

HTML



CSS

身につく実践入門

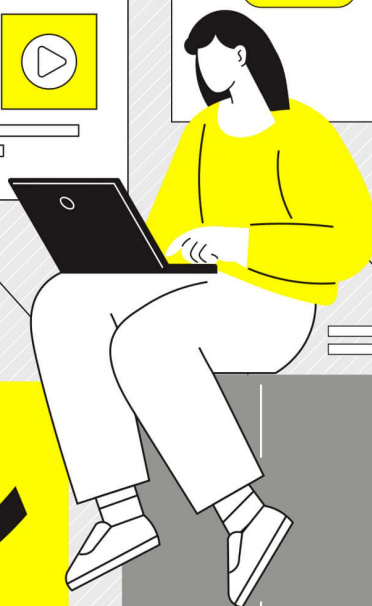
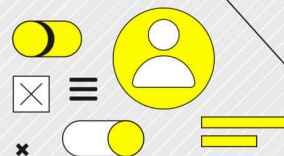
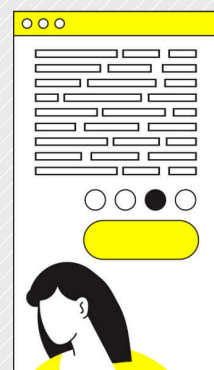
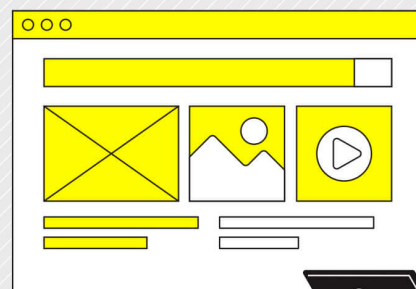
スキルが

コーディング・

現場レベルの

小豆沢 健 Ken Azukizawa

Web



デザイン

入門編から実践編まで
6ステップで上達できる！

Codejump
制作
練習ファイル
ダウンロード

技術評論社

実案件をもとに作成した練習サイトだから、プロの知識が学べます。

フレックスボックス / ポジション / CSS アニメーション / JavaScript (jQuery) / ドロップダウンメニュー / ハンバーガーメニュー /
アコーディオン / モーダルウィンドウ / シングルカラム / 2カラム / タイル型 / ブロッキンググリッド / LP (ランディングページ)

はじめに

「Webサイトが作れない」

これは、HTMLとCSSを学習した後、はじめて私がWebサイトを作ろうとした時の感想です。もともと新卒でシステム開発会社に入社した私は、ITやWebの現場で10年以上、システム開発の仕事に携わってきました。それまで多くのプログラミング経験を積んできた私にとって、HTMLやCSSの学習は特別難しいものではありませんでした。むしろ、比較的かんたんに理解できたように記憶しています。それなのに、いざWebサイトを作ろうとするとまったく手が動かないのです。

私は不思議でした。文章はpタグで記述し、コンテンツはdivタグでグルーピングすることは知っているし、記述したHTMLに対してCSSのプロパティを使うことで、装飾をつけたりレイアウトの配置ができることもしっかりと理解していました。それなのにWebサイトが作れない…。

当時は今ほどWebサイト制作を学ぶための情報が充実していなかったもので、困った私は、実際に公開されているサイトと同じものをマネして作ってみるといふ、最近では「模写コーディング」と呼ばれる練習方法でWebサイトの作り方を学ぶことにしました。とはいえ、公開されているサイトはいろいろなJavaScriptやCSSファイルが大量に読み込まれていたり、コードの書き方がサイトによってバラバラだったり、勉強するのに非常に苦労しました。それでもなんとか諦めずに練習の数をこなすことで、かなり遠回りをしましたが、Webサイトのコーディングスキルを身につけることができました。

その後、仕事を通してさまざまなWebサイト制作の経験を積んだ私は、HTMLとCSSを理解していたはずの私がなぜWebサイトを作れなかったのかが少しずつわかるようになってきました。そして、その気づきやノウハウをもとにコーディングを学習したい人が自由に学べる「Codejump」という学習サイトを立ち上げました。さらに、コーディングを仕事にしていきたい人が実際の案件をモデルにした練習サイトで学べる「Codejump Pro」というサービスも開始しました。Codejumpを立ち上げてから約5年がたち、今では累計10万人以上の人にこのサイトをご利用いただき、その効果を実感しています。

そんな私の10年以上にわたるWebサイト制作の経験と5年間のCodejumpの運営経験をもとに、仕事に必要な実践的なコーディングスキルを1冊の本で学べるようにまとめたのが、本書「HTML&CSS Webデザイン 現場レベルのコーディング・スキルが身につく実践入門」です。本書は、

いわゆるHTMLの基礎やかんたんなサンプルサイトの作り方を学ぶ本ではなく、HTMLの基礎学習を終えた方が、仕事に必要なスキルを習得することを目的にした学習本となっています。そして、私の過去の経験とノウハウをもとに、コーディングを学習する際に重要な以下の3つの要素を取り入れることで、挫折することなく最後まで着実にスキルを積み上げられるようにしました。

要素1 レイアウト構成を学ぶ

コーディングを学習するとすぐにコードを書きたくなりますが、レイアウト構成がイメージできていないのにコードを書こうとすると、手が止まる原因になってしまいます。当時、私がWebサイトを作れなかったのも、これが原因でした。本書では、最初にレイアウト構成を確認してからコードの解説を行うことで、デザインを形にするレイアウト構成力を養えるようになっています。

要素2 ステップアップ方式で学ぶ

サンプルサイトで学習を行う際は、自分のスキルにあったものから始めることが重要です。難しいものから始めてしまうと挫折の原因になりかねないので、なるべくかんたんなものから始め、少しずつ難易度を上げていくのがおすすめです。本書では、入門編から実践編と段階を踏みながら合計6個のサンプルサイトを制作することで、途中で挫折することなく着実にスキルを上げていくことができます。

要素3 実務で使うスキルを学ぶ

HTML、CSS、JavaScriptの技術は範囲が広く、奥も深いのですが、実は仕事で使う技術はある程度パターンが限られています。本書では、私がこれまでWebサイト制作の仕事の中で実際によく使ってきた定番の技術をピックアップし、1章から6章のサンプルサイトにもれなく盛り込みました。この6個のサンプルサイトを学習するだけで、実務に必要なスキルが一通り学べるようになっています。

本書は、タイトルの通り「現場レベルのコーディング・スキルが身につく」Webサイト制作の実践本です。本書を使って学習することで、実際の仕事でよく使う技術を一通り学べるように、私がこれまで培ってきた知識や経験をすべて詰め込みました。これからコーディングを仕事にしていこうと考えている方は、ぜひ本書を活用して実務レベルのスキルを身につけてください。

いつの日か、みなさまと仕事の現場でお会いできる日を楽しみにしています。

それでは、学習の扉の1ページ目を開いてみてください！

2025年1月 小豆沢 健

この本の使い方

本書は、サンプルサイトの制作を通してHTML、CSSのコーディングを学ぶ、Webサイト制作のコーディング学習本です。ご自身のスキルや学習状況にあわせて、以下の3通りの方法で学習することができます。

方法① 解説を読んで学ぶ

方法② 解説と一緒に作りながら学ぶ

方法③ デザインデータと仕様書から作って学ぶ

Webサイトを作るのがはじめてという方は、①→②→③の順に3回学習していただくことで、より理解を深めることができます。将来、コーディングを仕事にしていきたいことを目指している方は、③ができるようになるまで繰り返し学習してみてください。

本書の構成

本書は全6章で構成されており、6種類のサンプルサイトを作ることで実務に必要なスキルが身につくようになっています。また、各章の解説は実際のコーディングと同じ流れで構成されているため、解説の順序に沿って読み進めていくことで、コーディングの手順を学べるようになっています。各章の構成は、下記の通りです。

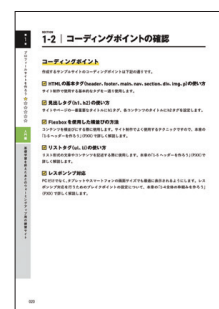
① 完成イメージを確認する

各章のはじめに完成イメージを掲載しています。まずは完成イメージを確認して、どんなサイトを作るのかを把握しましょう。



② コーディングポイントを理解する

この章で学べる、主なコーディングテクニックをまとめています。ここで紹介するポイントを意識しながら、学習に取り組んでみてください。



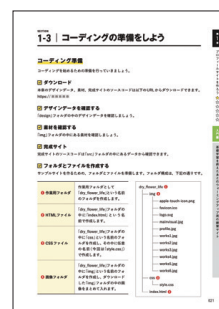
③ 仕様を確認する(第5章と第6章のみ)

第5章と第6章はWebサイトに動きが入るため、仕様をまとめたページがあります。デザインとあわせて、動きについても確認しておきましょう。



④ コーディングの準備を行う

必要なファイルをダウンロードし、コーディングに必要な準備を行います。書籍を読みながら一緒にコーディングしていく方法②の場合は、こちらの準備を行いましょう。



⑤ レイアウト構成の確認&コーディング

サンプルサイトの解説は、「レイアウト構成の確認」と「ソースコードの解説」がセットになっています。まずはレイアウト構成の確認を行った後、コードの解説を行っていきます。また、コードの中で特に重要な箇所は「OnePoint」として詳しい解説を行っています。



デザインデータと仕様書からコーディングを行う場合

方法③の「デザインデータと仕様書から作って学ぶ」場合は、各章のダウンロードデータ内にある「design」フォルダのデザインデータと、第5章、第6章については書籍内の「コーディング仕様の確認」ページをもとにコーディングを行っていきます。デザインデータはAdobe XDとFigmaの2種類のデータが入っているので、好きな方をご利用ください。完成後は、ダウンロードデータ内の「src」フォルダに入っているサンプルソースを参考に、表示や動作、コードの書き方などを確認してみてください。なお、ソースコードの書き方にはいろいろな方法があります。サンプルソースは、書き方の1つとして参考にしてください。

この本で学べること

本書を使って学習することで、実務に必要なコーディング・スキルを入門から実践へと段階を踏みながら身につけることができます。各章で学べる内容は、下記の通りです。

☑ 第1章 入門編 (フレックスボックス)

プロフィールサイト：1ページ

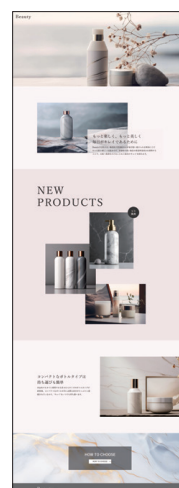
第1章で学ぶメインのテーマは「フレックスボックス」です。フレックスボックスはコンテンツを横並びにする技術で、Webサイト制作のレイアウト配置でもっともよく使う技術の1つです。本章では、かんたんなプロフィールサイトを作りながら、フレックスボックスの使い方について学びます。



☑ 第2章 初級編 (ポジション)

