
                borderColor: 'rgb(255, 99, 132)',
                yAxisID: 'y1'
            }
        ]
    },
    options: {
        scales: {
            y: {
                position: 'left',
                title: { display: true, text: '販売数量' },
                ticks: { callback: (v) => v + '個' }
            },
            y1: {
                position: 'right',
                title: { display: true, text: '平均単価' },
                ticks: { callback: (v) => v + '円' },
                grid: { drawOnChartArea: false }
            }
        }
    }
});
```

第 27 章

HTML カスタムツールチップ

27.1　学習目標

この章を学習すると、以下のことができるようになります。

- `tooltip.enabled: false` でデフォルトツールチップを無効化できる
- `tooltip.external` で HTML ツールチップハンドラーを実装できる
- HTML/CSS でリッチなツールチップをデザインできる
- ツールチップの位置を `caretX`、`caretY` で制御できる

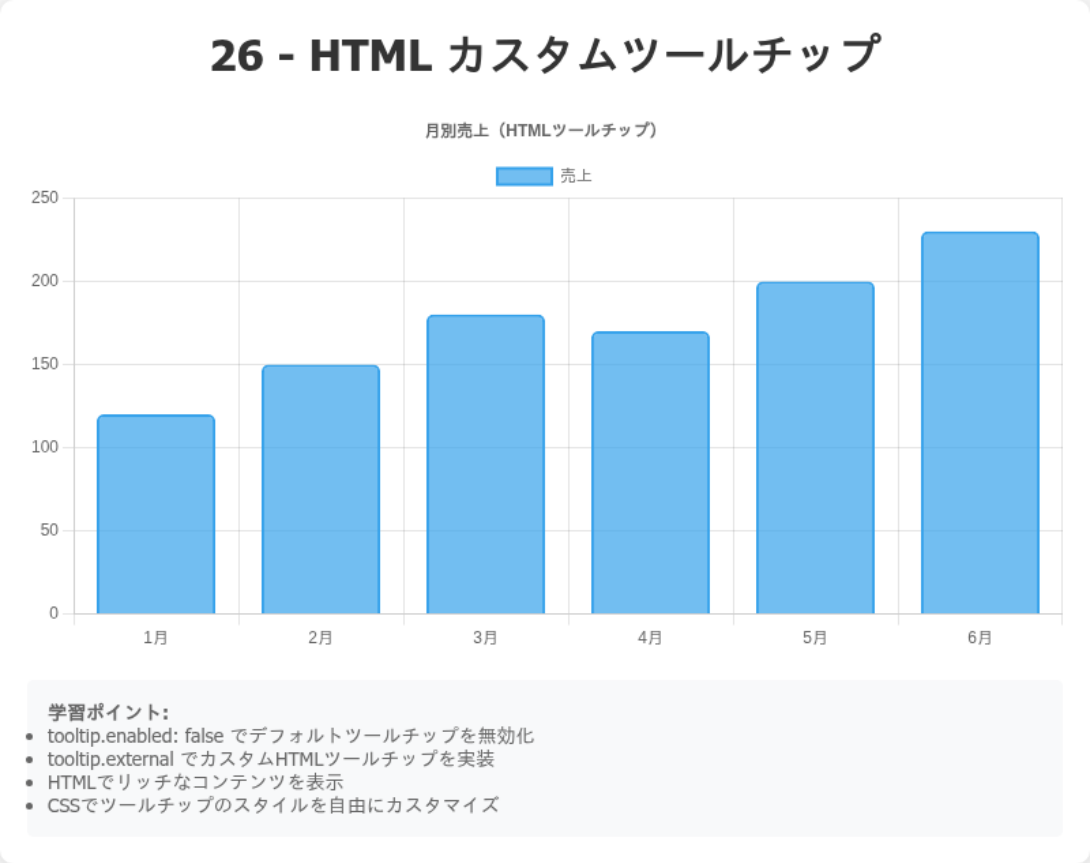

図 27.1　HTML カスタムツールチップの実行結果

27.2 新しく学ぶ関数・プロパティ

27.2.1 tooltip.external

座標計算の注意点

`tooltip.caretX/caretY` は canvas 内の座標です。ページ内に余白やスクロールがある場合、canvas の位置を `getBoundingClientRect()` で取得して補正する必要があります。

Listing 27.1 外部ツールチップ

```
options: {
    plugins: {
        tooltip: {
            enabled: false,  // デフォルトを無効化
            external: function(context) {
                const { chart, tooltip } = context;
                const tooltipEl = document.getElementById('tooltip');

                if (tooltip.opacity === 0) {
                    tooltipEl.style.opacity = 0;
                    return;
                }

                // canvasの位置を取得して座標を補正
                const { left, top } =
                    chart.canvas.getBoundingClientRect();

                // HTML内容を設定
                tooltipEl.innerHTML = `<div>${tooltip.title}</div>`;
                tooltipEl.style.left = (left + window.scrollX +
                    tooltip.caretX) + 'px';
                tooltipEl.style.top = (top + window.scrollY +
                    tooltip.caretY) + 'px';
                tooltipEl.style.opacity = 1;
            }
        }
    }
}
```

27.3 練習問題

問題 26-1

以下の要件を満たす HTML カスタムツールチップを作成してください。

- デフォルトツールチップを無効化
- HTML 要素でツールチップを表示
- カード形式のデザイン（角丸、影付き）
- ラベルと値を 2 行で表示
- ツールチップの位置をマウス位置に追従

模範解答 26-1

```
// HTML: <div id="tooltip" class="custom-tooltip"></div>
// CSS: .custom-tooltip { position: absolute; background: white;
//      border-radius: 8px; box-shadow: 0 4px 12px rgba(0,0,0,0.15);
//      padding: 10px; pointer-events: none; opacity: 0; }

const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: { labels: ['A', 'B', 'C'], datasets: [{ data: [10, 20,
       30] }] },
    options: {
        plugins: {
            tooltip: {
                enabled: false,
                external: function(context) {
                    const { chart, tooltip } = context;
                    const tooltipEl =
                       document.getElementById('tooltip');

                    if (tooltip.opacity === 0) {
                        tooltipEl.style.opacity = 0;
                        return;
                    }

                    // canvasの位置を取得して座標を補正
                    const { left, top } =
                       chart.canvas.getBoundingClientRect();

                    const label = tooltip.dataPoints[0].label;
                    const value = tooltip.dataPoints[0].raw;

```

