CHAPTER 5

HTML,CSS基礎（２）

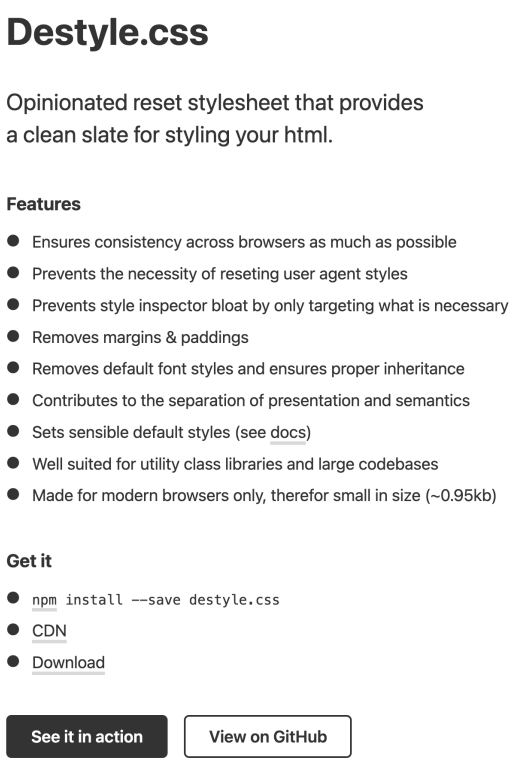
01 リセットCSS,ノーマライズCSS

- 「UAスタイルシート」を打ち消したり、
  ”いい感じ”に調整したりするテクニック
- “いいかんじにする” NormalizeCSS
- “完全にリセットする” resetCSS
- CDN (Content Delivery Network)の利用

01 リセットCSS,ノーマライズCSSのはじまり

https://meyerweb.com/eric/tools/css/reset/

01 リセットCSS - destyle.css



Destyle.css

Opinionated reset stylesheet that provides
a clean slate for styling your html.

Features

- Ensures consistency across browsers as much as possible
- Prevents the necessity of reseting user agent styles
- Prevents style inspector bloat by only targeting what is necessary
- Removes margins & paddings
- Removes default font styles and ensures proper inheritance
- Contributes to the separation of presentation and semantics
- Sets sensible default styles (see docs)
- Well suited for utility class libraries and large codebases
- Made for modern browsers only, therefor small in size (~0.95kb)

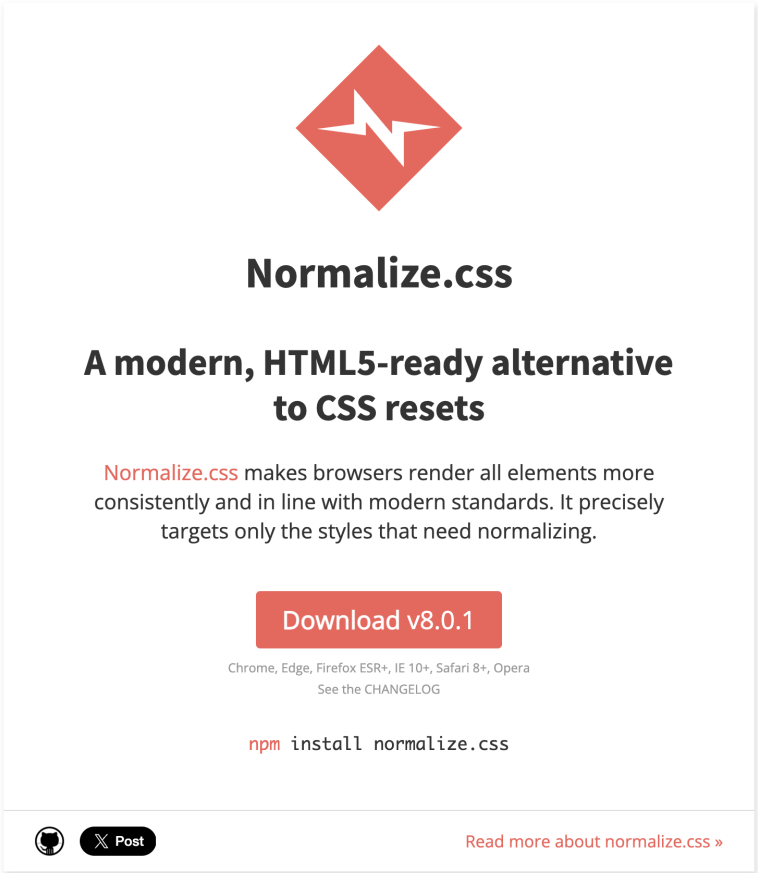
Get it

- `npm install --save destyle.css`
- CDN
- Download

See it in action | View on GitHub

```
<link rel="stylesheet"  href="https://cdn.jsdelivr.net/npm/destyle.css@1.0.15/destyle.css"/>
```

01 ノーマライズCSS - normalize.css



<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/normalize.min.css"/>

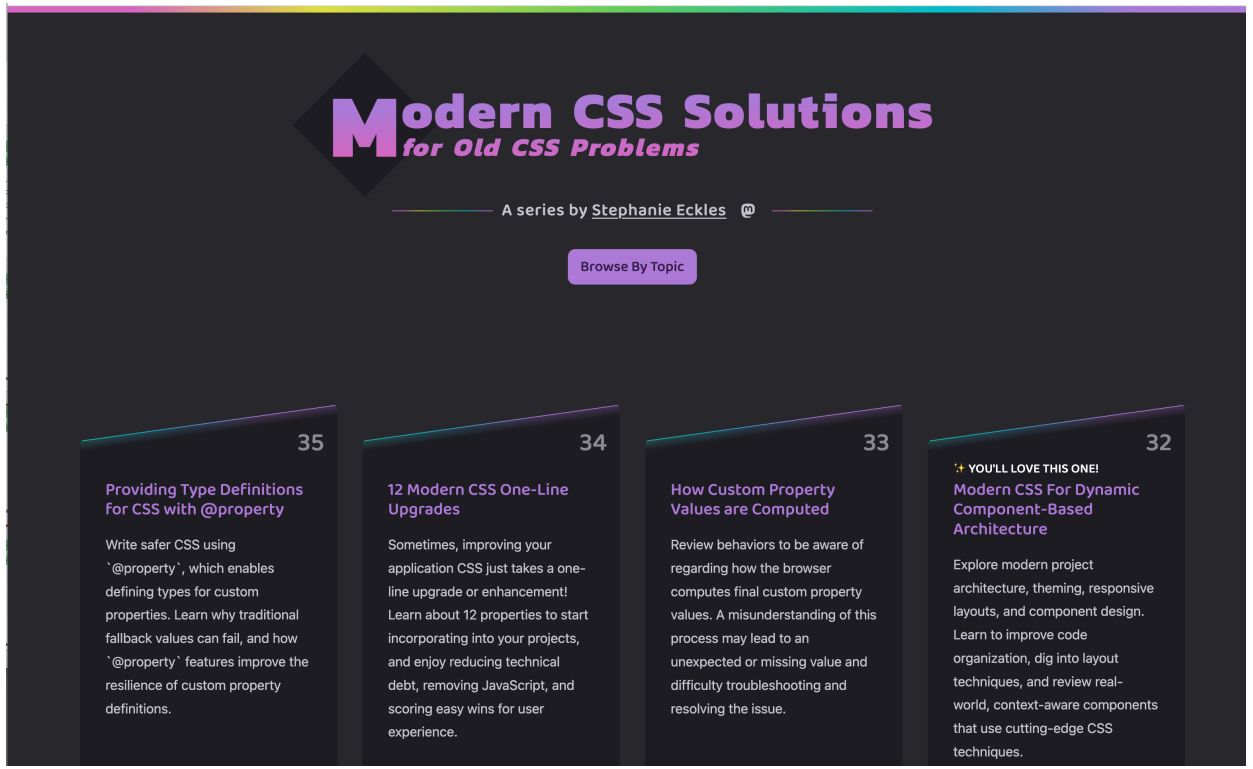
01 リセットCSS,ノーマライズCSS

- ブラウザ間の差異を均一化する

  （フォントなどは別途指定が必要、Safariがゴシック体になるとかではない）

- リストのパディングとマージン
- ボーダーの見た目
- ボタンやマージンのサイズなど

01 リセットCSS,ノーマライズCSS ③modern.css



<link rel="stylesheet” href="https://unpkg.com/modern-css-reset/dist/reset.min.css"/>