ブランドサイト：1ページ

第2章で学ぶメインのテーマは「ポジション」です。ポジションはコンテンツを任意の場所に配置する技術で、1章のフレックスボックスとあわせて実務で頻繁に使用する技術です。本章では、シンプルな1ページのブランドサイトの制作を通して、ポジションの使い方について学びます。



☑ 第3章 中級編 (複数ページ)

サービスサイト：4ページ

第3章で学ぶメインのテーマは「複数ページのサイト制作」です。複数ページのWebサイトを作る際のファイル構成やページ間でのリンクの貼り方などについて、サービスサイトの制作を通して学びます。



☑ 第4章 上級編 (複数レイアウト)

カフェサイト : 4ページ

第4章で学ぶメインのテーマは「様々な種類のレイアウト」です。Webサイトでよく見かける、シングルカラム、2カラム、タイル型、ブローキングリッドレイアウトの作り方について、カフェサイトの制作を通して学びます。



☑ 第5章 応用編 (動きのあるWebサイト)

ランディングページ (LP) : 1ページ

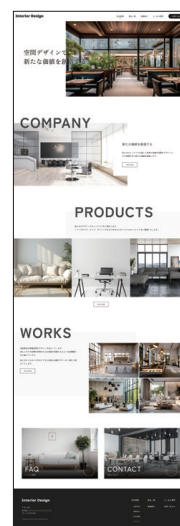
第5章で学ぶメインのテーマは「動きのあるWebサイト」です。ランディングページの制作を通して、CSSアニメーションやJavaScript (jQuery) を使った、動きのあるWebサイトの作り方について学びます。



☑ 第6章 実践編 (実務を想定したサイト制作)

コーポレートサイト : 6ページ

第6章では、第1章から第5章までの内容をすべて含んだコーポレートサイトの制作を通して、実務を想定したWebサイト制作について学びます。また、アコーディオンやモーダルウィンドウといった、実務で定番のテクニックについても学びます。



CONTENTS

はじめに	2
この本の使い方	4
この本で学べること	6
ダウンロードファイルの使い方	8
123RF について	16

第 1 章 | 入門編 プロフィールサイトを作ろう

SECTION 1-1	完成イメージの確認	18
SECTION 1-2	コーディングポイントの確認	20
SECTION 1-3	コーディングの準備をしよう	21
SECTION 1-4	全体の枠組みを作ろう	22
SECTION 1-5	ヘッダーを作ろう	29
SECTION 1-6	メインビジュアルとイントロダクションを作ろう	38
SECTION 1-7	「Profile」を作ろう	42
SECTION 1-8	「Works」を作ろう	47
SECTION 1-9	フッターを作ろう	52

第 2 章

初級編

ブランドサイトを作ろう

SECTION 2-1	完成イメージの確認	56
SECTION 2-2	コーディングポイントの確認	59
SECTION 2-3	コーディングの準備をしよう	60
SECTION 2-4	全体の枠組みを作ろう	61
SECTION 2-5	ヘッダーを作ろう	66
SECTION 2-6	メインビジュアルを作ろう	71
SECTION 2-7	「Concept」を作ろう	75
SECTION 2-8	「New Products」を作ろう	82
SECTION 2-9	「New Type」を作ろう	90
SECTION 2-10	「Online Store」を作ろう	95
SECTION 2-11	フッターを作ろう	103

第 3 章

中級編

サービスサイトを作ろう

SECTION 3-1	完成イメージの確認	106
SECTION 3-2	コーディングポイントの確認	114
SECTION 3-3	コーディングの準備をしよう	115
SECTION 3-4	トップページの枠組みを作ろう	117
SECTION 3-5	ヘッダーを作ろう	123
SECTION 3-6	メインビジュアルを作ろう	133
SECTION 3-7	「選ばれる理由」を作ろう	139
SECTION 3-8	「サービス」と「料金」を作ろう	145
SECTION 3-9	「ご利用者の声」を作ろう	152
SECTION 3-10	「お問い合わせ」を作ろう	158

SECTION 3-11	フッターを作ろう	161
SECTION 3-12	サービスページの枠組みを作ろう	165
SECTION 3-13	共通パーツ（ヘッダー、フッター、お問い合わせ）を作ろう	168
SECTION 3-14	ページヘッダーを作ろう	170
SECTION 3-15	ページヘッダー下テキストを作ろう	173
SECTION 3-16	「サービスの内容」を作ろう	176
SECTION 3-17	「ご利用の流れ」を作ろう	180
SECTION 3-18	料金ページの枠組みを作ろう	185
SECTION 3-19	共通パーツ（ヘッダー、フッター、お問い合わせ）を作ろう	187
SECTION 3-20	ページヘッダーとページヘッダー下テキストを作ろう	188
SECTION 3-21	「料金表」を作ろう	189
SECTION 3-22	お問い合わせページの枠組みを作ろう	197
SECTION 3-23	共通パーツ（ヘッダー、フッター）を作ろう	199
SECTION 3-24	ページヘッダーとページヘッダー下テキストを作ろう	200
SECTION 3-25	フォームを作ろう	201

第 4 章

上級編 カフェサイトを作ろう

SECTION 4-1	完成イメージの確認	208
SECTION 4-2	コーディングポイントの確認	218
SECTION 4-3	コーディングの準備をしよう	219
SECTION 4-4	トップページの枠組みを作ろう	221
SECTION 4-5	ヘッダーエリアを作ろう	226
SECTION 4-6	「コンセプト」を作ろう	236
SECTION 4-7	「メニュー」を作ろう	243
SECTION 4-8	「ブログ」を作ろう	251
SECTION 4-9	「アクセス」を作ろう	257
SECTION 4-10	フッターを作ろう	262

SECTION 4-11	コンセプトページの枠組みを作ろう	265
SECTION 4-12	ヘッダーエリア、フッター、アクセスを作ろう	268
SECTION 4-13	「コンセプト」を作ろう	272
SECTION 4-14	ブロッグ一覧ページの枠組みを作ろう	281
SECTION 4-15	ヘッダーエリア、フッター、アクセスを作ろう	284
SECTION 4-16	ブロッグリストを作ろう	286
SECTION 4-17	ブロッグ詳細ページの枠組みを作ろう	291
SECTION 4-18	ヘッダーエリア、フッター、アクセスを作ろう	294
SECTION 4-19	ブロッグ記事とサイドバーを作ろう	296

第 5 章 | 応用編 ランディングページを作ろう

SECTION 5-1	完成イメージの確認	310
SECTION 5-2	コーディング仕様の確認	314
SECTION 5-3	コーディングポイントの確認	316
SECTION 5-4	コーディングの準備をしよう	317
SECTION 5-5	全体の枠組みを作ろう	319
SECTION 5-6	ヘッダーを作ろう	326
SECTION 5-7	メインビジュアルを作ろう	336
SECTION 5-8	「About」を作ろう	340
SECTION 5-9	パララックスを作ろう	348
SECTION 5-10	「ツアー紹介」を作ろう	352
SECTION 5-11	「reservation」を作ろう	364
SECTION 5-12	「アクティビティ」を作ろう	370
SECTION 5-13	「アクティビティ MAP」を作ろう	376
SECTION 5-14	フッター上の背景を作ろう	387
SECTION 5-15	追従ボタンを作ろう	389
SECTION 5-16	フッターを作ろう	392

第 6 章

実践編

コーポレートサイトを作ろう

SECTION 6-1	完成イメージの確認	400
SECTION 6-2	コーディング仕様の確認	414
SECTION 6-3	コーディングポイントの確認	416
SECTION 6-4	コーディングの準備をしよう	417
SECTION 6-5	トップページを作ろう	420
SECTION 6-6	ヘッダーを作ろう	426
SECTION 6-7	メインビジュアルを作ろう	443
SECTION 6-8	「COMPANY」を作ろう	448
SECTION 6-9	「PRODUCTS」を作ろう	455
SECTION 6-10	「WORKS」を作ろう	462
SECTION 6-11	「FAQ&CONTACT」を作ろう	466
SECTION 6-12	フッターを作ろう	470
SECTION 6-13	フェードインを作ろう	475
SECTION 6-14	会社情報ページを作ろう	479
SECTION 6-15	共通パーツ（ヘッダー、フッター）を作ろう	483
SECTION 6-16	ページヘッダーを作ろう	485
SECTION 6-17	ページ内リンクを作ろう	489
SECTION 6-18	「企業理念」を作ろう	494
SECTION 6-19	「事業紹介」を作ろう	500
SECTION 6-20	「会社概要」を作ろう	505
SECTION 6-21	「アクセス」を作ろう	509
SECTION 6-22	商品一覧ページを作ろう	512
SECTION 6-23	共通パーツ（ヘッダー、フッター）を作ろう	515
SECTION 6-24	ページヘッダーを作ろう	516
SECTION 6-25	タブと画像一覧を作ろう	517
SECTION 6-26	実績紹介ページを作ろう	527

SECTION 6-27 共通パーツ（ヘッダー、フッター）を作ろう	530
SECTION 6-28 ページヘッダーを作ろう	531
SECTION 6-29 実績紹介一覧を作ろう	532
SECTION 6-30 よくある質問ページを作ろう	543
SECTION 6-31 共通パーツ（ヘッダー、フッター）を作ろう	546
SECTION 6-32 ページヘッダーを作ろう	547
SECTION 6-33 Q&A 一覧を作ろう	548
SECTION 6-34 お問い合わせページを作ろう	555
SECTION 6-35 共通パーツ（ヘッダー、フッター）を作ろう	558
SECTION 6-36 ページヘッダーを作ろう	559
SECTION 6-37 フォームを作ろう	560
索引	570
おわりに	574

免責

本書に記載された内容は、情報の提供のみを目的としています。したがって、本書を用いた運用は、必ずお客様自身の責任と判断によって行ってください。これらの情報の運用の結果、いかなる障害が発生しても、技術評論社および著者はいかなる責任も負いません。

また、本書掲載のWebページの内容は、企業、店舗、個人等、すべて架空のものとなります。実在の企業、店舗、個人等とは関係がございません。

本書記載の情報は、2024年12月現在のもを掲載しています。ご利用時には、変更されている可能性があります。OSやソフトウェアは更新や変更が行われる場合があり、本書での説明とは機能や画面などが異なってしまうこともあり得ます。OSやソフトウェア等の内容が異なることを理由とする、本書の返本、交換および返金には応じられませんので、あらかじめご了承ください。

以上の注意事項をご承諾いただいた上で、本書をご利用願います。これらの注意事項に関わる理由に基づく、返金、返本を含む、あらゆる対処を、技術評論社および著者は行いません。あらかじめ、ご承知おきください。

■ 本書に掲載した会社名、プログラム名、システム名などは、米国およびその他の国における登録商標または商標です。なお、本文に™マーク、®マークは明記していません。

第 1 章
—
入門編

プロフィール サイトを作ろう

難易度



基礎学習を終えたあとの
ウォーミングアップ用の練習サイト

基本的なHTMLとCSSだけで作れるかんたんなWebサイトを作ってみましょう。この章では主に、画像やコンテンツを横並びにするFlexboxの使い方について学びます。