```
                    tooltipEl.innerHTML = `
                        <div style="font-weight:
                            bold;">${label}</div>
                        <div>値: ${value}</div>
                    `;
                    tooltipEl.style.left = (left + window.scrollX +
                        tooltip.caretX) + 'px';
                    tooltipEl.style.top = (top + window.scrollY +
                        tooltip.caretY) + 'px';
                    tooltipEl.style.opacity = 1;
                }
            }
        }
    }
});
```

第 28 章

クリックイベント処理

28.1　学習目標

この章を学習すると、以下のことができるようになります。

- `onClick` でクリックイベントを処理できる
- `elements` 配列からクリックされたデータを取得できる
- `onHover` でホバーイベントを処理できる
- クリックに応じて外部要素（情報パネルなど）を更新できる

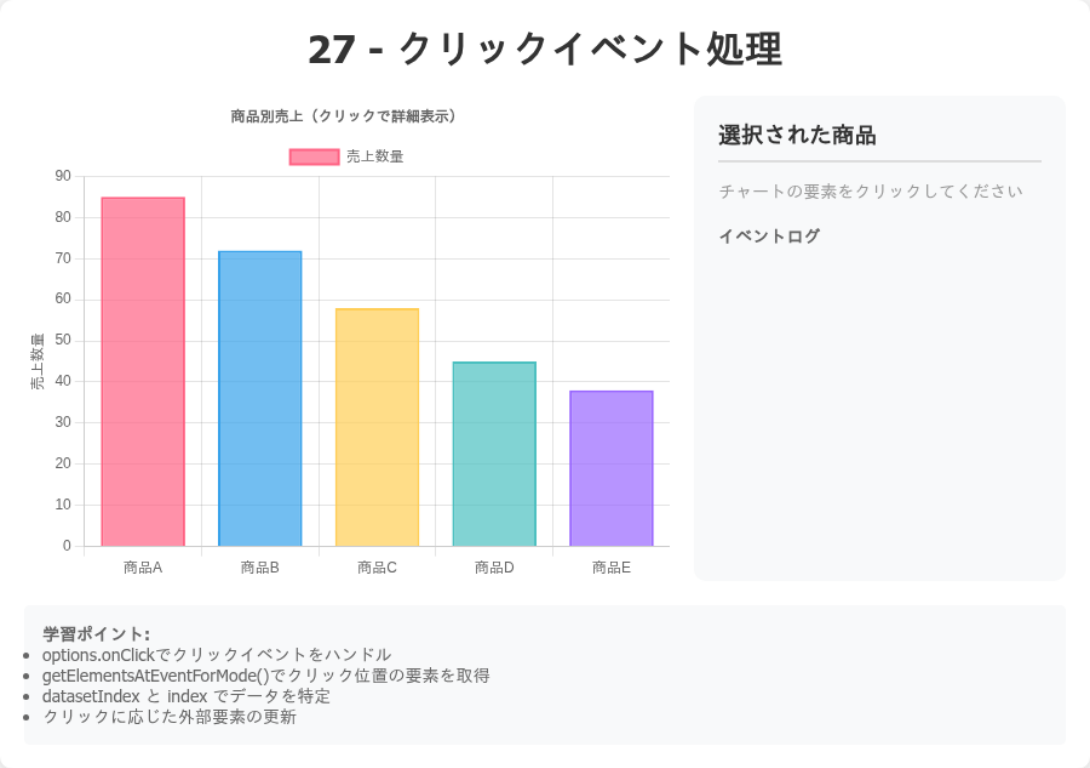

図 28.1　クリックイベント処理の実行結果

28.2 新しく学ぶ関数・プロパティ

28.2.1 onClick / onHover

Listing 28.1 イベント処理

```
options: {
    onClick: (event, elements, chart) => {
        if (elements.length > 0) {
            const element = elements[0];
            const datasetIndex = element.datasetIndex;
            const index = element.index;
            const value = chart.data.datasets[datasetIndex].data[index];
            console.log('クリックされた値:', value);
        }
    },
    onHover: (event, elements) => {
        event.native.target.style.cursor =
            elements.length > 0 ? 'pointer' : 'default';
    }
}
```

28.3 練習問題

問題 27-1

以下の要件を満たすクリックイベント処理を実装してください。

- 円グラフで部門別データを表示
- クリックしたセグメントの情報を別の HTML 要素に表示
- ホバー時にカーソルをポインターに変更
- 表示する情報：部門名、値、全体に対する割合

模範解答 27-1

```
const ctx = document.getElementById('myChart').getContext('2d');
const infoPanel = document.getElementById('info');
const data = [35, 25, 20, 20];
const labels = ['営業部', '開発部', '管理部', 'その他'];

new Chart(ctx, {
    type: 'pie',
    data: { labels: labels, datasets: [{ data: data }] },
```

```
    options: {
        onClick: (event, elements, chart) => {
            if (elements.length > 0) {
                const index = elements[0].index;
                const label = labels[index];
                const value = data[index];
                const total = data.reduce((a, b) => a + b, 0);
                const percent = ((value / total) * 100).toFixed(1);