02 レイアウトのためのCSS① Flex

- ボックスモデルの復習
- display: flex（Flexbox）を使った横並びのレイアウト
- Flexboxは要素を一列に並べる

02 レイアウトのためのCSS① Flex

HTML

```
<div class="wrap-flex">
<div class="item">Flex item</div>
<div class="item">Flex item</div>
<div class="item">Flex item</div>
</div>
```

CSS

```
.wrap-flex {  display: flex; }
```

02 レイアウトのためのCSS① Flex

HTML

```
<div class="container">
<div class="box"></div>
</div>
```

CSS

```
.container {
border: 2px solid #1b0303;
box-sizing: border-box;
width: 100%;
 height: 300px;
 }
```

```
.box {
background-
color: #f53838;
width: 300px;
height: 200px;
 }
```

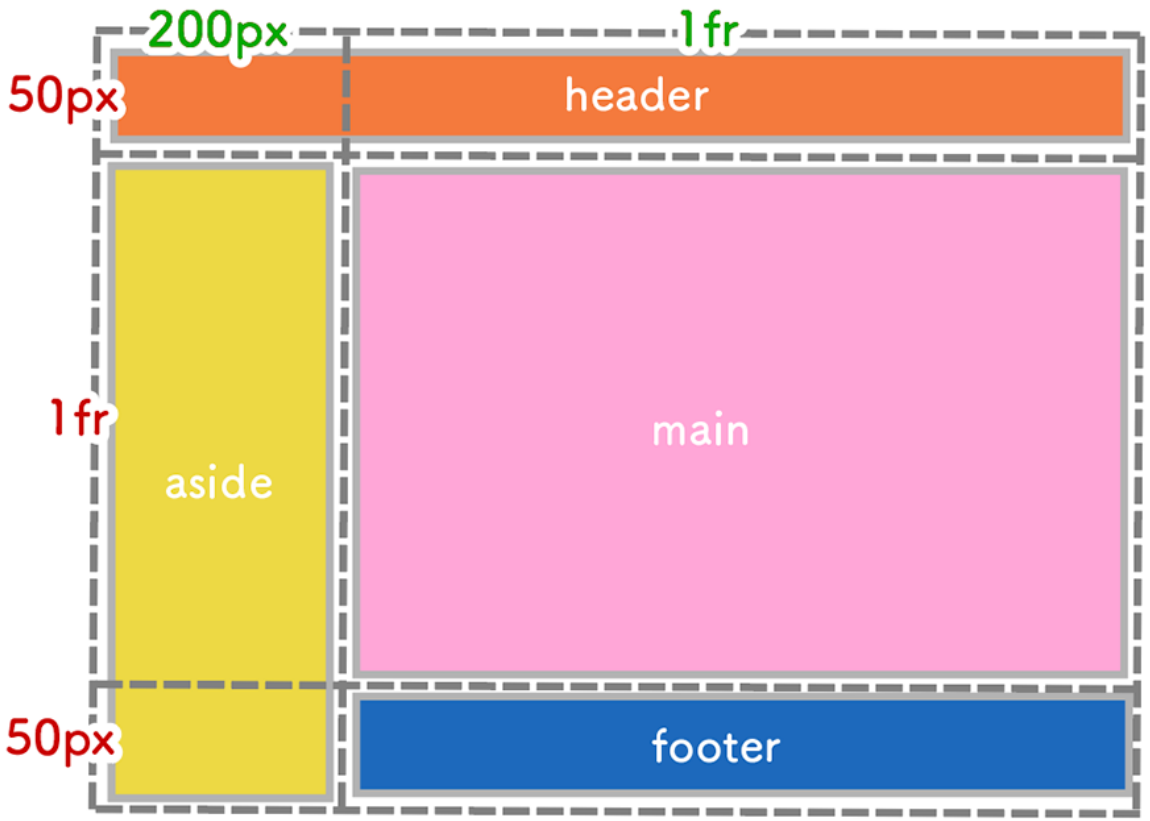
↓追加してみよう

```
 .container {
display: flex;
justify-content: center;
align-items: center;
}
```