SECTION

1-1 | 完成イメージの確認

完成サイトのイメージ

この章で作成するサンプルサイトの完成イメージは、以下の通りです。ロゴとグローバルナビゲーションの下に、メインビジュアル、プロフィール、作品一覧が入るシンプルなサイトです。

■ トップページ(PC)



■ トップページ(モバイル)

Dry Flower Life Profile Works



ドライフラワーのある生活で暮らしを豊かに

アンティークな雰囲気を持ったドライフラワーは、インテリアとして生活を彩るだけでなく香りを楽しむこともできます。
あなたの暮らしにもドライフラワーを取り入れてみませんか？

Profile

自己紹介






趣味でドライフラワー作りを始め、現在はドライフラワー作家として活動しています。
インテリアとしてだけでなく、お誕生日や記念日等、お祝い用のフラワーアレンジメントも行っています。



Works

作品一覧





© Dry Flower Life

1-2 | コーディングポイントの確認

コーディングポイント

作成するサンプルサイトのコーディングポイントは下記の通りです。

☑ HTMLの基本タグ(header、footer、main、nav、section、div、img、p)の使い方

サイト制作で使用する基本的なタグを一通り使用します。

☑ 見出しタグ(h1、h2)の使い方

サイトやページの一番重要なタイトルにh1タグ、各コンテンツのタイトルにh2タグを設定します。

☑ Flexboxを使用した横並びの方法

コンテンツを横並びにする際に使用します。サイト制作でよく使用するテクニックですので、本章の「1-5 ヘッダーを作ろう」(P.29)で詳しく解説します。

☑ リストタグ(ul、li)の使い方

リスト形式の文章やコンテンツを記述する際に使用します。本章の「1-5 ヘッダーを作ろう」(P.29)で詳しく解説します。

☑ レスポンシブ対応

PCだけでなく、タブレットやスマートフォンの画面サイズでも最適に表示されるようにします。レスポンシブ対応を行うためのブレイクポイントの設定について、本章の「1-4 全体の枠組みを作ろう」(P.22)で詳しく解説します。

コーディング準備

コーディングを始めるための準備を行っていきましょう。

☑ ダウンロード

本章のデザインデータ、素材、完成サイトのソースコードは、P.8を参考にダウンロードしてください。

☑ デザインデータを確認する

「design」フォルダの中のデザインデータを確認しましょう。

☑ 素材を確認する

「img」フォルダの中にある素材を確認しましょう。

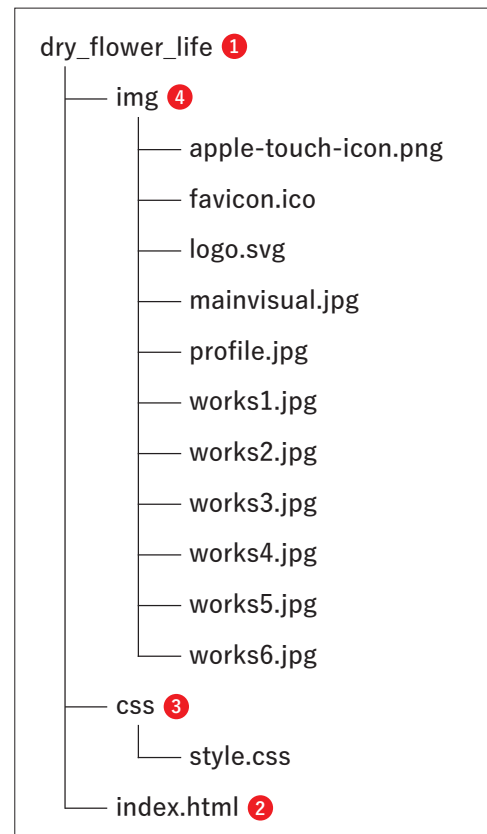
☑ 完成サイト

完成サイトのソースコードは「src」フォルダの中にあるデータから確認できます。

☑ フォルダとファイルを作成する

サンプルサイトを作るための、フォルダとファイルを準備します。フォルダ構成は、下記の通りです。

① 作業用フォルダ	作業用フォルダとして「dry_flower_life」という名前のフォルダを作成します。
② HTMLファイル	「dry_flower_life」フォルダの中に「index.html」という名前で作成します。
③ CSSファイル	「dry_flower_life」フォルダの中に「css」という名前のフォルダを作成し、その中に任意の名前(今回は「style.css」)で作成します。
④ 画像フォルダ	「dry_flower_life」フォルダの中に「img」という名前のフォルダを作成し、ダウンロードした「img」フォルダの中の画像をまとめて入れます。



SECTION

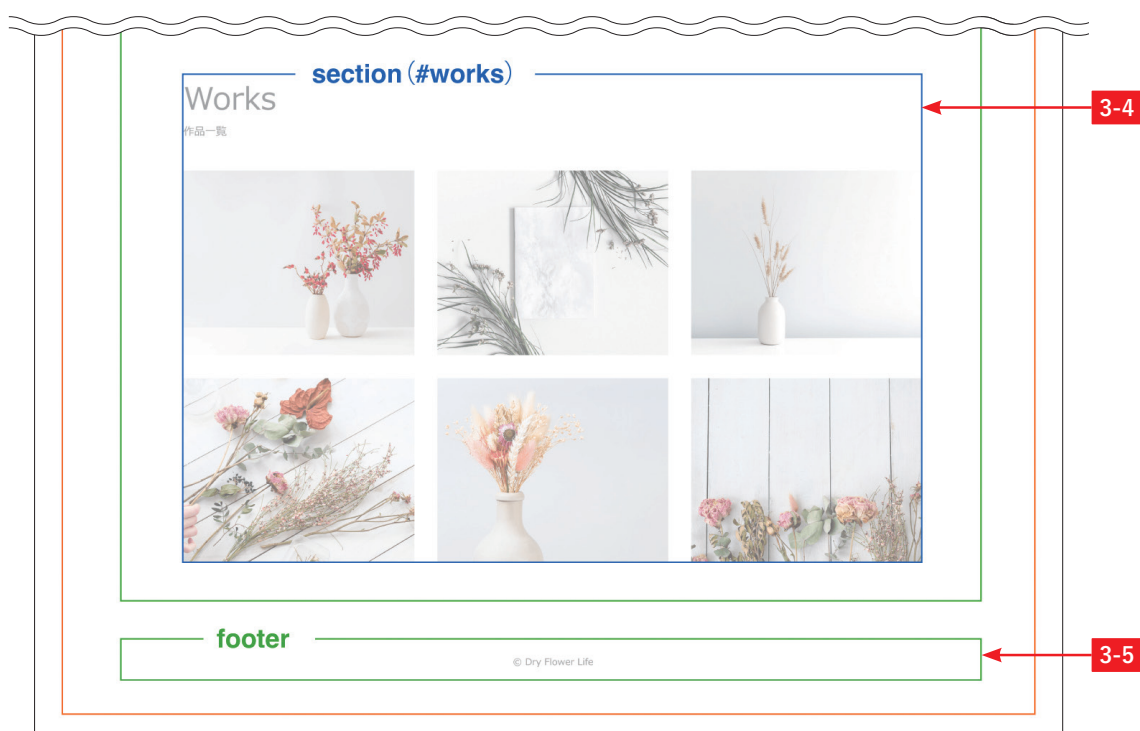
1-4 | 全体の枠組みを作ろう

レイアウト構成の確認

サイト全体の枠組みを作成します。ここでの学習の目的は、画像やコンテンツを横並びにするFlexboxの使い方について学ぶことです。各パーツのコーディングを始める前に、全体のサイト構成の確認とHTMLファイル、CSSファイルの共通部分をコーディングしておくことで、後の作業がスムーズに進められます。

全体のレイアウト構成は、以下の通りです。大きく分けると、header、main、footerの3つのブロックで構成されます。また、mainの中には各コンテンツが入ります。





① body	Webサイトのヘッダー、フッター含むコンテンツ全体を囲みます。
② header	ロゴとグローバルナビゲーションを囲みます。
③ main	コンテンツのメインエリア全体を囲みます。
3-1 div (.mainvisual)	メインビジュアルをdivタグで囲みます。divタグはコンテンツをグルーピングしたり、レイアウト調整を行いたい場合に使用します。divはタグ自体に意味を持たないため、レイアウトや装飾を変更したい際に汎用的に使うことができます。
3-2 div (.introduction)	メインビジュアル下のテキストをdivタグで囲みます。
3-3 section (#profile)	プロフィールエリア全体をsectionタグで囲みます。
3-4 section (#works)	作品一覧エリア全体をsectionタグで囲みます。
3-5 footer	フッターを囲みます。

OnePoint divタグとsectionタグの違いと使い分けについて

sectionタグは、文章やコンテンツの1つのまとまりを作るために使用します。divタグが意味を持たないのに対し、sectionタグはそのコンテンツが1つのまとまりであるという意味をもちます。divタグを使用するか、sectionタグを使うかは、そのまとまりにh1～h6といった見出しがつけられるかどうかを1つの基準にするとわかりやすいです。コンテンツのまとまりにh1～h6の見出しがつけられる場合は、sectionタグが適しています。

HTMLのコーディング

全体の枠組みと head 部分のコーディングを行っていきましょう。head 部分には、meta タグや CSS ファイル等、外部ファイルの読み込みを記述します。文字化けを防ぐための charset と Web サイトのタイトルと説明を記載するための title、description タグは基本的には必須で設定します。また、レスポンシブの Web サイトを作る場合は、viewport の記述も必須となります。

index.html

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="utf-8"> ①
    <title>Dry Flower Life</title> ②
    <meta name="description" content="趣味のドライフラワーを紹介するプロフィールサイトです。これまでに作った作品を掲載していますので、よろしければご覧になってみてください。"> ②
    <meta name="viewport" content="width=device-width, initial-scale=1"> ③
    <link rel="icon" href="img/favicon.ico"> ④
    <link rel="apple-touch-icon" href="img/apple-touch-icon.png"> ⑤
    <link rel="stylesheet" href="https://unpkg.com/ress/dist/ress.min.css"> ⑥
    <link rel="stylesheet" href="css/style.css"> ⑦
  </head>

  <body>
  </body>
</html>
```

① charset

文字化けを防ぐために、使用する文字コードを指定します。通常は UTF-8 を指定します。

② title と description

title はサイトのタイトル、description はサイトのかんたんな説明を記載します。基本は必須で設定します。

③ viewport

ブラウザの表示領域を指定します。レスポンシブ対応を行う際は、必須で設定します。

④ ファビコン

ブラウザのタブなどに表示するためのアイコンを設定します。

5 アップルタッチアイコン

スマートフォンやタブレットのホーム画面に追加した時に表示されるアイコンを設定します。

6 リセットCSS

CSSを初期化するための、リセット用のCSSを読み込みます。本書では、「ress.min.css」を使用します。

7 CSS読み込み

CSSファイル「style.css」を読み込みます。参照先は、cssフォルダ内の「css/style.css」です。

CSSのコーディング

CSSのコーディングを行っていきましょう。html、body、img、li、aタグ等、共通のタグに対して基本の設定を行います。一般的には、htmlタグに対して「font-size: 100%;」を設定したり、bodyタグでサイト内のメインフォントとフォントカラーを設定したりします。ここで設定するのはサイト全体の基本設定ですので、個別のレイアウトや装飾については、以降のCSSでこの設定内容を上書きしたり追加していきます。

style.css