                infoPanel.innerHTML = `
                    <h3>${label}</h3>
                    <p>値: ${value}</p>
                    <p>割合: ${percent}%</p>
                `;
            }
        },
        onHover: (event, elements) => {
            event.native.target.style.cursor =
                elements.length > 0 ? 'pointer' : 'default';
        }
    }
});
```

第 29 章

画像エクスポート

29.1　学習目標

この章を学習すると、以下のことができるようになります。

- `canvas.toDataURL()` でチャートを Base64 画像に変換できる
- PNG 形式と JPEG 形式でエクスポートできる
- `toBlob()` と Clipboard API でクリップボードにコピーできる
- ダウンロードリンクを動的に生成できる

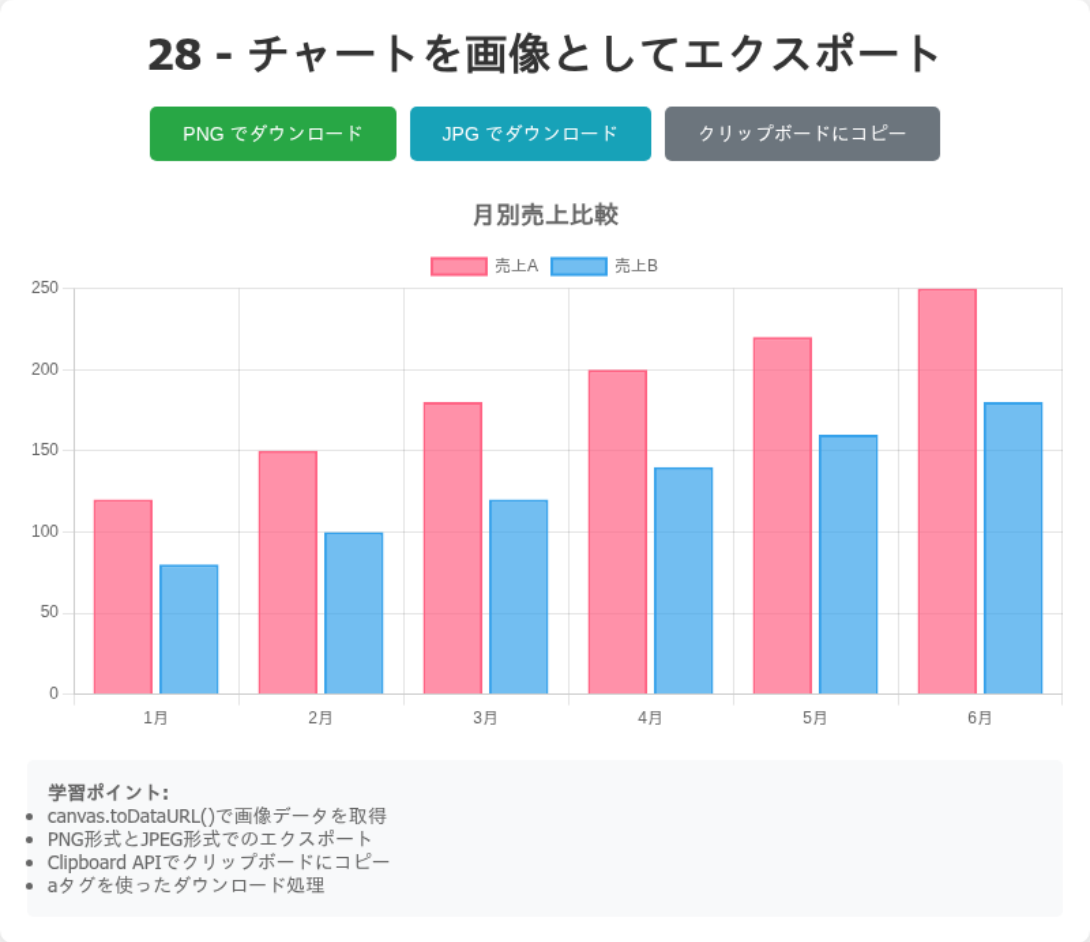

図 29.1　画像エクスポートの実行結果

29.2　新しく学ぶ関数・プロパティ

29.2.1　toDataURL

品質引数について

toDataURL の第 2 引数（品質）は **JPEG や WebP でのみ有効**です。PNG は可逆圧縮のため品質指定は無視されます。

Listing 29.1　画像エクスポート

```
function downloadPNG() {
    const canvas = document.getElementById('myChart');
    // PNGは可逆圧縮のため第2引数は無視される
    const dataUrl = canvas.toDataURL('image/png');

    const link = document.createElement('a');
    link.download = 'chart.png';
    link.href = dataUrl;
```

```
    link.click();
}

function downloadJPG() {
    const canvas = document.getElementById('myChart');
    // JPEGでは0.0〜1.0で品質を指定可能（0.9 = 90%品質）
    const dataUrl = canvas.toDataURL('image/jpeg', 0.9);
    // ...
}
```

29.2.2 toBlob / Clipboard API

Listing 29.2 クリップボードにコピー

```
async function copyToClipboard() {
    const canvas = document.getElementById('myChart');
    canvas.toBlob(async (blob) => {
        await navigator.clipboard.write([
            new ClipboardItem({ 'image/png': blob })
        ]);
    }, 'image/png');
}
```

29.3 練習問題

問題 28-1

以下の要件を満たす画像エクスポート機能を実装してください。

- チャートを PNG と JPEG の両形式でダウンロード可能に
- ダウンロードボタンを 2 つ用意
- ファイル名に現在の日時を含める
- JPEG 品質：90%

模範解答 28-1

```
const ctx = document.getElementById('myChart').getContext('2d');
new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['A', 'B', 'C', 'D'],
        datasets: [{ label: 'データ', data: [10, 20, 30, 40] }]
    }
```

```
});

function getFileName(ext) {
    const now = new Date();
    const timestamp = now.toISOString().slice(0,
        19).replace(/[:-]/g, '');
    return `chart_${timestamp}.${ext}`;
}

function downloadPNG() {
    const canvas = document.getElementById('myChart');
    const link = document.createElement('a');
    link.download = getFileName('png');
    // PNGは品質指定不要
    link.href = canvas.toDataURL('image/png');
    link.click();
}