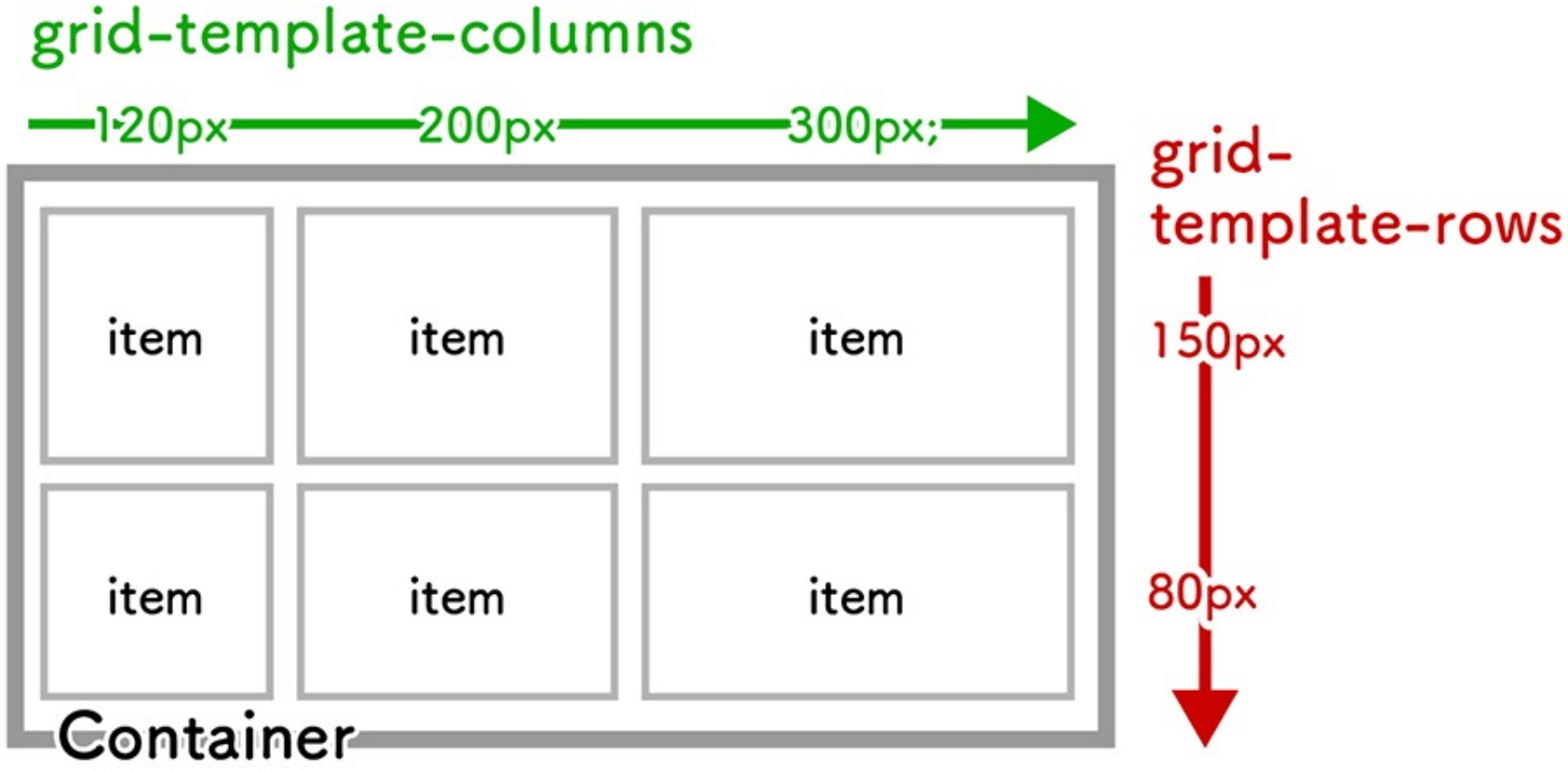
03 レイアウトのためのCSS ② Grid

- display:grid
- Gridでは行と列の格子状にレイアウトを定義する必要がある
- 基本はまず横方向のみでOK ▶️ grid-template-column
- 縦方向に制御したい場合は▶️ grid-template-rows

03 レイアウトのためのCSS ② Grid



03 レイアウトのためのCSS ② Grid



03 レイアウトのためのCSS ② Grid

HTML

```
<div class="wrap-grid">
    <div class="item">Grid item 1</div>
    <div class="item">Grid item 2</div>
    <div class="item">Grid item 3</div>
    <div class="item">Grid item 4</div>
<div class="item">Grid item 5</div>
</div>
```

CSS

```
.wrap-grid {
display: grid;
grid-template-columns: 1fr 1fr;
gap: 10px;
width: 400px; border: 2px solid #00a0e9; padding: 10px; }

.item { /*見た目に関する部分*/
background-color: #e6f7ff; border: 1px solid #00a0e9;  padding: 10px; text-align: center; }
```

03 レイアウトのためのCSS ② Grid

- grid-template-columns: 1fr 1fr; の意味と役割

  Gridコンテナの列（Columns）の数とそれぞれの幅を
  定義するために使われる

  「fr」はGridコンテナ内の利用可能な残りのスペースを分割する比率のこと

  grid-template-columns: 100px auto 1fr; /*3列を定義*/
  1列目　2列目　3列目

03 レイアウトのためのCSS ② Grid

単位	意味	動作	主な用途
fr	残りの利用可能なスペースの比率	コンテナ幅に合わせて均等に幅が変化する（レスポンシブ）	等分割レイアウト（50% / 50% など）
auto	コンテンツの量に応じて幅が決定される	コンテンツの幅を優先的に取り、残りのスペースを分配する	サイズの決まっていないサイドバーやメインコンテンツ

両者＋絶対値（●●pxなど）を組み合わせるのがセオリー

04 Displayプロパティについて

displey:プロパティの基本の値6種類

1. block⇒要素が横までいっぱいに広がり、縦に並んでいく
2. inline⇒要素が平ぺったく横に並んでいく
3. inline-block⇒blockとinlineの中間（ざっくり言うと）
4. none⇒非表示になる
5. flex⇒フレックス化
6. grid⇒グリッド化

04 Displayプロパティについて

displey:blockの特徴

- 縦に積まれていく
- 幅 width と高さ height が指定できる
- デフォルトCSSでは上下に余白ができる
- marginとpaddingを指定できる
- text-align は要素の中身に適応される
- vertical-align は指定できない

04 Displayプロパティについて

displey:inline;の特徴

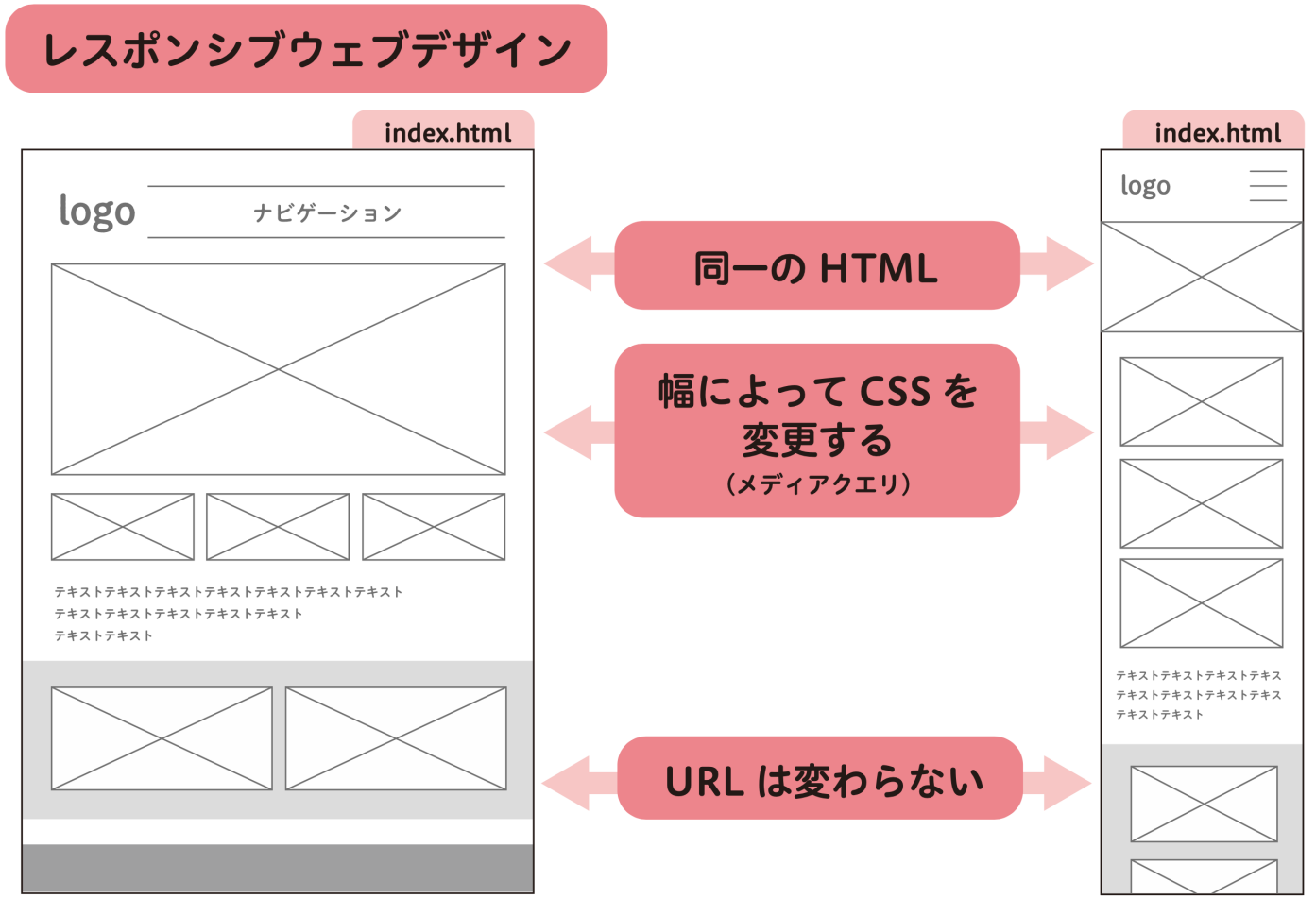
- 横に並んでいく
- 幅 width と高さ height は指定できない。内容物のサイズで大きさが決まる
- 左右だけ margin を指定できる
- 左右に padding を指定できる（実は上下も指定できるけど、前後の行や要素にかぶってしまうので、あまり効果はわからない）
- text-align を親ブロックに付けることで揃えを指定できる
- vertical-align を指定できる