```
@charset "UTF-8";

html { ①
  font-size: 100%;
}
body { ②
  color: #707070;
  font-family: sans-serif;
}
img { ③
  max-width: 100%;
  vertical-align: bottom;
}
li { ④
  list-style: none;
}
a { ⑤
  color: #707070;
  text-decoration: none;
}
```

```

}
a:hover { ❸
  opacity: 0.7;
}

/*-----
スマートフォン
-----*/

@media screen and (max-width: 767px) { ❹
}

```

❶ html

htmlのフォントサイズを100%で指定しておくことで、ユーザーがブラウザで設定したフォントサイズが正しく反映されるようになります。

❷ body

サイト全体の基本となるフォントカラーやフォントファミリー等を指定します。

❸ img

レスポンシブで作る場合、すべての画像に最大幅を100%で指定しておくことで、親のコンテンツから画像がはみ出すのを防ぐことができます。また、「vertical-align: bottom;」を指定しておくことで、画像の下にできる隙間を消すことができます。

❹ li

サイト全体で共通するliタグの設定などがあれば指定します。本章のサイトでは、リストのスタイルプロパティ（初期設定でリスト項目の先頭につく丸や四角などのマーク）は使用しないため、「list-style: none;」でスタイルを消しておきます。

❺ a

リンクのテキストカラーや装飾等を設定します。本章のサイトではリンクに下線を引かないため、「text-decoration: none;」を設定します。

❻ a:hover

カーソルをリンクの上に乗せた際の装飾を指定します。本章では、よく使用される「opacity: 0.7;」を設定して、カーソルに乗せた際に少しだけ透過するようにします。

■ カーソルを乗せる前

Dry Flower Life

■ カーソルを乗せた後

Dry Flower Life

⑦ ブレイクポイント

767px以下の場合にはスマートフォン用のCSSを適用させるため、メディアクエリに「@media screen and (max-width: 767px)」を指定します。メディアクエリとは、異なるデバイスやビューポートに対してページが最適な表示になるようにスタイルを調整するための機能です。

☑ ブレイクポイントについて

レスポンシブに対応させるため、レイアウトの切り替えを行うためのブレイクポイントを設定します。ブレイクポイントの数値や個数に決まりはありませんので、コーディング仕様等で指定がない場合はデザインにあわせて調整していきます。以下で、よく使われる指定方法をご紹介します。

PC、タブレットをメインにしたデザインの場合

PC、タブレット用のCSSを記述してから、メディアクエリの「max-width」でスマートフォン用のCSSを追加、上書きします。スマートフォン用のブレイクポイントとしてよく使用される767px以下で切り替えを行う場合、以下のようになります。

例 767px以下の場合にスマートフォン用のデザインを適用

```
@charset "UTF-8";

/* PC用のCSSを記述する */

@media screen and (max-width: 767px) {
  /* スマートフォン用のCSSを追加、上書きする */
}
```

スマートフォン、タブレットをメインにしたデザインの場合(モバイルファースト)

スマートフォン、タブレット用のCSSを記述してから、メディアクエリの「min-width」でPC用のCSSを追加、上書きします。PC用のブレイクポイントとしてよく使用される1025pxで切り替えを行う場合、以下のようになります。

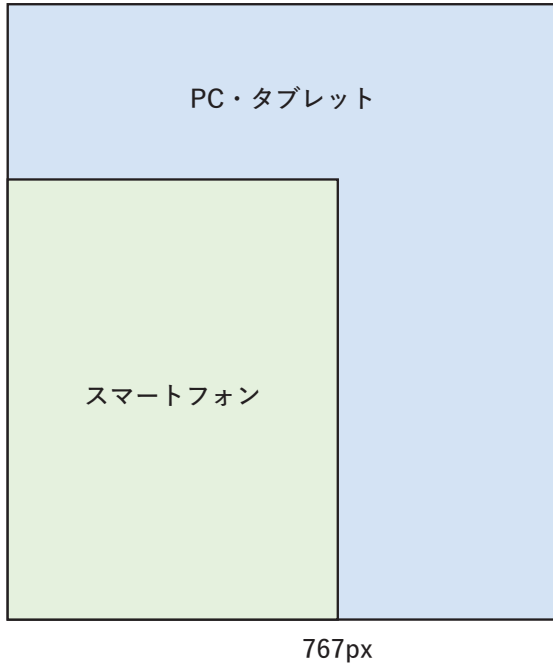
例 1025px以上の場合にPC用のデザインを適用

```
@charset "UTF-8";

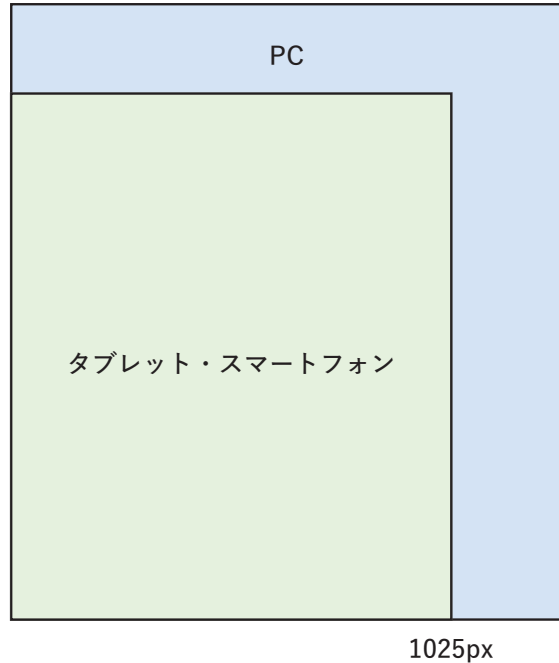
/* スマートフォン用のCSSを記述する */
```

```
@media screen and (min-width: 1025px) {  
  /* PC用のCSSを追加、上書きする */  
}
```

■ 「max-width: 767px」の場合



■ 「min-width: 1025px」の場合



今回はPCとタブレットは同じ表示にして、スマートフォンだけ表示を切り替えたいので、PCを基準にコーディングしていきます。ブレイクポイントは768px (768px以上がPC表示、767px以下がスマートフォン表示) に設定します。

ブレイクポイントの数値の決め方については、特に仕様等で指定がなければPC + タブレットとスマートフォンで分ける768pxがよく使用されていておすすめです。デザインによって複数のポイントで表示を切り替えたい場合は、ブレイクポイントを複数個設定してもOKです。

以上で、全体のHTML、CSSのコーディングは完了です。

レイアウト構成の確認

ページのヘッダー部分を作成します。ここでの学習の目的は、header タグの使い方とグローバルナビゲーションの記述方法について学ぶことです。ヘッダーはロゴとグローバルナビゲーションで構成されます。

ヘッダーのレイアウト構成は、以下の通りです。全体を header タグで囲み、ロゴとグローバルナビゲーションを横並びにします。



<p>① ロゴ</p>	<p>h1 タグで囲みます。h1 タグは、サイトやページでもっとも重要な見出しに対して設定します。トップページの場合、サイト名やサイト名を含む説明テキストに対して設定することが一般的ですが、テキストがない場合はロゴ画像に設定したりします。その際、ロゴ画像の alt 属性にはサイト名等、そのサイトを表すテキストを必ず設定しておきましょう。</p>
<p>② グローバルナビゲーション</p>	<p>グローバルナビゲーションは、nav タグを使用します。メニュー部分は ul、li タグを使って記述します。ul、li タグは、リスト形式のテキストやコンテンツに対して使用します。リスト形式のタグには、ul タグの他に、ol タグと dl タグがあります。</p>

OnePoint リストタグについて

リストタグの種類と使用方法は、以下の通りです。

ul、li: 順不同のリスト

箇条書きのように、順序の決まっていないリストに使用します。

```
<ul>
  <li>リストの項目A</li>
  <li>リストの項目B</li>
  <li>リストの項目C</li>
</ul>
```

- リストの項目A
- リストの項目B
- リストの項目C

ol、li: 順序性のあるリスト

1番目、2番目のように、順序が決まっているリストに使用します。

```
<ol>
  <li>リストの1番目</li>
  <li>リストの2番目</li>
  <li>リストの3番目</li>
</ol>
```

1. リストの1番目
2. リストの2番目
3. リストの3番目

dl、dt、dd: 用語の定義リスト

「説明する言葉」と「説明文」のように、言葉と説明がセットになっているリストに使用します。

```
<dl>
  <dt>言葉1</dt>
  <dd>言葉1の説明文</dd>
  <dt>言葉2</dt>
  <dd>言葉2の説明文</dd>
  <dt>言葉3</dt>
  <dd>言葉3の説明文</dd>
</dl>
```

- 言葉1
言葉1の説明文
- 言葉2
言葉2の説明文
- 言葉3
言葉3の説明文

HTMLのコーディング

HTMLのコーディングを行っていきましょう。ヘッダーは、全ページ共通で表示されるページ上段の部分を指します。一般的には、ロゴやグローバルナビゲーション等がヘッダーにあたります。ヘッダーをコーディングする際は、全体をheaderタグで囲みます。今回はロゴとグローバルナビゲーションがヘッダーに該当するため、ロゴをh1タグ、グローバルナビゲーションをnavタグで囲み、headerタグの中に入れます。

index.html

```
<header id="header" class="wrapper"> ①
  <h1 class="logo">
    <a href="index.html">
       ②
    </a>
  </h1>
  <nav>
    <ul class="navi">
      <li><a href="#profile">Profile</a></li> ③
    </ul>
  </nav>
```

```
<li><a href="#works">Works</a></li>
</ul>
</nav>
</header>
```

① 全体

headerタグにidを設定します。また、全体の横幅を設定するための「wrapper」というクラスを設定します。クラス名は任意ですが、全体を囲むクラスの名前として「wrapper」や「container」という名前がよく使用されます。HTMLタグに名前をつける方法には、「id」と「class」という2種類の指定方法があります。両者の使い分けについては、以下の「idとclassの使い分けについて」で詳しく解説しています。

② ロゴ画像

全体をh1タグで囲み、さらにaタグでトップページへのリンクを設定します。ページがたくさんある場合、ロゴにトップページへのリンクが張ってあるとサイトを巡回しやすくなるので、ユーザビリティの向上につながります。

③ グローバルナビゲーション

グローバルナビゲーションは、navタグで囲みます。中身のメニューは、リストタグ(ul、liタグ)で記述します。各メニューをaタグで囲み、それぞれのセクションへジャンプできるようにページ内リンクを設定しておきます。

OnePoint idとclassの使い分けについて

CSSを記述するためにHTMLタグに名前をつける方法として、「id」と「class」2種類の指定方法があります。両者の違いは、idが同じページ内で1度しか使用できないのに対し、classは何度でも使用することができます。一般的にはclassを使用することが多いですが、主に下記のような場合にidを使用します。

- 内部リンクの遷移先として指定する場合(遷移先が重複しないよう一意にする必要があるため)
- JavaScriptなどの処理で要素を特定する必要がある場合
- コード内で明示的に一度しか使用していないことを示したい場合

本書では、ヘッダー、フッターおよび内部リンクの遷移先、JavaScriptの処理で使用する要素に対してidを使用し、その他の要素に対してはclassを使用することとします。

CSSのコーディング

CSSのコーディングを行っていきましょう。ポイントは、「全体の横幅を設定するためのwrapperクラス」「ロゴとグローバルナビゲーションの横並び」「ナビゲーションメニューの横並び」の3点です。

wrapperクラスは、レイアウトの横幅を定義するための共通クラスとして設定します。横並びについてはFlexboxがよく使われますので、両方ともFlexboxを使ってコーディングしていきます。

style.css