function downloadJPEG() {
    const canvas = document.getElementById('myChart');
    const link = document.createElement('a');
    link.download = getFileName('jpg');
    link.href = canvas.toDataURL('image/jpeg', 0.9);
    link.click();
}
```

第 30 章

レスポンシブデザイン

30.1　学習目標

この章を学習すると、以下のことができるようになります。

- `responsive: true` でチャートを自動リサイズできる
- `maintainAspectRatio: false` で親要素にフィットさせられる
- CSS メディアクエリと組み合わせてレイアウトを調整できる
- 画面サイズに応じてチャートオプションを動的に変更できる

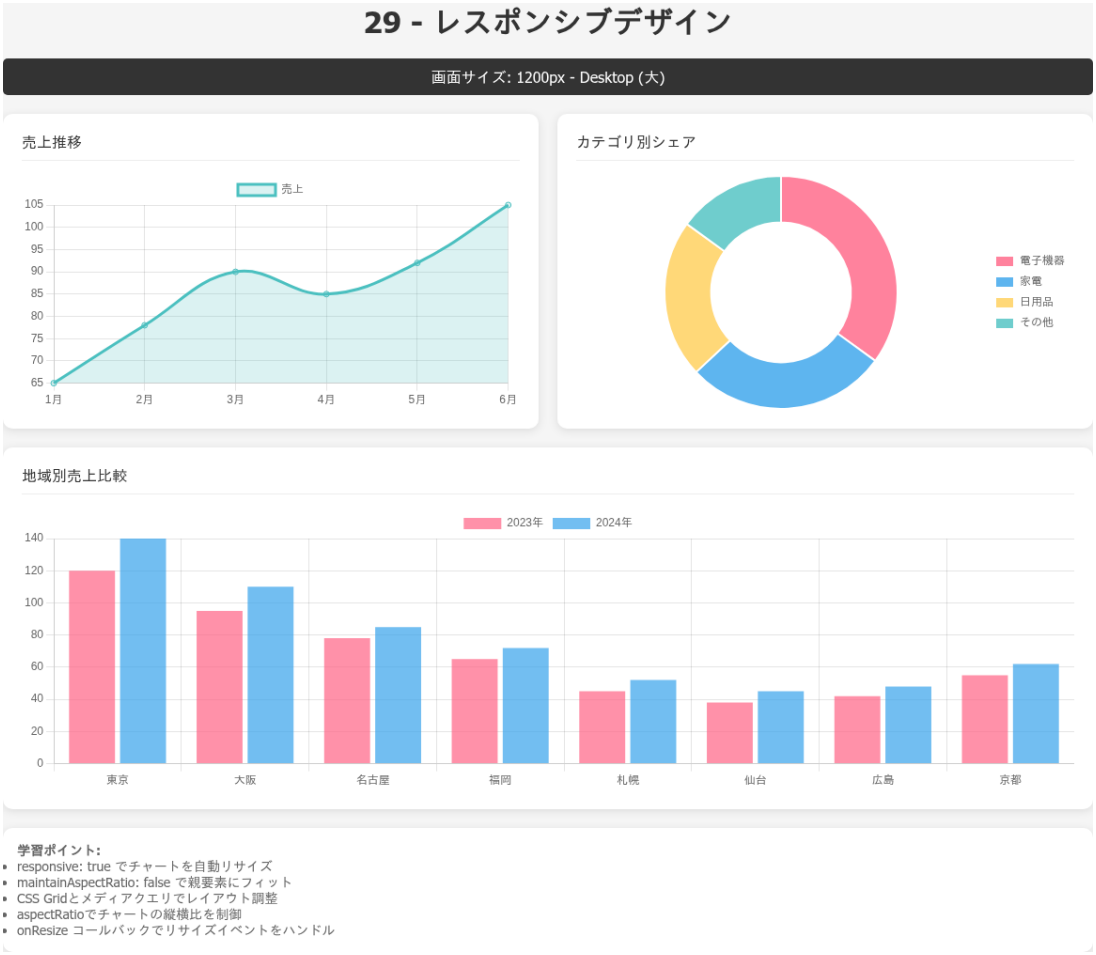

図 30.1　レスポンシブデザインの実行結果

30.2　新しく学ぶ関数・プロパティ

30.2.1　responsive / maintainAspectRatio

Listing 30.1　レスポンシブ設定

```
options: {
    responsive: true,           // ウィンドウサイズに追従
    maintainAspectRatio: false, // 親要素のサイズに合わせる
    onResize: (chart, size) => {
        console.log('リサイズ:', size.width, size.height);
    }
}
```

30.2.2　画面サイズに応じた設定

```
// 画面幅に応じて設定を変更
```

```
const isMobile = window.innerWidth < 576;
options: {
    indexAxis: isMobile ? 'y' : 'x',
    plugins: {
        legend: {
            position: isMobile ? 'bottom' : 'right'
        }
    }
}
```

30.3 練習問題

問題 29-1

以下の要件を満たすレスポンシブチャートを作成してください。

- モバイル（576px 未満）とデスクトップで異なる表示
- モバイル：横棒グラフ、凡例は下側
- デスクトップ：縦棒グラフ、凡例は右側
- ウィンドウリサイズ時に設定を動的に更新

模範解答 29-1

```
const ctx = document.getElementById('myChart').getContext('2d');
let chart = null;

function getChartOptions() {
    const isMobile = window.innerWidth < 576;
    return {
        responsive: true,
        maintainAspectRatio: false,
        indexAxis: isMobile ? 'y' : 'x',
        plugins: {
            legend: { position: isMobile ? 'bottom' : 'right' }
        }
    };
}

function createChart() {
    if (chart) chart.destroy();
    chart = new Chart(ctx, {
        type: 'bar',
        data: {
            labels: ['1月', '2月', '3月', '4月'],
            datasets: [{
```

```
                label: '売上',
                data: [120, 150, 180, 200],
                backgroundColor: 'rgba(54, 162, 235, 0.7)'
            }]
        },
        options: getChartOptions()
    });
}

createChart();
window.addEventListener('resize', () => {
    createChart();
});
```

第 31 章

ダッシュボード

31.1　学習目標

この章を学習すると、以下のことができるようになります。

- 複数のチャートを 1 つのページに配置できる
- `Chart.defaults` でグローバルスタイルを設定できる
- 共通オプションをスプレッド演算子で再利用できる
- KPI カードとチャートを組み合わせた実務的なダッシュボードを構築できる

図 31.1　ダッシュボードの実行結果

31.2　新しく学ぶ関数・プロパティ

31.2.1　Chart.defaults

グローバルなデフォルト設定を変更します。

Listing 31.1　グローバル設定

```
// すべてのチャートに適用されるデフォルト値
Chart.defaults.color = '#888';
Chart.defaults.borderColor = 'rgba(255, 255, 255, 0.1)';
Chart.defaults.font.family = 'Arial';
```

31.2.2 共通オプションの再利用

```
const commonOptions = {
    responsive: true,
    maintainAspectRatio: false,
    plugins: {
        legend: { labels: { color: '#888' } }
    }
};

// 各チャートで共通オプションを展開
new Chart(ctx1, { options: { ...commonOptions, /* 個別設定 */ } });
new Chart(ctx2, { options: { ...commonOptions, /* 個別設定 */ } });
```

31.2.3 条件付きスタイル

Listing 31.2 条件付き色設定

```
backgroundColor: (context) => {
    const value = context.dataset.data[context.dataIndex];
    return value > 700 ? '#f5576c' : '#667eea';
}
```

31.3 練習問題

問題 30-1

以下の要件を満たすダッシュボードを作成してください。

- 4 つの KPI カード（総売上、平均単価、総顧客数、成長率）
- 3 つのチャート：月別売上（棒）、カテゴリ別構成（ドーナツ）、トレンド（折れ線）
- `Chart.defaults` でグローバルスタイルを設定
- 共通オプションをスプレッド演算子で再利用
- レスポンシブなグリッドレイアウト

模範解答 30-1