06

マルチデバイス対応について

- レスポンシブ（RWD）
- アダプティブ
- レスポンシブデザインのほうが実装難易度が高い

06 マルチデバイス対応について



07

RWDのための3つの用語

- メディアクエリ
- 2つのピクセル（ デバイスピクセル / CSSピクセル ）
- ビューポート

08 RWDのための3つの用語 - メディアクエリ

横幅を元にデザインを切り替える CSS のメディアクエリ

- **メディアクエリの例（CSS）**

```
body{color:red;}
@media  (max-width: 800px) {
body{color:pink;} }
@media (max-width: 450px) {
body{color:orange;} }
```

(max-width: xxxpx)＝最大幅
もしくは
(min-width: xxxpx)＝最小幅
と記述する

切り替わる数値を「ブレークポイント」と言う

08 RWDのための3つの用語 - メディアクエリ

モバイルファーストの書き方

```
/* 1. まずモバイル（最小幅）のスタイルを記述する */
body { color: orange; }

/* 2. 次にmin-widthを使って、画面を大きくしていく */
@media (min-width: 451px) {
  body { color: pink; } /* タブレットサイズになったら変更 */
}
@media (min-width: 801px) {
  body { color: red; } /* PCサイズになったら変更 */
}
```

08 RWDのための3つの用語 - メディアクエリ

- カンプはどちらから作る？→どちらでもOK

- まとめ

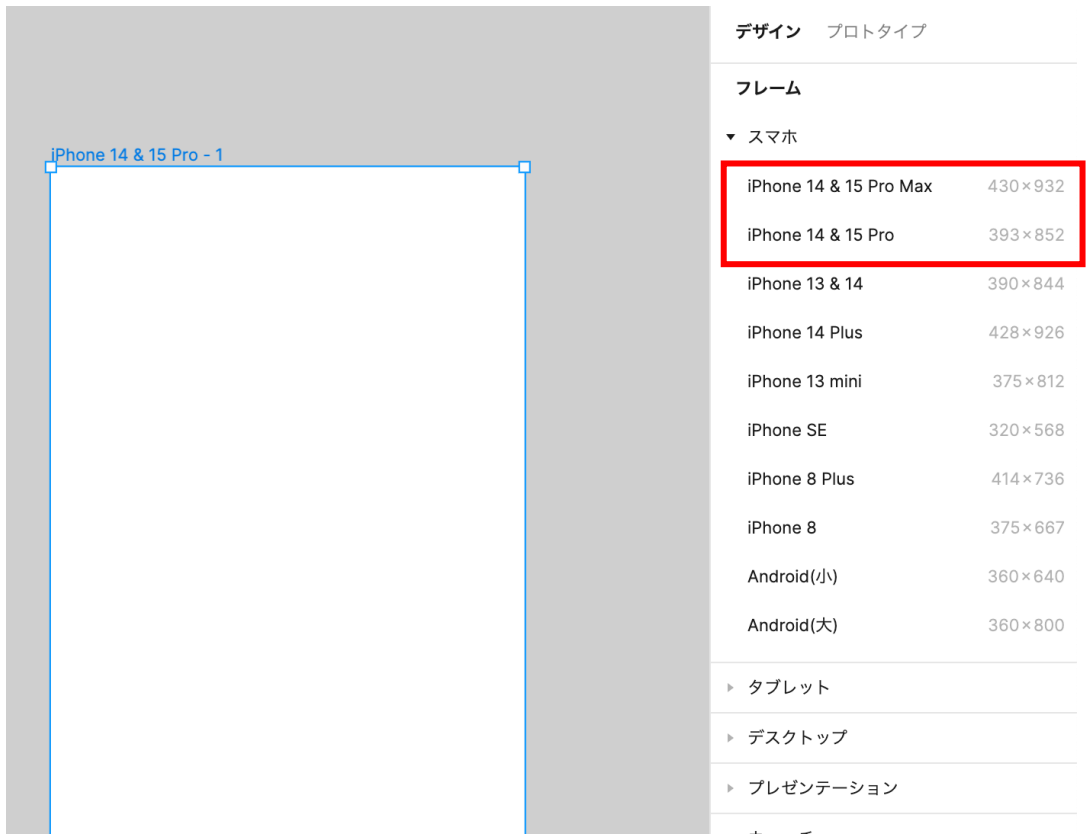
単位	意味	主な用途
max-width	最大幅（この幅以下で適用）	PCファースト（大きい画面から小さい画面へ）で使われる。
min-width	最小幅（この幅以上で適用）	モバイルファースト（小さい画面から大きい画面へ）で使われる。 推奨されるアプローチ。

08 RWDのための3つの用語 - ２つのピクセル

- 画面解像度を示す「デバイスピクセル」
- 同じサイズで表示できる「CSSピクセル」
- デバイスピクセルとCSSピクセルの 比率を示した
  デバイスピクセル比