```
.wrapper { ❶  
  max-width: 1000px;  
  padding: 0 20px;  
  margin: 0 auto;  
}  
  
#header { ❷  
  display: flex;  
  align-items: center;  
  justify-content: space-between;  
  padding-top: 35px;  
  padding-bottom: 35px;  
}  
  
#header .logo { ❸  
  max-width: 190px;  
  line-height: 0;  
}  
  
#header .logo a { ❹  
  display: block;  
}  
  
#header .navi { ❺  
  display: flex;  
  align-items: center;  
}  
  
#header .navi li { ❻  
  font-size: 14px;  
  margin-left: 40px;  
}
```

❶ .wrapper

レイアウトの最大幅「max-width」と画面幅を狭めた際の両サイドの余白「padding: 0 20px;」、中央寄せ「margin: 0 auto;」を設定します。「margin: 0 auto;」で、横幅を設定したボックスを中央に配置します。

❷ #header

「display: flex;」で、ロゴとグローバルナビゲーションを横並びに設定します。その際に、「align-items:

center;」で縦方向に中央揃え、「justify-content: space-between;」で横方向に均等割り付け(コンテンツが2つの場合は両端に配置される)を設定します。

3 #header .logo

「max-width: 190px;」で、ロゴ画像の最大幅を設定します。h1 タグの行間をそのまま使用すると画像の上下に余白が入ってしまうため、「line-height: 0;」で行間を0にすることで、上下にできる余白を消してロゴ画像の高さにあわせませす。

4 #header .logo a

a タグの display プロパティは初期値に「inline」が設定されており、そのままではリンクの範囲が親要素からずれてしまいます。そのため、「display: block;」を設定してリンクの範囲を親要素の範囲まで広げます。

OnePoint display プロパティについて

サイト制作において、display プロパティは非常によく使われる重要なプロパティです。以下に、使用頻度の高い値をご紹介します。

● inline

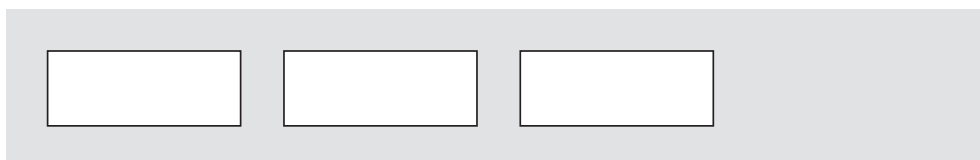
要素は横に並び、横幅や高さは指定できません。主に、文章の一部として使用されます。

タグ例

a、img、span 等

特徴

- 要素は横に並ぶ
- width と height は指定できない
- padding と margin は左右のみ指定できる (padding は上下の指定ができるが、正しく適用されないため非推奨)



● inline-block

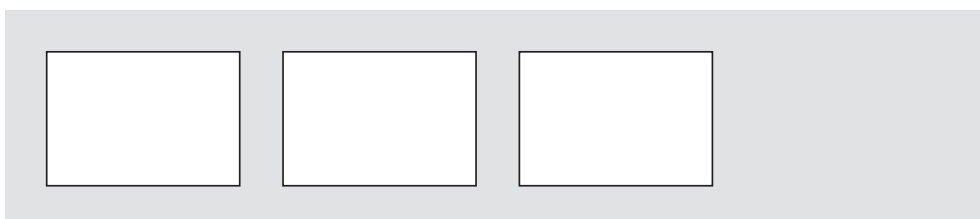
要素は横に並び、横幅、高さ、余白の調整が可能です。inline の要素に対してレイアウト調整を行いたい場合などに使用します。

タグ例

初期値が inline-block のタグはなし

特徴

- 要素は横に並ぶ
- width と height が指定できる
- padding と margin が指定できる



● block

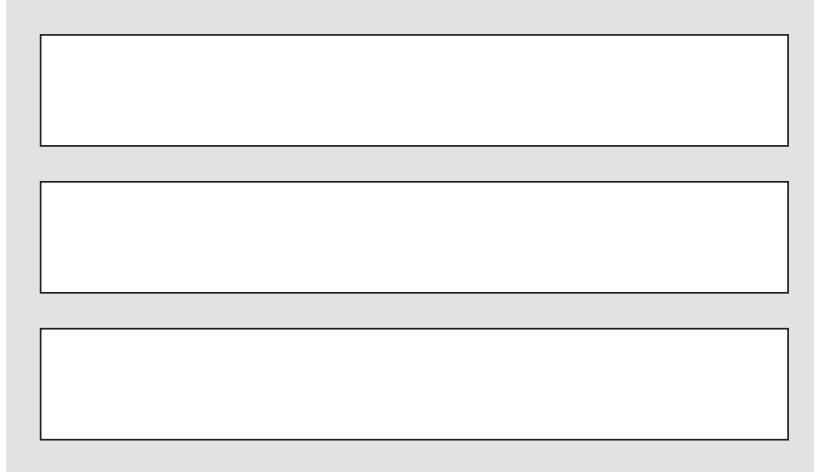
要素は縦に並び、横幅、高さ、余白の調整が可能です。inlineの要素を縦に並べたい場合などに使用します。

タグ例

div、p、h1～(見出しタグ)等

特徴

- 要素は縦に並ぶ
- widthとheightが指定できる
- paddingとmarginが指定できる



● flex

Flexbox(フレックスボックス)と呼ばれ、主に要素を横に並べる際に使用します。「inline」「inline-block」と比べて並び方や折り返し方の設定方法が幅広く用意されているため、要素を横並びにする際によく使用されます。Flexboxについては、「Flexboxについて」で詳しく解説しています(P.35参照)。

● grid

グリッドレイアウトとは、格子状のマスを使ってレイアウトを組み立てていく手法のことを言います。要素を行と列のグリッド状に配置することができ、複雑なレイアウトの作成も可能です。gridについては、4章の「グリッドレイアウトについて」で詳しく解説しています(P.247参照)。

● none

要素が非表示になります。主に、PCの場合は表示してスマートフォンの場合は表示しない、最初は表示されていない要素をボタンを押したタイミングで表示する等、要素の表示と非表示を切り替える際に使用します。

5 #header .navi

「display: flex;」で、メニューを横並びに設定します。headerタグと同じく「align-items: center;」で縦方向の中央に揃えます。

6 #header .navi li

メニューの各項目の間に余白を入れるため、「margin-left: 40px;」を設定します。ここでは右端を揃えたいので、margin-rightではなくmargin-leftを指定しています。

OnePoint Flexboxについて

Flexboxは、コンテンツを横並びにする際によく使用するテクニックです。実務でも頻繁に使用しますので、今回はその中でも「display: flex;」とともに使用する使用頻度の高いプロパティをご紹介します。

justify-content (横方向の配置場所を指定します)

flex-start	左揃え (初期値)
flex-end	右揃え
center	中央揃え
space-between	均等割り付け (両端に揃うよう均等に配置されます)

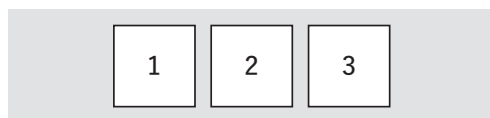
● flex-start



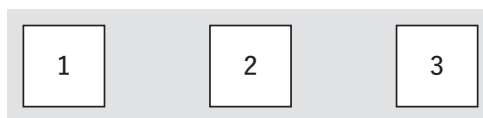
● flex-end



● center

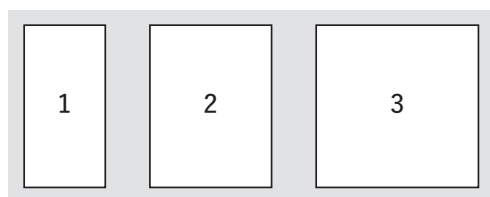


● space-between

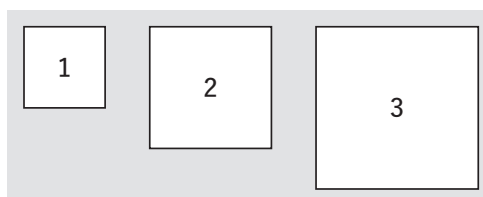
**align-items (縦方向の配置場所を指定します)**

stretch	上揃え (アイテムの高さは揃う 初期値)
flex-start	上揃え (アイテムの高さは揃わない)
flex-end	下揃え
center	中央揃え

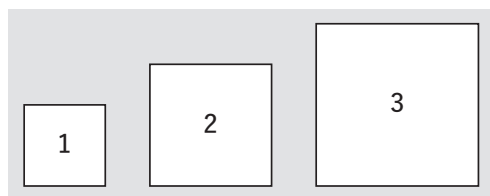
● stretch



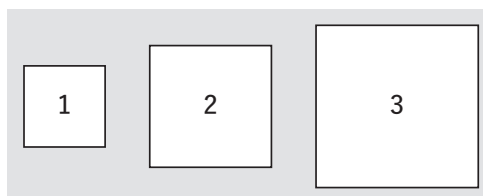
● flex-start



● flex-end



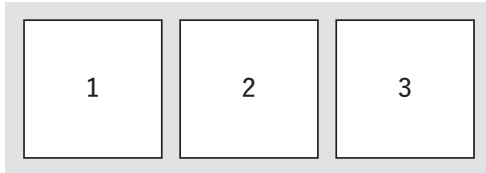
● center



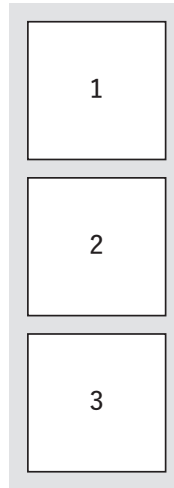
flex-direction(アイテムの配置方向を指定します)

row	左から右に向けての配置になります(初期値)。
column	上から下に向けての配置順になります(よく使用するパターンとして、PCが横並びスマートフォンは縦並びのデザインの場合、PCで「display: flex;」を設定して横並びにし、スマートフォンで「flex-direction: column;」を追加して縦並びにします。
row-reverse	「row」の逆順に配置します。
column-reverse	「column」の逆順に配置します。

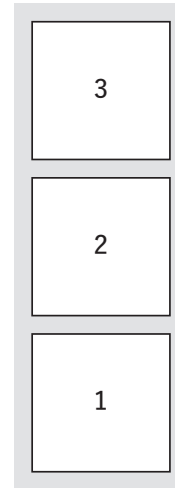
● row



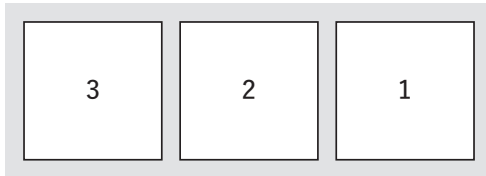
● column



● column-reverse

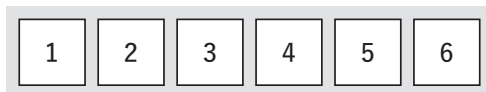


● row-reverse

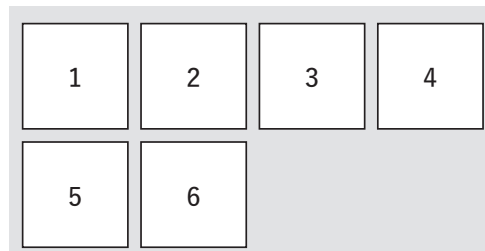
**flex-wrap(アイテムの折り返しを指定します)**

nowrap	折り返しなし(初期値)
wrap	アイテムは左から順に配置され、親要素の幅からはみ出してしまう場合に折り返して複数行で表示されます。2行目以降のアイテムも左から順に配置されます。

● nowrap



● wrap



レスポンス対応

レスポンス用のCSSを設定します。ヘッダーのレスポンス対応を行っていきましょう。メディアクエリに、スマートフォン用のCSSをコーディングしていきます。すべてのコーディングが終わってからまとめてレスポンスの調整を行ってもよいのですが、レイアウトが崩れた場合に修正箇所が特定しづらくなるので、はじめのうちはパーツごとにレスポンス対応を行っていくのがおすすめです。

■ PC表示



■ モバイル表示



style.css