```
// グローバル設定
Chart.defaults.color = '#666';
Chart.defaults.font.family = 'Arial, sans-serif';

```

```
// 共通オプション
const commonOptions = {
    responsive: true,
    maintainAspectRatio: false,
    plugins: {
        legend: { labels: { padding: 15 } }
    }
};

// 売上チャート（棒グラフ）
const salesCtx =
    document.getElementById('salesChart').getContext('2d');
new Chart(salesCtx, {
    type: 'bar',
    data: {
        labels: ['1月', '2月', '3月', '4月', '5月', '6月'],
        datasets: [{
            label: '売上',
            data: [120, 150, 180, 160, 200, 220],
            backgroundColor: 'rgba(54, 162, 235, 0.7)'
        }]
    },
    options: { ...commonOptions }
});

// カテゴリチャート（ドーナツ）
const categoryCtx =
    document.getElementById('categoryChart').getContext('2d');
new Chart(categoryCtx, {
    type: 'doughnut',
    data: {
        labels: ['製品A', '製品B', '製品C'],
        datasets: [{ data: [45, 30, 25] }]
    },
    options: { ...commonOptions, cutout: '60%' }
});

// トレンドチャート（折れ線）
const trendCtx =
    document.getElementById('trendChart').getContext('2d');
new Chart(trendCtx, {
    type: 'line',
    data: {
        labels: ['1月', '2月', '3月', '4月', '5月', '6月'],
        datasets: [{
            label: '成長率',
            data: [5, 8, 12, 10, 15, 18],
```

```
            borderColor: 'rgb(75, 192, 192)',
            tension: 0.3
        }]
    },
    options: { ...commonOptions }
});
```

第 IV 部

LLM 活用学習ガイド

第 32 章

最短で上達するプロンプトセット 30 本

32.1　本章の目的

本章では、ChatGPT、Claude 等の大規模言語モデル（LLM）を活用して、Chart.js を効率的に学習するためのプロンプトセットを紹介します。上から順に実行すると「描ける → 整う → 使える → 速い/壊れない」まで到達できる設計になっています。

LLM 学習のメリット

- 疑問点をその場で質問・解決できる
- 自分のペースで段階的に学習できる
- 実践的なコード例をすぐに得られる
- エラーの原因と解決策を詳しく説明してもらえる

LLM 利用時の注意

- **機密情報を入力しない**：社内データ、顧客情報、API キー等は貼り付けない
- **生成コードは必ず検証する**：LLM は誤ったコードを出力することがある
- **エラー時は情報を整理して質問**：エラーメッセージ、期待する動作、最小再現コードを提示すると解決しやすい
- **公式ドキュメントで照合する**：特に `options` や `plugins` の設定は公式で確認

32.2　この章の使い方

以下の手順でプロンプトを活用すると、効率的に学習できます。

1. **共通前置きを貼る**：次節の共通前置きプロンプトを LLM に入力
2. **該当プロンプトを貼る**：まずは Vanilla JS / HTML1 枚で動かす
3. **動いたら理解を固める**：「重要ポイント」と「よくあるミス」を読む
4. **要件追加は差分で依頼**：全コード再生成を避け、変更箇所だけ依頼
5. **詰まったら情報を整理**：最小再現コード・エラー全文・期待する動作を提示（機密は伏字）

理解を深めるコツ

LLM の回答をそのまま使うのではなく、まず自分の言葉で「何をしているか」を 3 行で要約してから実装しましょう。この一手間が LLM 依存を防ぎ、知識の定着につながります。

32.3 共通前置き（毎回コピペ推奨）

以下のプロンプトを各質問の前に付けることで、一貫性のある回答が得られます。

共通前置きプロンプト

```
あなたはChart.js（v4想定）の講師兼エンジニアです。
目的：初心者が「ブラウザで確実に動く」ことを最優先に、
段階的に理解できるように説明してください。

【前提】
- HTML1ファイル（CDN読み込み）で完結させる
- Chart.jsはcdn.jsdelivr.net/npm/chart.js@4.4.1を使用
- 追加依存（日付アダプタ等）がある場合は、
  理由と導入方法（CDN例）を必ず書く

【出力ルール】
0. 前提・仮定（3行以内。原則は仮定で進め、
   実行不能になる致命点のみ質問1〜2個）
1. 何を作るか（3行）
2. 完全なコード（HTML1ファイル、コピペで動く）
3. 重要ポイント（最大5つ）
4. 動作確認手順（3ステップ）
5. よくあるミス（最大3つ）
6. 次の練習（1つ）

【注意】
- 架空のAPIや存在しないオプションは使わない
- クォートは半角 ' " を使う
```

32.4 Phase 1：まず「確実に描ける」（難易度★ 1〜★ 2）

このフェーズでは、Chart.js の基本的な描画方法を習得します。

32.4.1 プロンプト 1：折れ線グラフの最小構成（★ 1）

到達目標

- Chart.js で「キャンバスに描画する流れ（canvas → ctx → new Chart）」を説明できる

- data.labels と data.datasets の意味が分かる

プロンプト

```
Chart.jsで折れ線グラフを「最小構成」で描くHTMLを
1ファイルで作ってください。
- Chart.jsはCDNで読み込み
- データ：labels = ["Mon","Tue","Wed","Thu","Fri"],
  values = [12,19,3,5,2]
- レスポンシブ
- コードに初心者向けコメントを付ける
```

32.4.2 プロンプト 2：棒グラフと単位付きツールチップ（★ 1）

到達目標

- 棒グラフの基本設定（軸、凡例、ツールチップ）を触れる
- ツールチップの値加工（単位付与）ができる

プロンプト

```
縦棒グラフのHTML1ファイル例を作ってください。
要件：
- Y軸は0始まり
- ツールチップに単位「円」を付ける（例：1200円）
- 値はカンマ区切り（例：1,200円）
- 初心者向けコメント付き
```

32.4.3 プロンプト 3：複数系列の折れ線グラフ（★ 2）

到達目標

- datasets を複数にして系列を増やせる
- 凡例の意味と、系列ごとの見た目設定を理解する

プロンプト

```
折れ線2系列（A/B）のグラフを作ってください。
- labels = ["1月","2月","3月","4月","5月"]
- A = [10,12,9,14,13]
- B = [8,11,7,10,12]
```