09

RWDのための3つの用語 - ２つのピクセル



https://www.apple.com/jp/iphone-15-pro/specs/

09 RWDのための3つの用語 - ２つのピクセル

こちら（設定あり）

と

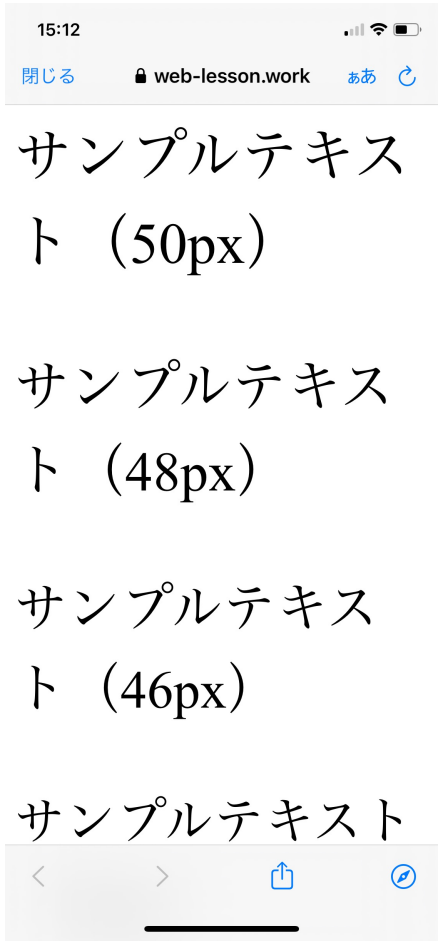
こちら（設定なし）

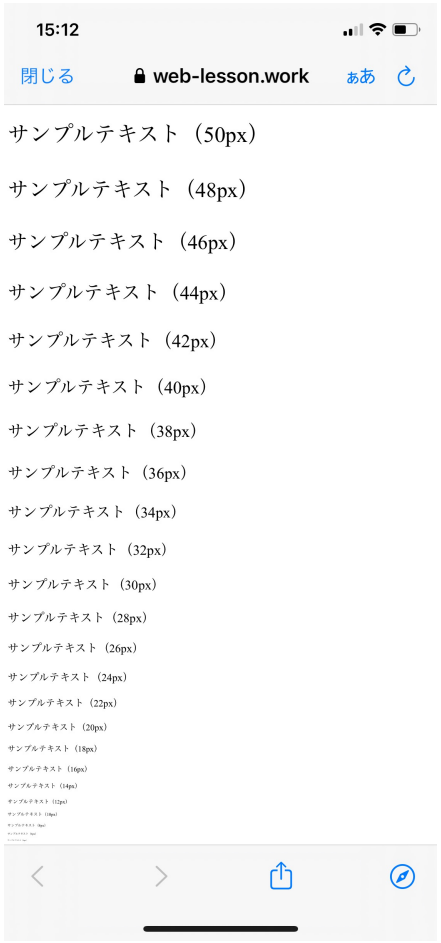
を携帯で見比べてみましょう

設定あり

設定なし

09 RWDのための3つの用語 - ２つのピクセル





表示はデバイスピクセル比で違います

iPhone10で見ています

RWDのための3つの用語 - ビューポート

- 「設定」とは
- HTMLのviewport（ビューポート）指定
- viewportを設定することでデバイスピクセルではなくCSSピクセルでサイトが表示される
  - CSSピクセルで表示させるためのmetaタグの指定

```
<meta name=”viewport” content=”width=device-width, initial-scale=1″>
```

CHAPTER 7

Web制作のワークフロー

サーバー・ドメイン基礎

01 Webの色表現と書き出し形式

- ディスプレイでの表現はRGBを使用する
- ディスプレイによって変わる表示
- カラーコードの種類

- **Hex**

  `#000`

  `#000000`

- **RGB**

  `rgb(0, 0, 0) ;`

- **RGBa**

  `rgb(0, 0, 0,0.5) ;`

  ※ a → アルファ（透明度）

01 Webの色表現と書き出し形式

- PNG,JPGを基本とする
- ファイル名は半角英数字を使用
- SVGのメリットと書き出しの注意点、デモ
- webP, AVIF

01 Webの色表現と書き出し形式

- 書き出す画像には「カラープロファイル」を埋め込む
- sRGB
- Display P3

02 Webデザインの単位

- 文字のサイズ（本文のサイズ）は
  いくつを使用すべきなのか？
  - 標準は16px
  - 10px以下は可読性が損なわれる

    10px：一般的なモニタでの表示サイズでは7～8pt（ 2～3mm ）相当の大きさ

03 「ピクセル」単位の問題点と解決策

- 絶対的な固定値であることのデメリット

  レイアウト全体や文字サイズをユーザーの

  ニーズに合わせて柔軟に調整できない

- レスポンシブ対応の手間（PCでは20px,スマホでは18pxなど）

⬇

そこで（基準のPixelを持ちつつ）別の「相対的な」単位を用いる

03 「ピクセル」単位の問題点と解決策

- 「em（イーエム）」は、親要素のフォントサイズに対する相対単位 1emは親要素のフォントサイズと等しくなる

- 親の親の…と連鎖するとかなり複雑になる

⬇

これを解決するのが、「rem（レム）」

03 「ピクセル」単位の問題点と解決策

- 「rem（レム）」は、ルート要素（<html>タグに指定されているフォントサイズ）が基準になる
- 基準が一定なので計算が簡単
- レスポンシブとアクセシビリティに配慮

03 「ピクセル」単位の問題点と解決策

- 1remが10pixelになるようにするテクニックがよく使われる

```
html { font-size: 62.5%; }
```

1.6rem：今は16pxの値だが、環境に応じて柔軟に変化する
　　16px相当の値

16px　：16 pxから動かない値

04 画面や要素の幅を示す単位

- vh と vw は、ブラウザのビューポート（画面表示領域）のサイズを基準とする相対単位
- width: 50vw; と指定すると、要素の幅は画面幅のちょうど半分（50%）になる
- 100% は**親要素のサイズ**を基準とするのに対し、vh/vw などの単位は、ブラウザの表示領域全体（ビューポート）が基準となる

04 画面や要素の幅を示す単位

単位	名称	意味	主な用途
svh / svw	Small Viewport	最小の安定したビューポートサイズ（URLバーが表示されている状態）を基準とする。	スクロール中にアドレスバーが消えても、レイアウトが崩れないようにしたい場合に使う。
lvh / lvw		最大のビューポートサイズ（URLバーが非表示の状態）を基準とする。	画面全体を最大限利用したい場合に使う。
dvh / dvw	Dynamic Viewport	スクロールに応じて動的に変化する実際のビューポートサイズを基準とする。	常に画面のサイズにぴったり合わせたい場合に使う。

04 画面や要素の幅を示す単位

```
<div class="key-visual-container">
    <img src="your-key-visual.jpg" alt="キービジュアル">
</div>
```

```
.key-visual-container {
        height: 100dvh;
        width: 100vw; /* 幅は画面幅全体に固定 */
        overflow: hidden; /* はみ出た画像部分を隠す */
}

 .key-visual-container img {
        width: 100%; height: 100%; /* 画像のアスペクト比を保ちつつ、コンテナ全体を覆うように拡大 */
        object-fit: cover;
}
```

05 タイポグラフィのこだわりはどこまで？

	指定箇所	CSS 例
行間	Web の技術でできる タイポグラフィ	line-height:
文字間	Web の技術でできる タイポグラフィ	letter-spacing:
文字サイズ	Web の技術でできる タイポグラフィ	font-size

05 タイポグラフィのこだわりはどこまで？

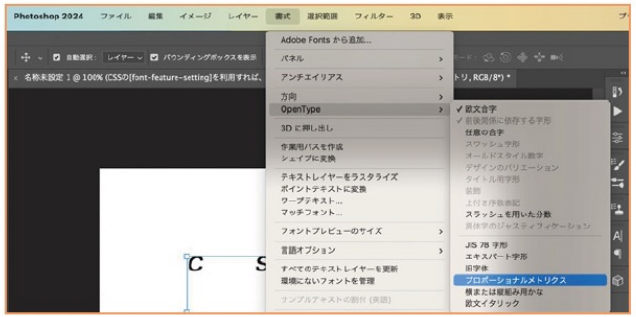
OpenTypeフォントにfont-feature-settingsを適用する

CSSで自動カーニングを有効化した場合の表示

【自動カーニング有効】CSSの[font-feature-setting]を利用すれば、OpenTypeフォントならWebでもプロポーショナルメトリクスを有効化できる。

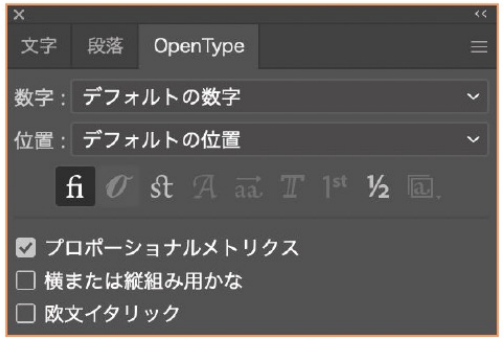
【自動カーニング無効】CSSの[font-feature-setting]を利用すれば、OpenTypeフォントならWebでもプロポーショナルメトリクスを有効化できる。

Photoshopのプロポーショナルメトリクスの設定



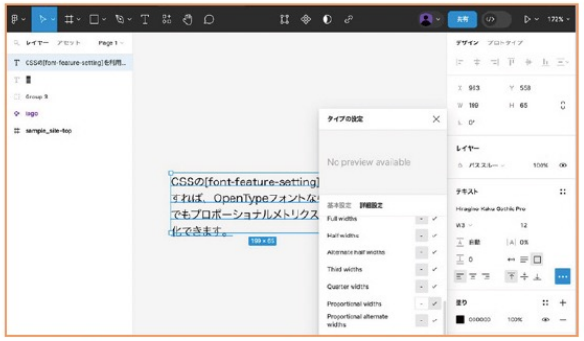
[書式]メニュー→[OpernType]→[プロポーショナルメトリクス]