```
@media screen and (max-width: 767px) {  
  #header {  
    padding-top: 25px;  
    padding-bottom: 25px;  
  }  
  #header .logo {  
    max-width: 120px;  
  }  
}
```

ヘッダーはPC、スマートフォンともにデザインに違いはありませんので、ロゴのサイズを小さくして上下の余白を調整するだけで完了です。

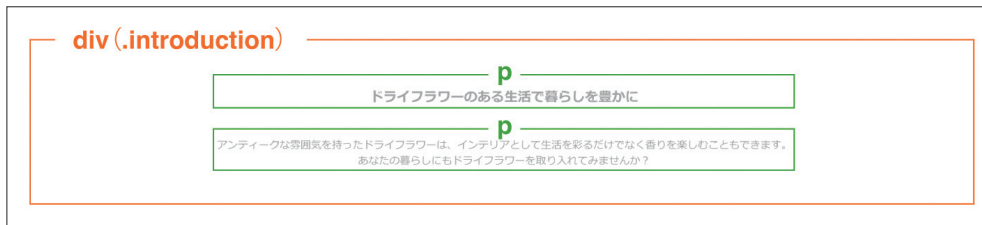
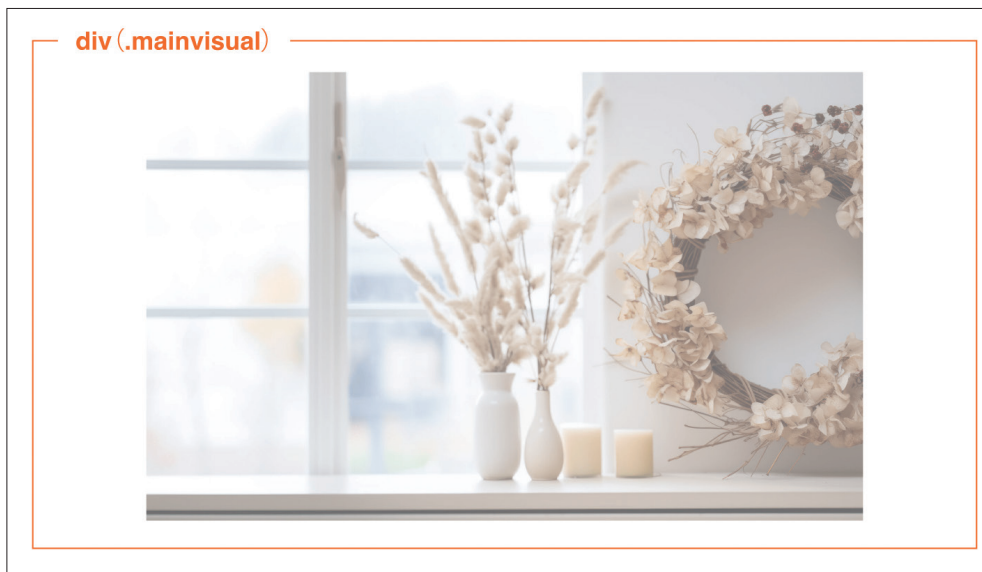
以上で、ヘッダーのHTML、CSSのコーディングは完了です。

SECTION

1-6 | メインビジュアルとイントロダクションを作ろう

レイアウト構成の確認

メインビジュアルとイントロダクションを作成します。ここでの学習の目的は、画像を囲むいろいろなタグについて学ぶことです。サイト制作の基本となる、画像とテキストのコーディングを行います。メインビジュアルとイントロダクションのレイアウト構成は、以下の通りです。画像とテキストのエリアを、それぞれdivタグで囲みます。



HTMLのコーディング

HTMLのコーディングを行っていきましょう。ページのメインコンテンツ全体をmainタグで囲みます。mainタグの中にdivタグでメインビジュアル(mainvisual)と説明テキスト(introduction)のブロックを作り、それぞれの中にimgタグで画像の設定、pタグでテキストの設定を行っていきます。

`index.html``<main class="wrapper"> 1`

```
<div class="mainvisual">
   2
</div>
```

```
<div class="introduction"> 3
  <p class="catchphrase">ドライフラワーのある生活で暮らしを豊かに</p>
  <p class="text">アンティークな雰囲気を持ったドライフラワーは、インテリアとして生活を彩るだけでなく香りを楽しむこともできます。<br>
  あなたの暮らしにもドライフラワーを取り入れてみませんか? </p>
</div>
```

```
</main>
```

1 main タグ

メインコンテンツ全体を main タグで囲みます。横幅を指定するため、ヘッダーで作成した wrapper クラスを設定します。

2 メインビジュアル

img タグで記述し、全体を div タグで囲みます。

OnePoint 画像を囲むタグについて

画像は、意味を明確にしたりレイアウト調整をしやすくするためにタグで囲んで使用されることが多いです。一般的に、下記のタグで囲んで使用されます。今回のメインビジュアルは、div タグで囲みます。

p タグ	文章中で使用する画像で、文脈上必要になってくる画像に使用します。この場合、img タグの alt 属性には画像の内容を説明するような文章を設定します。
figure タグ	文章から参照されるようなイラスト、図、写真等に使用します。figure タグを使用した場合も、img タグの alt 属性に画像の内容を設定します。また、figcaption で注釈をつけることもできます。
div タグ	上記以外の場合で、画像に対してデザインやレイアウトを変更したい場合に使用します。

3 イン트로ダクション

エリア全体を div タグで囲み、各テキストは p タグで記述します。

CSSのコーディング

CSSのコーディングを行っていきましょう。メインビジュアル、イントロダクションともに、marginの設定を行い余白を調整します。テキストは「text-align: center;」で中央揃えにし、pタグにフォントの設定を行います。

```
style.css

.mainvisual { ①
  margin-bottom: 80px;
}

.introduction { ②
  margin-bottom: 80px;
  text-align: center;
}

.introduction .catchphrase { ③
  font-size: 18px;
  font-weight: bold;
  margin-bottom: 40px;
}

.introduction .text { ④
  font-size: 14px;
}
```

① .mainvisual

「margin-bottom: 80px;」で、メインビジュアルの下に80pxの余白を設定します。

② .introduction

「margin-bottom: 80px;」で、イントロダクションの下に80pxの余白を設定します。また、「text-align: center;」でイントロダクション内のテキストを中央寄せにします。

③ .introduction .catchphrase

「font-size: 18px;」「font-weight: bold;」で、キャッチフレーズのフォントを18pxの太字に設定します。「margin-bottom: 40px;」で、キャッチフレーズの下に40pxの余白を設定します。

④ .introduction .text

「font-size: 14px;」で、キャッチフレーズ下のテキストのフォントサイズを14pxに設定します。

レスポンシブ対応

レスポンシブ用のCSSを設定します。メインビジュアルとイントロダクション下の余白を調整します。また、イントロダクションのフォントサイズを小さくして左寄せにします。ヘッダーと同じく、メディアクエリに追記していきます。

style.css