```
- 凡例は下
- hoverした系列だけ少し目立つように
  （初心者向けにやり方説明も）
```

32.4.4 プロンプト 4：データ整形入門（★ 2）

到達目標

- ”生データの形” から Chart.js 用データに変換できる
- 欠損（null）を扱える

プロンプト

```
次の生データ（配列のオブジェクト）を
Chart.jsのlabels/datasetsに変換し、折れ線で表示してください。
入力例：
[
  {"date":"2026-01-01","a":10,"b":12},
  {"date":"2026-01-02","a":null,"b":11},
  {"date":"2026-01-03","a":14,"b":null}
]
要件：
- 欠損はnullのまま
- 変換処理を関数にしてコメントで説明
- HTML1ファイルで動く形
```

32.4.5 プロンプト 5：データ更新（destroy しない）（★ 2）

到達目標

- チャートを”作り直さず” 更新できる
- `chart.data` と `chart.update()` の関係が分かる

プロンプト

```
ボタンを押すとデータが切り替わる例を作ってください（HTML1ファイル）。
要件：
- Chartインスタンスは維持（destroyしない）
- 2つのデータセット（例：週次と月次）を切り替える
- 初心者向けに「どこを変更すると動くか」を解説
```

32.4.6 プロンプト 6：デバッグ入門（真っ白問題）（★ 2）

到達目標

- ”真っ白” の典型原因をチェックリストで潰せる
- Console で見るポイントを理解する

プロンプト

```
Chart.jsで「真っ白になる」原因を、
優先度順にチェックリスト化してください。
各項目に：
- 症状
- 確認方法（どこを見るか）
- 典型の修正例（短いコード）
を付けてください。初学者向けに。
```

32.5 Phase 2：「業務資料っぽく整う」（難易度★ 2〜★ 3）

このフェーズでは、見た目を整えて業務で使えるレベルにします。

32.5.1 プロンプト 7：見た目の基本（★ 2）

到達目標

- options の役割（見た目・挙動）を理解する
- ”見やすさ調整” のよく使う場所が分かる

プロンプト

```
白背景の業務ダッシュボード向けに、見やすい折れ線グラフの設定を入れた
HTMLを作ってください。
要件：
- グリッドは薄め
- 軸ラベルは読みやすいサイズ
- 余白が詰まりすぎない
- どの設定が何に効くかをコメントで説明
```

32.5.2 プロンプト 8：数値フォーマット（★ 3）

到達目標

- Intl.NumberFormat('ja-JP') を使える
- 軸とツールチップの” 同じフォーマット関数” を再利用できる

プロンプト

```
数値フォーマット関数を作り、
軸の目盛りとツールチップの両方で使ってください。
モード：
- "currency": 1,234円
- "percent": 12.3%
- "plain": 1,234
要件：
- 関数は1つに集約
- 初心者向けに、どこで呼ばれているか説明
- HTML1ファイルで動く
```

32.5.3 プロンプト 9：長いラベル問題（★ 3）

到達目標

- ラベルの” 見切れ” を 3 通りで解決できる
- それぞれのメリット・デメリットを言語化できる

プロンプト

```
X軸ラベルが長いときの対処を3案で示してください。
案：
1) 45度回転
2) 省略（…）
3) 改行（スペース区切りなど）
各案について：
- 設定例（Chart.js）
- 使い分け基準
- 初心者向け注意点
```

32.5.4 プロンプト 10：棒グラフの見栄え（★ 2）

到達目標

- ” 棒の印象” を調整する主要パラメータを理解する
- 見た目を整えるときの触りどころが分かる

プロンプト

```
棒グラフを「角丸」「間隔調整」「太さ調整」して見やすくしてください。
要件：
- 角丸を付ける
- 棒が詰まりすぎない
- 何の設定が効いているか初心者向けに説明
- HTML1ファイルで動く例
```

32.5.5 プロンプト 11：複合グラフと 2 軸（★ 3）

到達目標

- 2 軸（left/right）の意味を理解する
- 棒と線を同じチャートに載せられる

プロンプト

```
棒＝売上（円）、線＝成長率（%）の複合グラフを
作ってください（HTML1ファイル）。
要件：
- 左軸：円（カンマ＋円）
- 右軸：%（0〜100）
- ツールチップで両方表示
- 初心者向けに「なぜ軸IDが必要か」を説明
```

32.5.6 プロンプト 12：注釈プラグイン（★ 3）

到達目標

- ”プラグインで描画を足せる”ことを理解する
- afterDraw 等の考え方を掴む

プロンプト

```
チャート右上に「更新日: 2026-01-05」のような
注釈を描画してください。
要件：
- Chart.jsのカスタムプラグインで実装
- 文字サイズは小さめ
- 初心者向けに、プラグインがいつ呼ばれるか説明
```

```
- HTML1ファイルで動く
```

32.6 Phase 3：操作性で差がつく（難易度★ 3）

このフェーズでは、インタラクティブな機能を追加します。

32.6.1 プロンプト 13：レポート形式ツールチップ（★ 3）

到達目標

- ツールチップ表示内容をカスタムできる
- 前の点との差分計算ができる

プロンプト

```
ツールチップに以下を複数行表示してください：
- 値
- 前の点との差分（前年差）
- 差分率（%）
要件：
- 差分は「前のデータ点」と比較
  （なければ表示しない）
- 数値フォーマットは関数にまとめる
- HTML1ファイルで動く例
```

32.6.2 プロンプト 14：HTML ツールチップ（★ 3）

到達目標

- ”canvas 外” に情報を出せる
- DOM 操作と Chart.js を繋げられる

プロンプト

```
external tooltip（HTMLツールチップ）の最小実装を作ってください。
要件：
- canvasの外にdivを作り、ホバー時に表示/更新
- 項目名と値を表形式で表示
- 位置がズレないように（簡易でOK）
- 初心者向けコメント付き
```

32.6.3 プロンプト 15：凡例クリック制御（★ 3）

到達目標

- 凡例クリックの挙動をカスタマイズできる
- ” 見たい系列だけ” に絞り込む UI を作れる

プロンプト

```
凡例クリックで「その系列だけ表示」、もう一度で「全系列表示」に戻る
挙動を実装してください。
要件：
- 折れ線2〜3系列の例で
- 初心者向けに、どのプロパティが
  可視/不可視を決めるか説明
- HTML1ファイルで動く
```

32.6.4 プロンプト 16：ドリルダウン（★ 3）

到達目標

- クリックイベントから” どの棒を押したか” 取れる
- データを入れ替えて画面遷移風の体験を作れる

プロンプト