Illustratorのプロポーショナルメトリクスの設定



[OpernType]パネルから設定

Figmaのプロポーショナルメトリクスの設定



[タイプの設定]→[詳細設定]→[Proportional widths]

06 Webにおける明朝体

- Webサイトに明朝体が少ない理由
- 明朝体は使えないのか
- 余談：縦書きは使えないのか？
  - CSSのwriting-modeプロパティ

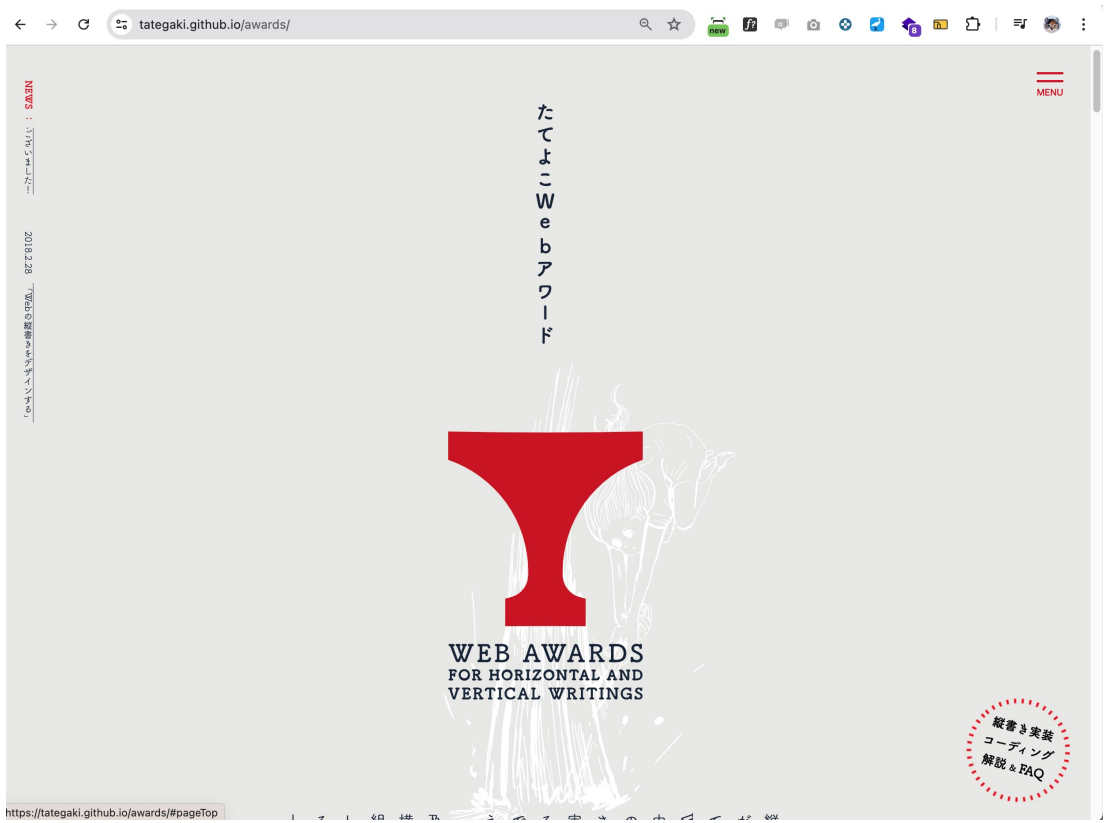
```
h1 {
writing-mode: vertical-rl;
}
```

06 Webにおける明朝体

参考資料：https://www.adobe.com/jp/creativecloud/design/discover/web-fonts.html

06

Webにおける明朝体



参考資料：たてよこWebアワード　https://tategaki.github.io/awards/

07 Webサイトのフォントは環境で変化する

- デバイスフォントを指定する
- Webフォントを指定する
- どちらもCSSのfont-family:プロパティで指定する
  - CSSのfont-familyプロパティ

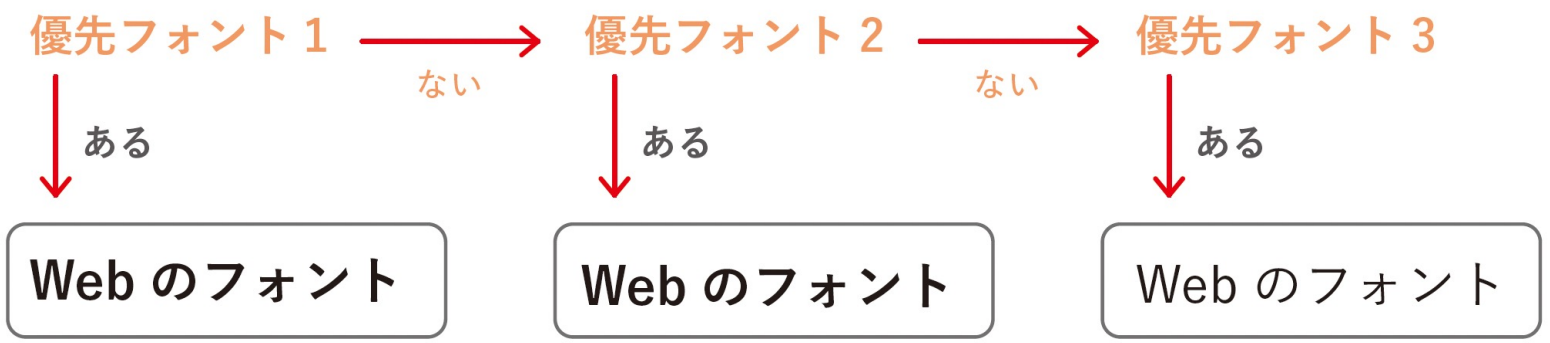
```
body {
font-family: "MS Pゴシック" ;
}
```

07 Webサイトのフォントは環境で変化する

デバイスフォントの表示の仕組み

優先フォント１　優先フォント２　優先フォント３

```
font-family: "ヒラギノ角ゴ ProN", "遊ゴシック", sans-serif;
```



07 Webサイトのフォントは環境で変化する

デバイスフォントの表示の仕組み

```
font-family: "Futura",  "ヒラギノ角ゴ ProN";
```

優先フォント１　　優先フォント２

08 特定の書体を使いたいときはWebフォントを利用する

- Webフォントを使わない場合、OSに無いフォントは表示されない

OS	デフォルトで入っている書体の例	フォントを確認できるURL
macOS	ヒラギノ角ゴシック、ヒラギノ明朝、游ゴシック、游明朝	https://support.apple.com/ja-jp/103197
Windows	メイリオ、游ゴシック、游明朝	https://learn.microsoft.com/en-us/typography/fonts/windows_11_font_list
iOS	ヒラギノ角ゴシック、ヒラギノ明朝	https://developer.apple.com/fonts/system-fonts/
Android 6.0以降	Noto Sans CJK	なし