```
@media screen and (max-width: 767px) {  
  .mainvisual {  
    margin-bottom: 50px;  
  }  
  .introduction {  
    margin-bottom: 50px;  
  }  
  .introduction .catchphrase {  
    font-size: 16px;  
  }  
  .introduction .text {  
    text-align: left;  
  }  
}
```



ドライフラワーのある生活で暮らしを豊かに

アンティークな雰囲気を持ったドライフラワーは、インテリアとして生活を彩るだけでなく香りを楽しむこともできます。
あなたの暮らしにもドライフラワーを取り入れてみませんか？

以上で、メインビジュアルとイントロダクションのHTML、CSSのコーディングは完了です。

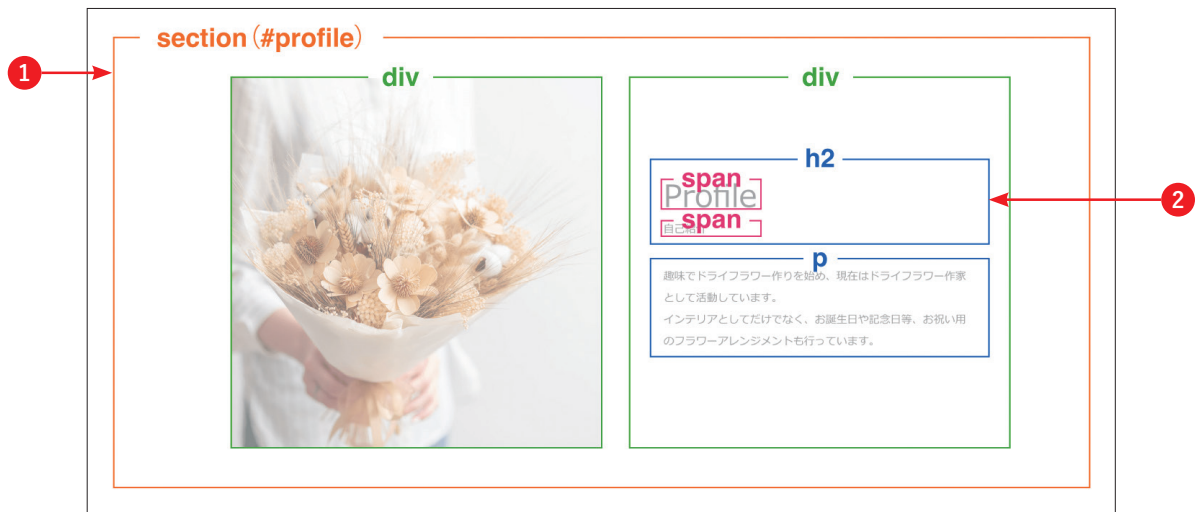
SECTION

1-7 | 「Profile」を作ろう

レイアウト構成の確認

「Profile」を作成します。この章の目的は、sectionタグ、h2タグ、spanタグなど各種タグの使い方について学ぶことです。プロフィール画像の横に、見出しとプロフィールの説明文が入ります。

「Profile」のレイアウト構成は、以下の通りです。全体をsectionタグで囲み、画像と説明文を横並びにします。タイトルはh2タグで記述します。

**1 全体**


全体をsectionタグで囲みます。見出しを含む1つのまとまった情報のため、divタグではなくsectionタグを使用します。divタグとsectionタグの違いについては、次ページの「divタグとsectionタグの違いについて」で詳しく解説しています。

2 タイトル

英語と日本語を、まとめてh2タグで囲みます。それぞれフォントサイズが異なるため、個別にCSSが設定できるようspanタグで囲んでおきます。

OnePoint div タグと span タグの違いについて

div タグと span タグは、どちらのタグもデザインの一部を変更したりレイアウトを調整したりする時に使用するタグで、タグ自体に意味を持たないのが特徴です。両者の違いと使い分けは、以下の通りです。

div タグ	<p>幅と高さの指定ができ、前後に改行が入るのが特徴です。主に、コンテンツをグルーピングしたりデザインやレイアウト調整を行う際に使用します。</p> <p>例 div タグでグルーピングして画像とテキストを横並びにする。</p> <div data-bbox="450 519 1289 958" style="border: 1px solid red; padding: 10px;">  <p>趣味でドライフラワー作りを始め、現在はドライフラワー作家として活動しています。インテリアとしてだけでなく、お誕生日や記念日等、お祝い用のフラワーアレンジメントも行っています。</p> </div>
span タグ	<p>幅と高さの指定ができず、前後に改行が入らないのが特徴です。主に、文章の中の一部のデザインを変更したい場合に使用します。</p> <p>例 span タグでテキストの一部を黄色背景にする。</p> <div data-bbox="450 1146 1289 1384" style="border: 1px solid gray; padding: 10px;"> <p>インテリアとしてだけでなく、お誕生日や記念日等、お祝い用のフラワーアレンジメントも行っています。</p> </div>

今回はタイトル中のテキストデザインを部分的に変更したいので、span タグを使用します。また、英語と日本語のタイトルを縦に配置したいので、display プロパティに「block」を設定してそれぞれの要素が縦に並ぶようにします。display プロパティの詳細については、「display プロパティについて」で詳しく解説しています (P.33 参照)。

HTML のコーディング

HTML のコーディングを行っていきましょう。「Profile」エリアは見出しを含む1つのまとまった情報のため、全体を section タグで囲みます。さらに画像とテキストを横並びに配置するため、それぞれのエリアを div タグでグルーピングします。タイトルは h2 タグで囲み、その中の英語と日本語にそれぞれの CSS を設定できるよう、span タグで囲んでおきます。

index.html

```
<section id="profile"> ❶
  <div class="img">
    
  </div>
  <div class="detail">
    <h2 class="section-title">
      <span class="en">Profile</span> ❷
      <span class="ja">自己紹介</span>
    </h2>
    <p>趣味でドライフラワー作りを始め、現在はドライフラワー作家として活動しています。
    <br>
    インテリアとしてだけでなく、お誕生日や記念日等、お祝い用のフラワーアレンジメントも
    行っています。</p>
  </div>
</section>
```

❶ section

グローバルナビゲーションのリンクをクリックした時に「Profile」セクションまでジャンプできるように、「profile」という名前でidを設定します。

❷ タイトル

h2タグで囲み、日本語と英語それぞれをspanタグで囲みます。

CSSのコーディング

CSSのコーディングを行っていきましょう。ポイントは、見出しのCSSを「Works」セクションでも使用できるように共通化しておくという点です。見出し全体を「section-title」、英語と日本語のspanタグをそれぞれ「en」「ja」というクラス名で定義し、フォントやマージン等の設定を行います。画像とテキストの横並びは、Flexboxを使用して縦中央で揃うようにします。

style.css

```
.section-title { ❶
  font-weight: normal;
  margin-bottom: 40px;
}
```

```
.section-title .en { ❷  
  display: block;  
  font-size: 40px;  
}  
.section-title .ja { ❷  
  display: block;  
  font-size: 14px;  
}  
  
#profile { ❸  
  display: flex;  
  align-items: center;  
  margin-bottom: 120px;  
}  
#profile .img {  
  width: 50%;  
}  
#profile .detail {  
  width: 50%;  
  padding-left: 80px;  
}  
#profile .detail p {  
  font-size: 14px;  
  line-height: 2;  
}
```

❶ タイトル (.section-title)

「Works」セクションのタイトルも同じデザインのため、「section-title」というクラス名で共通のCSSとして設定します。このようにデザインが同じものはCSSを共通化しておくことで、効率的なコードを書くことができます。

❷ タイトル (.section-title .en、.section-title .ja)

タイトルの英語と日本語にそれぞれ「display: block;」を設定して、縦に配置されるようにします。

❸ #profile

「display: flex;」で、画像とテキストエリアのボックスを横並びに設定します。「align-items: center;」で、縦中央で揃うようにします。



レスポンシブ対応

レスポンシブ用のCSSを設定します。スマートフォンでは、プロフィールの画像とテキストエリアを縦に並べます。その際に、「テキストエリア→画像」の並び順にするのがポイントです。

style.css

```
@media screen and (max-width: 767px) {
  .section-title { ❶
    margin-bottom: 25px;
  }
  .section-title .en { ❶
    font-size: 32px;
  }

  #profile { ❷
    flex-direction: column-reverse;
    margin-bottom: 60px;
  }
  #profile .img { ❸
    width: 100%;
  }
  #profile .detail { ❸
    width: 100%;
    padding-left: 0;
    margin-bottom: 20px;
  }
}
```

Profile

自己紹介

趣味でドライフラワー作りを始め、現在はドライフラワー作家として活動しています。

インテリアとしてだけでなく、お誕生日や記念日等、お祝い用のフラワーアレンジメントも行っています。



❶ .section-title、.section-title .en

英語のフォントサイズを小さくし、タイトル下の余白を調整します。

❷ #profile

「flex-direction: column-reverse;」で並び順を逆にして、テキストエリア→画像の順に表示します。

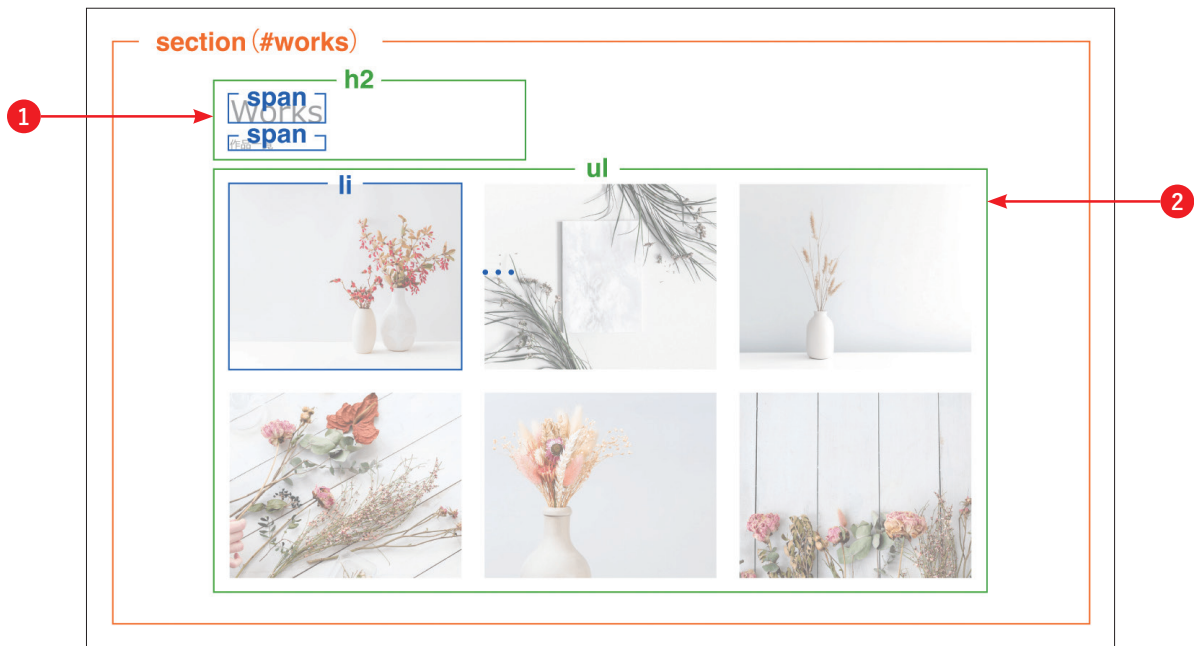
❸ 画像とテキストエリア (#profile .img、#profile .detail)

ともに横幅に「width: 100%」を設定して、横幅いっぱいまで広がるようにします。テキストエリアは、「padding-left: 0;」を設定して、左側の余白を削除しておきます。

以上で、「Profile」のHTML、CSSのコーディングは完了です。

レイアウト構成の確認

「Works」を作成します。Webサイトでよく見かける、画像一覧のコーディングを行っていきます。ここでの学習の目的は、リストタグを使った画像一覧の作り方と擬似クラスの使い方について学ぶことです。「Works」のレイアウト構成は、以下の通りです。画像の一覧は、リストタグを使用してコーディングしていきます。



① タイトル

「Profile」と同じく、h2タグとspanタグで囲みます。

② 画像一覧

画像一覧は、リストタグを使ってコーディングしていきます。今回は画像の並び順に順序性がないため、ul、liタグを使用します。

HTMLのコーディング

HTMLのコーディングを行っていきましょう。「Profile」セクションと同じく、全体をsectionタグで囲み「works」という名前でidを設定します。タイトルはProfileセクションで定義したCSSを使用するため、同じクラス名「section-title」を設定します。画像一覧はul、liタグで記述します。

index.html

```
<section id="works">
  <h2 class="section-title">
    <span class="en">Works</span>
    <span class="ja">作品一覧</span>
  </h2>

  <ul class="works-list">
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
  </ul>
</section>
```

ここでは、解説の必要な新しい書き方はありません。

CSSのコーディング

CSSのコーディングを行っていきましょう。画像一覧はFlexboxを使用して横並びにし、「flex-wrap: wrap;」で折り返されるようにします。また、liタグのwidthに32%を設定することで、画像が3枚ずつ並ぶようにします。

style.css