```
棒グラフで、棒をクリックすると
「詳細データのグラフ」に切り替わる例を作ってください。
要件：
- 戻るボタンで元に戻る
- クリックしたカテゴリ名を画面に表示
- HTML1ファイル、初心者向けコメント付き
```

32.6.5 プロンプト 17：クロスヘア（★ 3）

到達目標

- hover 中の状態を使って描画を追加できる
- ” 読み取りやすさ” のための小技が作れる

プロンプト

```
hover位置に縦線を描画するプラグインを
作ってください（クロスヘア風）。
要件：
- ツールチップと一緒に動く
- 線の描画が重くならないように簡易でOK
- 初心者向けに「どこからhover位置を取るか」説明
```

32.6.6 プロンプト 18：範囲ハイライト（★ 3）

到達目標

- インデックス範囲を可視化できる
- ” 重要期間の強調” ができる

プロンプト

```
startIndex/endIndex（例：2〜5）で指定した区間だけ、背景を薄く
ハイライトしてください。
要件：
- 折れ線で例示
- カスタムプラグインで実装
- 初心者向けに、座標変換（index→pixel）の
  考え方を説明
```

32.7 Phase 4：時系列をちゃんと扱う（難易度★ 3〜★ 4）

このフェーズでは、時系列データの扱い方を習得します。

32.7.1 プロンプト 19：time スケール入門（★ 4）

到達目標

- time スケールの概念が分かる
- ” 日付表示にはアダプタが必要” を理解する

プロンプト

```
X軸をtimeスケールにして、
ISO文字列（YYYY-MM-DD）で日付を渡す折れ線を作ってください。
```

```
要件：
- 必要な「日付アダプタ」を初心者向けに説明
  （CDNの例も）
- データ例も含める
- 動作しない時のチェックポイントも書く
```

32.7.2 プロンプト 20：欠損の扱い（★ 3）

到達目標

- null がある時の描画差を理解する
- spanGaps など欠損関連オプションを使える

プロンプト

```
欠損（null）がある時系列で、
次の2パターンを同じデータで比較してください：
A）欠損で線を切る
B）欠損をまたいで線をつなぐ
要件：
- 設定差分が分かるように
- それぞれの注意点（誤解リスク）を説明
```

32.7.3 プロンプト 21：移動平均（★ 4）

到達目標

- 前処理（計算）と描画（Chart.js）の分担を理解する
- 複数系列で” 生値＋指標” を出せる

プロンプト

```
時系列データから7日移動平均と30日移動平均を計算し、
元系列と合わせて3本の折れ線で表示してください。
要件：
- 移動平均の計算関数を別にする
- 初心者向けに「端の扱い（最初の数日）」を説明
- HTML1ファイルで動く例
```

32.7.4 プロンプト 22：表示粒度の切替（★ 4）

到達目標

- ” 見せ方は期間で変える” を実装できる
- tick のフォーマット設計ができる

プロンプト

```
表示期間が短いときは「日」、長いときは「月」で
目盛り表示を切り替える設計と実装例を出してください。
要件：
- 期間判定のルールを明文化
- tickのformatter例を提示
- 初心者向けに、なぜ必要か
  （読めなくなる問題）を説明
```

32.7.5 プロンプト 23：目標ラインと塗り分け（★ 4）

到達目標

- ” 目標” という参照線を追加できる
- 条件付きで背景を塗れる（プラグイン）

プロンプト

```
目標値targetを水平線で表示し、
未達の区間は薄赤、達成区間は薄青で背景を塗り分けてください。
要件：
- 折れ線で例示
- カスタムプラグインで実装
- 初心者向けに「どのデータを基準に塗り分けるか」
  を説明
```

32.8 Phase 5：大量データと高速化（難易度★ 4）

このフェーズでは、パフォーマンス最適化を学びます。

32.8.1 プロンプト 24：10 万点の最適化（★ 4）

到達目標

- ” 重くなる原因” を構造的に説明できる
- Chart.js 側と前処理側の対策を分けて考えられる

プロンプト

```
折れ線で10万点を想定したときの最適化方針を、
初心者にも分かるようにまとめてください。
含める内容：
- Chart.js側の設定
  （例：間引き/アニメーションなど）
- 前処理側の工夫
- やってはいけない例
- 最小のサンプルコード（重くしない設定）
```

32.8.2 プロンプト 25：リアルタイム更新（★ 4）

到達目標

- 末尾追加と古い点の破棄を実装できる
- ” 更新頻度が高い時の設計” が分かる

プロンプト

```
1秒ごとにデータ点を1つ追加し、
古い点を捨てて一定点数を保つ例を作ってください（HTML1ファイル）。
要件：
- リングバッファ方式（配列が無限に増えない）
- 更新は軽く（アニメ抑制など）
- 初心者向けに「なぜこの設計が必要か」説明
```

32.8.3 プロンプト 26：重い原因の診断（★ 4）

到達目標

- 自分の config を” レビュー依頼” できる
- 改善案を段階的に適用できる

プロンプト

```
次のChart.js config / データで描画が重いです。
原因候補を「確度が高い順」に並べ、改善案を
```

```
- 軽（すぐできる）
- 中（少し手間）
- 重（設計変更）
の3段階で提案してください。
修正版configも提示してください。
---ここにconfigとデータを貼る---
```

32.9 Phase 6：実務導入（難易度★ 4〜★ 5）

このフェーズでは、React/Next.js/TypeScript 環境での実装を学びます。

前提知識

Phase 6 のプロンプトは、Vanilla JS で Chart.js を理解した後に取り組むことをお勧めします。React/Next.js/TypeScript の基礎知識も必要です。

32.9.1 プロンプト 27：React 入門（★ 4）

到達目標

- React で Chart インスタンスのライフサイクルを守れる
- 再レンダリングでバグらない構成が作れる

プロンプト

```
React（関数コンポーネント）でChart.jsを
安全に使う最小例を作ってください。
要件：
- useRefでcanvasとChartインスタンスを保持
- useEffectで初期化し、アンマウントでdestroy
- データ更新のやり方も1パターン示す
- 初心者向けに「なぜdestroyが必要か」を説明
```

32.9.2 プロンプト 28：Next.js SSR 対策（★ 5）

到達目標

- SSR で落ちる理由（window/document）を説明できる
- 安全な読み込み（dynamic import 等）の方針が分かる

プロンプト