08 特定の書体を使いたいときはWebフォントを利用する

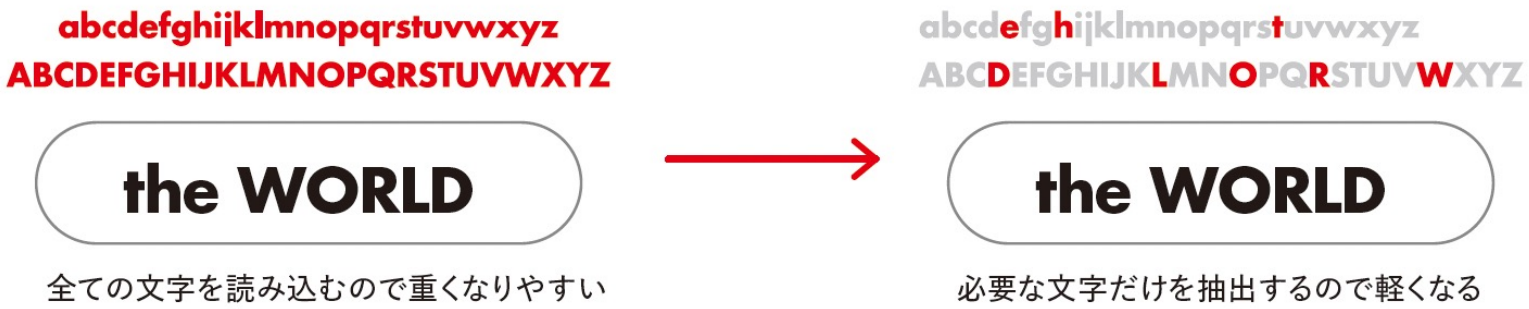
特定のフォントを使いたいときはWebフォントを利用する

- Webサイトのデータなどと同様にネットワーク上にフォントデータを設置し、そこにアクセスすることで様々な環境でも同じフォントが使用できる仕組み

- Google Fonts
- Adobe Fonts
- FONT PLUS
- モリサワ TypeSquare

08 特定の書体を使いたいときはWebフォントを利用する

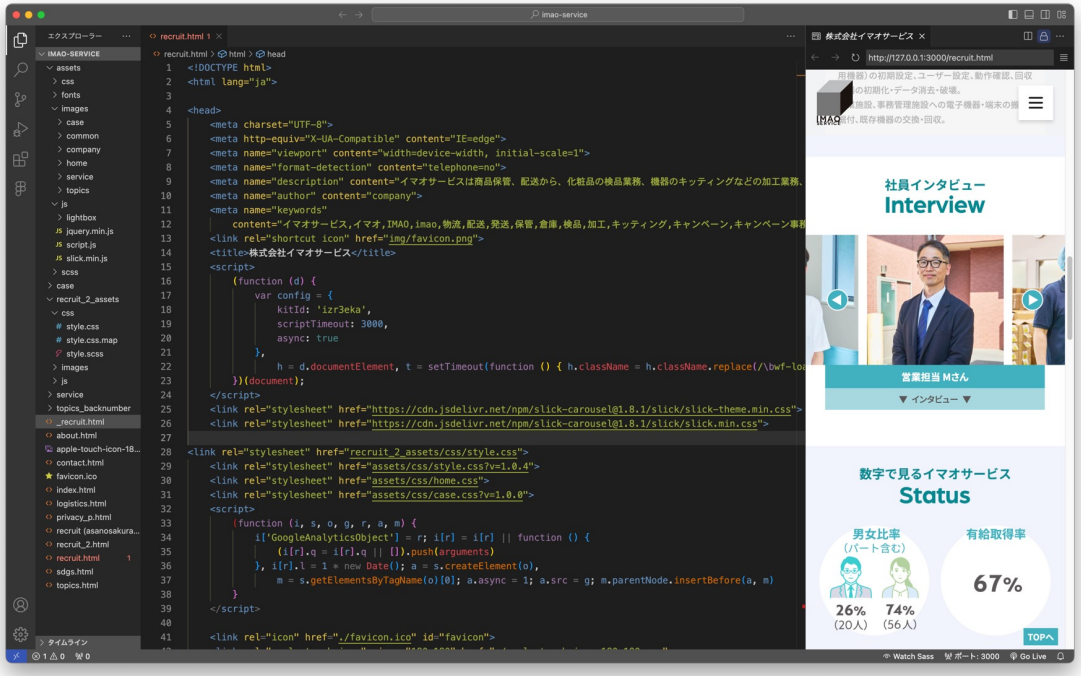
- フォントデータを自社のサーバーに格納する方式とCDNで利用する方式がある
- 必要な文字だけを抽出するサブセット化