```
#works {
  margin-bottom: 120px;
}
#works .works-list { ①
  display: flex;
  flex-wrap: wrap;
}
#works .works-list li ②
  width: 32%;
  margin: 0 2% 2% 0;
```

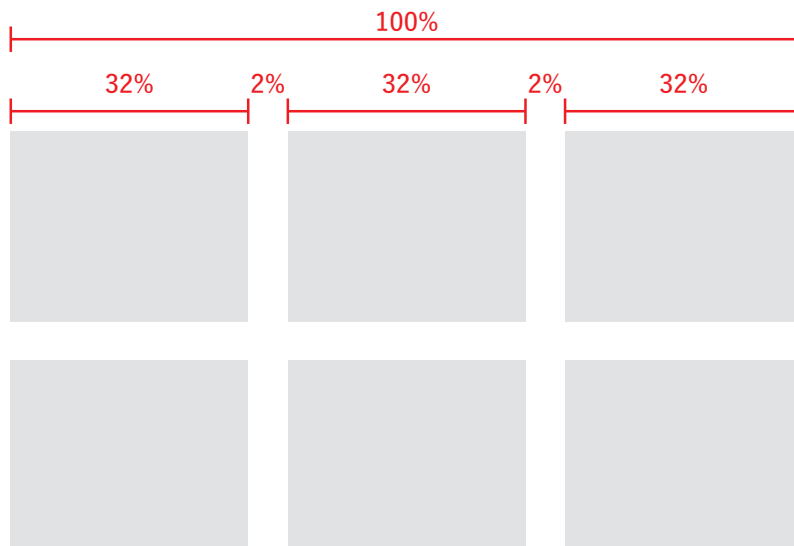
```
}  
#works .works-list li:nth-child(3n) {  
  margin-right: 0;  
}
```

① #works .works-list

「display: flex;」で横並びにして、「flex-wrap: wrap;」で折り返しの設定を行います。

② #works .works-list li

横3列に並べるので、「width: 32%;」を設定して画像の間に2%ずつ余白ができるようにします。3列目は右端に揃えるため右に余白は必要ないので、「#works .works-list li:nth-child(3n)」に「margin-right: 0;」を設定することで、3の倍数のアイテムだけ右側に余白が入らないようにします(「32% + 2% + 32% + 2% + 32%」で横幅の合計が100%になるようにします)。「nth-child」は擬似クラスと呼ばれ、特定の要素に対してCSSを設定することができます。詳細は、「擬似クラスについて」で詳しく解説します。



OnePoint 擬似クラスについて

擬似クラスは、特定の要素がある状態の場合にスタイルを適用させることができるセレクタです。今回のように、「何番目のliタグだけデザインを変えたい」といった場合も擬似クラスが有効です。ここでは、実務でよく使用する擬似クラスをご紹介します。

:hover	カーソルを要素の上に乗せた際に適用
:visited	訪問済みのリンクに適用
:checked	ラジオボタン、チェックボックスなどで選択されている場合に適用
:first-child	最初の要素に適用
:last-child	最後の要素に適用
:nth-child(odd)	奇数の要素に適用
:nth-child(even)	偶数の要素に適用
:nth-child(2)	2番目の要素に適用（※3番目以降は中の数字が変わります）
:nth-child(2n)	2の倍数の要素に適用（※3番目以降は中の数字が変わります）

レスポンシブ対応

レスポンシブ用のCSSを設定します。スマートフォンでは、画像一覧を横並びから縦並びに変更します。「flex-direction: column;」で、Flexboxの並び方向を縦に変更します。また、各アイテムの横幅を「width: 100%;」、余白を「margin: 0 0 20px;」で下側にだけ設定して、画像が横幅いっぱいまで広がるようにします。

style.css

```
@media screen and (max-width: 767px) {
  #works {
    margin-bottom: 60px;
  }
  #works .works-list {
    flex-direction: column;
  }
  #works .works-list li {
    width: 100%;
    margin: 0 0 20px;
  }
}
```

Works

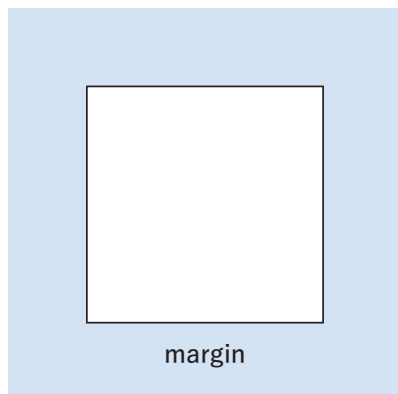
作品一覧



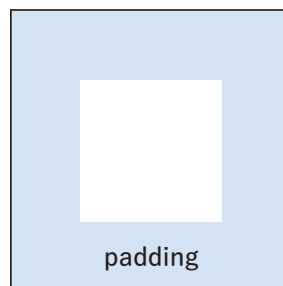
OnePoint marginとpaddingについて

marginとpaddingはどちらも余白の設定を表しますが、marginが「要素の外側の余白」を表すのに対し、paddingは「要素の内側の余白」を表します。

● margin:要素の外側の余白

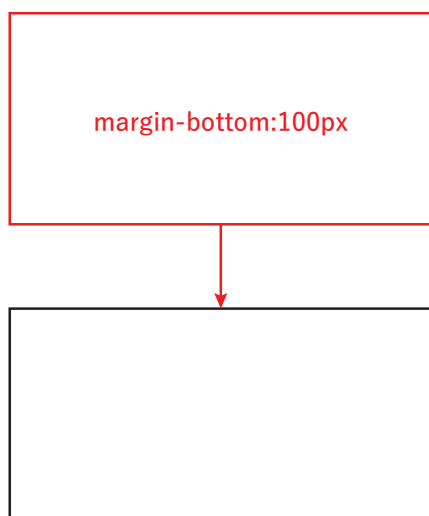


● padding:要素の内側の余白

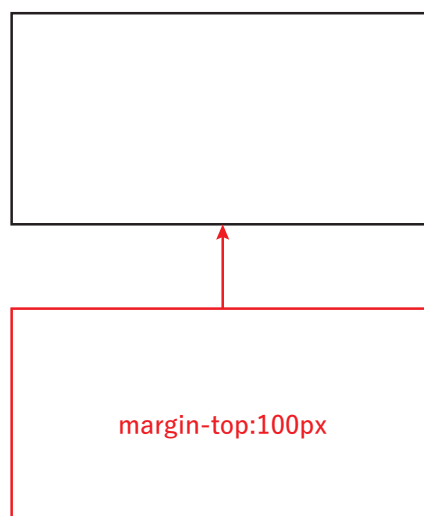
**OnePoint** marginは上か下か？

縦に並んだボックスの間に余白を設定する際、上のボックスに「margin-bottom」を設定するか下のボックスに「margin-top」を設定するかで悩まれる方も多いかと思います。結果はどちらも同じですが、その時々で上につけたり下につけたりしていると、どのmarginが効いているのかわからなくなるので、どちらかに基準を決めておくといよいでしょう。例えば、基本は余白を下に取りつつ、必要に応じて上に余白を取るという方法がわかりやすくおすすめです。

● 上のボックスに「margin-bottom」を設定した場合



● 下のボックスに「margin-top」を設定した場合



以上で、「Works」のHTML、CSSのコーディングは完了です。

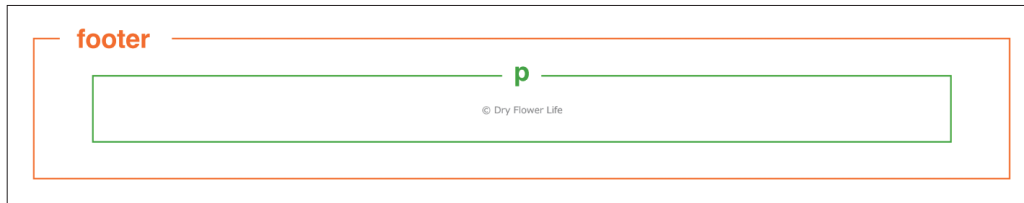
SECTION

1-9 | フッターを作ろう

レイアウト構成の確認

フッターを作成します。フッターはサイトの最下部にあたる部分で、通常、メニューやロゴ、コピーライト等が含まれる部分を指します。

フッターのレイアウト構成は、以下の通りです。コピーライトのエリア全体を footer タグで囲み、コピーライトは p タグで記述します。



HTMLのコーディング

HTMLのコーディングを行っていきましょう。全体を footer タグで囲み、コピーライトを p タグで記述します。

index.html

```
<footer id="footer" class="wrapper"> ①  
  <p class="copyright">© Dry Flower Life.</p>  
</footer>
```

① footer

フッター全体を footer タグで囲み、idを設定します。また、横幅を設定するため共通の wrapper クラスを設定します。

CSSのコーディング

CSSのコーディングを行っていきましょう。フォントサイズと余白を設定し、「text-align: center;」で中央寄せに設定します。

style.css

```
#footer {
  font-size: 12px;
  padding-bottom: 20px;
  text-align: center;
}
```

ここでは、解説の必要な新しい書き方はありません。フッターはスマートフォンも同じデザインのため、レスポンシブ用の調整は必要ないのでこれで完了です。

以上で、プロフィールサイトのコーディングはすべて完了です。

Column**クラス名のつけ方****基本ルール**

- 半角英数字を使用する(日本語は使用しない)
- アルファベットから開始する(数字から開始しない)
- 単語の組み合わせはハイフンかアンダースコアを使用する
- 極端に長い名前は避ける(例: container_content_main_box_info_item_text_1)
- 処理の内容をある程度推測できる名前にする

表記方法

クラス名をつける際に複数の英単語をつなげる方法として、主に下記の4種類の方法があります。本書は、ケバブケースで統一しています。

パスカルケース	単語の先頭を大文字にする	SectionTitle
キャメルケース	2つ目以降の単語の先頭を大文字にする	sectionTitle
スネークケース	単語の間をアンダースコアでつなぐ	section_title
ケバブケース	単語の間をハイフンでつなぐ	section-title

命名規則の参考

クラス名でよく使う単語の一部をまとめましたので、命名の際の参考にしてみてください。

● ブロック

container	全体を囲む
wrapper	外側を囲む
content	内容
outer	外側

inner	内側
area	範囲
box	ボックス



● パーツ

title	タイトル
text	テキスト
img	画像
list	リスト
item	項目
detail	詳細
summary	要約
description	説明

info	情報
news	お知らせ
work	実績
service	サービス内容
contact	お問い合わせ
link	リンク
btn	ボタン

なお、HTML、CSSをコーディングする際のスタイルガイドとして、Google社が提供している「Google HTML/CSS Style Guide」があります。英語版しかありませんが、HTML、CSSをコーディングする際のガイドとしてとても参考になりますので、ぜひ参照してみてください。

[Google HTML/CSS Style Guide] <https://google.github.io/styleguide/htmlcssguide.html>