```
Next.jsでChart.jsを使うときの
SSR対策込みの実装例を出してください。
要件：
- SSRで起きがちなエラー例と原因
- 回避方法（クライアントのみで描画）
- 最小のコンポーネント例
- 初心者向けに手順を説明
```

32.9.3 プロンプト 29：TypeScript 型安全（★ 4）

到達目標

- ChartConfiguration<'line'> 等で型に守られた実装ができる
- よくある型エラーを読んで直せる

プロンプト

```
TypeScriptで型安全にChart.jsのconfigを書く例を出してください。
要件：
- lineチャートのChartConfigurationで書く
  （satisfies演算子を使う方法も併記）
- よくある型エラー例を2つ挙げ、どう直すか解説
- 初心者向けに「型安全のメリット」を短く説明
```

32.10 Phase 7：仕上げ（難易度★ 5）

最終フェーズでは、再利用可能な設計を学びます。

32.10.1 プロンプト 30：再利用設計（★ 5）

到達目標

- ” 設定生成関数” で再利用できる構造にできる
- テーマ共通化・テストしやすい設計が分かる

プロンプト

```
Chart.js実装を、再利用しやすい構造にリファクタしてください。
要件：
```

```
- buildConfig(data, theme, options) の純粋関数中心
- theme（色/フォント/グリッド）を共通化
- 1) フォルダ構成案　2) 実装例
  3) テストしやすくする工夫　を出す
前提はVanillaでもReactでもOK（両方あると嬉しい）。
初心者向けに説明も付ける。
```

32.11 この 30 本を終えると到達できるレベル

- Chart.js の **基本構造（data/options/plugins）**を説明できる
- よくある要件（複数系列、2 軸、単位、欠損、時系列、注釈、クリック）を実装できる
- ” 真っ白/反映されない/重い” を **自力で切り分ける手順** を持てる
- React/Next/TS で **壊れにくい実装** の型が分かる

継続学習のヒント

プロンプトセットを一通り終えたら、実際のプロジェクトで使ってみましょう。分からないことがあれば LLM に質問し、エラーメッセージをそのまま貼り付けて解決策を聞くのも効果的です。

次の一歩

30 本のプロンプトを終えたら、ぜひ自分の研究データや授業で使う実データを使って、オリジナルのダッシュボードを 1 つ作ってみてください。「サンプルデータで動かす」から「自分のデータで価値を出す」へ踏み出すことで、本当の意味で Chart.js が身につきます。

自分のデータに置き換えるチェックリスト

データ形式を確認（CSV/JSON/API）
`labels` と `datasets` に変換するロジックを書く
欠損値（null）の扱いを決める
軸の範囲・単位を実データに合わせる
機密情報が含まれていないか確認

参考文献・参考サイト

公式ドキュメント

- Chart.js 公式サイト
  https://www.chartjs.org/
- Chart.js 公式ドキュメント
  https://www.chartjs.org/docs/latest/
- Chart.js GitHub リポジトリ
  https://github.com/chartjs/chart.js
- chartjs-plugin-datalabels 公式ドキュメント
  https://chartjs-plugin-datalabels.netlify.app/

Web 技術関連

- MDN Web Docs - Canvas API
  https://developer.mozilla.org/ja/docs/Web/API/Canvas_API
- MDN Web Docs - JavaScript ガイド
  https://developer.mozilla.org/ja/docs/Web/JavaScript/Guide
- HTML Living Standard（日本語訳）
  https://momdo.github.io/html/

参考書籍

- 山田祥寛『JavaScript 本格入門 改訂 3 版』技術評論社, 2023 年
- Murray, S. *Interactive Data Visualization for the Web: An Introduction to Designing with D3*, 2nd Edition, O'Reilly Media, 2017 年
- Meeks, E. *D3.js in Action: Data Visualization with JavaScript*, 2nd Edition, Manning Publications, 2017 年

データ可視化関連

- Tufte, E. R. *The Visual Display of Quantitative Information*, 2nd Edition, Graphics Press, 2001 年
- Few, S. *Show Me the Numbers: Designing Tables and Graphs to Enlighten*, 2nd Edition, Analytics Press, 2012 年
- Cairo, A. *The Functional Art: An Introduction to Information Graphics and Visualization*, New Riders, 2012 年

オンライン学習リソース

- Chart.js サンプルギャラリー
  https://www.chartjs.org/docs/latest/samples/
- CodePen - Chart.js タグ
  https://codepen.io/tag/chartjs
- Stack Overflow - Chart.js タグ
  https://stackoverflow.com/questions/tagged/chart.js

学習のヒント

Chart.js は活発に開発が続けられているライブラリです。最新の機能や API の変更については、公式ドキュメントを定期的に確認することをお勧めします。また、GitHub のリリースノートで各バージョンの変更点を確認できます。

索引

addColorStop, 78
afterBody, 82
afterDatasetsDraw, 108
afterDraw, 108
animation, 112

backgroundColor, 16, 20
beforeBody, 82
beforeDatasetsDraw, 108
beforeDraw, 108
borderColor, 16, 20
borderRadius, 20
borderWidth, 20

Canvas, 10
CDN, 10
Chart
    コンストラクタ, 16
Chart.defaults, 136
Chart.js, 9
chartjs-plugin-datalabels, 94
clearInterval, 101
Clipboard API, 129
createLinearGradient, 78
createLinearGradient(), 78
cutout, 32

data, 16
datalabels, 94
datasets, 16, 40
destroy, 105

easing, 112

fill, 52
footer, 82

generateLabels, 86
getContext('2d'), 11
grid, 90

hoverOffset, 28

indexAxis, 24

label, 16, 82
labels, 16
legend.display, 86

maintainAspectRatio, 16, 132

onClick, 104
onHover, 124
options, 16

plugins.title, 16
pointHoverRadius, 58
pointRadius, 58
push, 100

responsive, 16, 132

scales, 116
scales.r, 36
setInterval, 101
shift, 100
stacked, 48
stepped, 74

tension, 16
ticks.callback, 90
title, 82
toBlob, 129
toDataURL, 128
toggleDataVisibility, 87
tooltip.callbacks, 82
tooltip.external, 120
type
    bar, 20
    bubble, 62
    doughnut, 32
    line, 16
    pie, 28
    polarArea, 66
    radar, 36
    scatter, 58
    データセット別, 70

update, 87, 101

yAxisID, 70

データ形式
    {x, y, r}, 62
    {x, y}, 58
プラグイン, 108

条件付きスタイル, 137