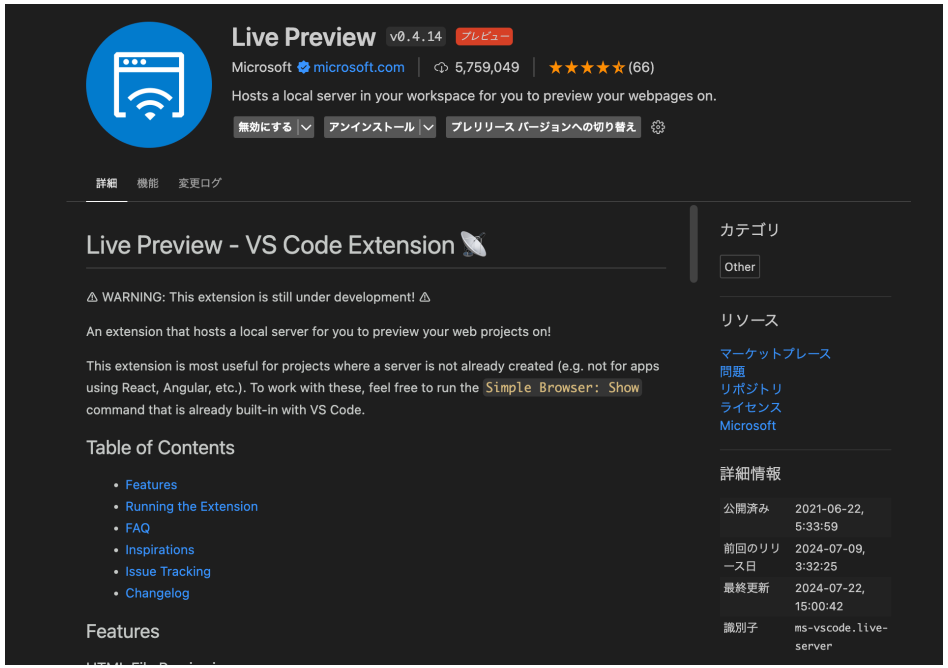


JavaScriptとPHP

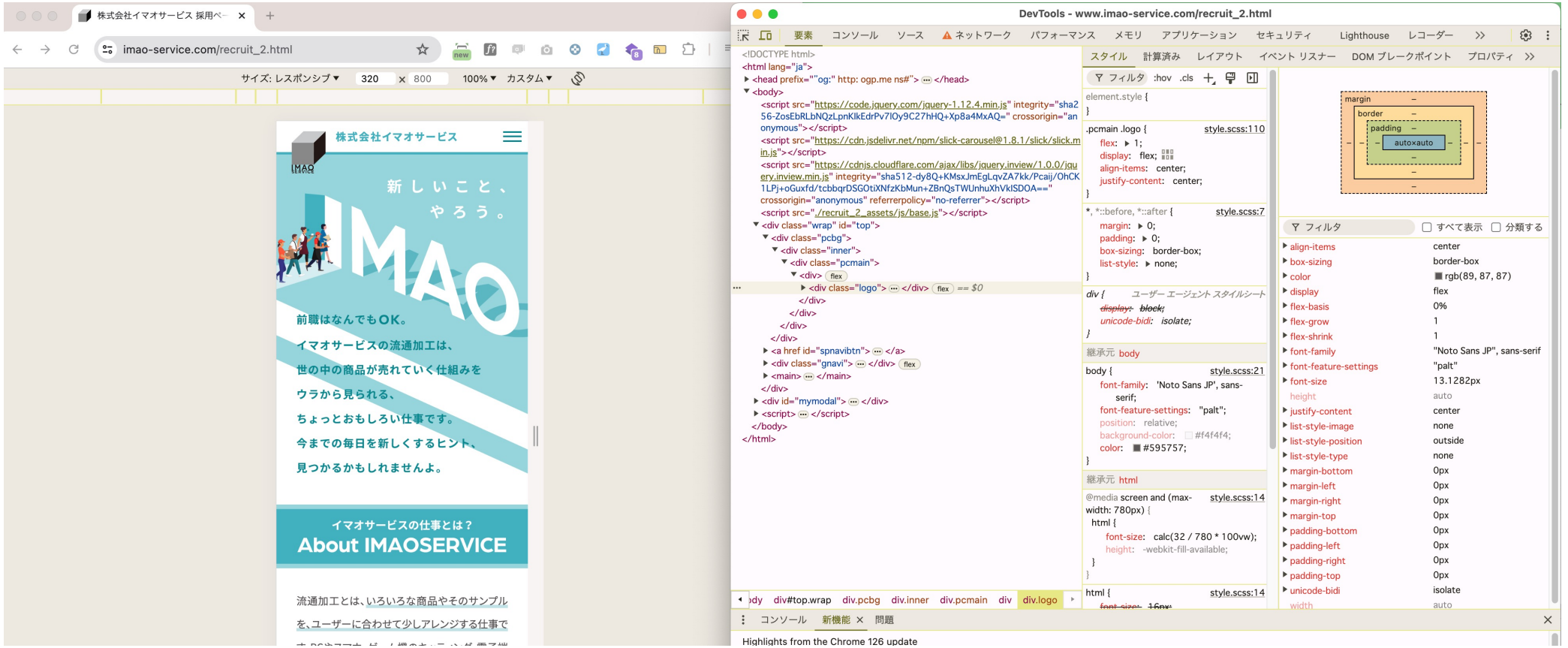
- PHPを使った共通要素のインクルード（読み込み）
  1. headerを用意して、headerだけをheader.phpにする
  2. index.htmlをindex.phpにする
  3. Index,php内に<?php include ‘common/header.php’; ?>等と記述

11 検証（確認）する


imao-service
recruit.html 1
株式会社イマオサービス
http://127.0.0.1:3000/recruit.html
社員インタビュー
Interview
営業担当 Mさん
▼ インタビュー ▼
数字で見るイマオサービス
Status
男女比率
（パート含む）
26%
（20人）
74%
（56人）
有給取得率
67%
TOPへ


Live Preview
v0.4.14
プレビュー
Microsoft
microsoft.com
5,759,049
(66)
Hosts a local server in your workspace for you to preview your webpages on.
無効にする
アンインストール
プレリリース バージョンへの切り替え
詳細
機能
変更ログ
Live Preview - VS Code Extension
⚠ WARNING: This extension is still under development! ⚠
An extension that hosts a local server for you to preview your web projects on!
This extension is most useful for projects where a server is not already created (e.g. not for apps using React, Angular, etc.). To work with these, feel free to run the Simple Browser: Show command that is already built-in with VS Code.
Table of Contents
Features
Running the Extension
FAQ
Inspirations
Issue Tracking
Changelog
Features
カテゴリ
Other
リソース
マーケットプレース
問題
リポジトリ
ライセンス
Microsoft
詳細情報
公開済み
2021-06-22, 5:33:59
前回のリリース日
2024-07-09, 3:32:25
最終更新
2024-07-22, 15:00:42
識別子
ms-vscode.live-server

11 検証（確認）する


株式会社イマオサービス 採用ペ
imao-service.com/recruit_2.html
サイズ: レスポンシブ
320
800
100%
カスタム
株式会社イマオサービス
新しいこと、
やろう。
IMAO
前職はなんでもOK。
イマオサービスの流通加工は、
世の中の商品が売れていく仕組みを
ウラから見られる、
ちょっとおもしろい仕事です。
今までの毎日を新しくするヒント、
見つかるかもしれませんよ。
イマオサービスの仕事とは？
About IMAOSERVICE
流通加工とは、いろいろな商品やそのサンプルを、ユーザーに合わせて少しアレンジする仕事で
DevTools - www.imao-service.com/recruit_2.html
要素
コンソール
ソース
ネットワーク
パフォーマンス
メモリ
アプリケーション
セキュリティ
Lighthouse
レコーダー
スタイル
計算済み
レイアウト
イベント リスナー
DOM ブレークポイント
プロパティ
<!DOCTYPE html>
<html lang="ja">
<head prefix="" og:" http: ogp.me ns#"> </head>
<body>
<script src="https://code.jquery.com/jquery-1.12.4.min.js" integrity="sha256-ZosEbRLbNQzLpnKIkEdrPv7lOy9C27hHQ+Xp8a4MxAQ=" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/slick-carousel@1.8.1/slick/slick.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery.inview/1.0.0/jquery.inview.min.js" integrity="sha512-dy8Q+KMsxJmEgLqvZA7kk/Pcaij/OhCK1LPj+oGuxfd/tcbbqrDSGOtiXNfzKbMun+ZBnQsTWUnhuXhVkISDOA==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>
<script src="./recruit_2_assets/js/base.js"></script>

 flex
  flex == $0

<a href id="spnavibtn">
  flex
<main> </main>

<script> </script>
</body>
</html>
element.style {
}
.pcmain .logo { style.scss:110
flex: ▶ 1;
display: flex;
align-items: center;
justify-content: center;
}
*, *::before, *::after { style.scss:7
margin: ▶ 0;
padding: ▶ 0;
box-sizing: border-box;
list-style: ▶ none;
}
div { ユーザー エージェント スタイルシート
display: block;
unicode-bidi: isolate;
}
継承元 body
body { style.scss:21
font-family: 'Noto Sans JP', sans-serif;
font-feature-settings: "palt";
position: relative;
background-color: #f4f4f4;
color: #595757;
}
継承元 html
@media screen and (max-width: 780px) { style.scss:14
html {
font-size: calc(32 / 780 * 100vw);
height: -webkit-fill-available;
}
}
html { style.scss:14
margin
border
padding
auto×auto
フィルタ
すべて表示
分類する
align-items center
box-sizing border-box
color rgb(89, 87, 87)
display flex
flex-basis 0%
flex-grow 1
flex-shrink 1
font-family "Noto Sans JP", sans-serif
font-feature-settings "palt"
font-size 13.1282px
height auto
justify-content center
list-style-image none
list-style-position outside
list-style-type none
margin-bottom 0px
margin-left 0px
margin-right 0px
margin-top 0px
padding-bottom 0px
padding-left 0px
padding-right 0px
padding-top 0px
unicode-bidi isolate
width auto
div#top.wrap
div.pcbg
div.inner
div.pcmain
div
div.logo
コンソール
新機能
問題
Highlights from the Chrome 126 update

アップロードして公開する

- サーバーが必要

  「実際のサイト」の他に「検証環境」があると良い

- FTPアプリでのアップロード
- SSH（Secure Shell）を使ったアップロード

ドメインとサーバー

- レンタルサーバーから提供されるアドレス（URL）とは別に独自のアドレス＝ドメインを紐づける
- この作業を指してDNSの設定という
- 小規模なサイト向けのレンタルサーバーであれば高度な知識は不要

Other CHAPTER

コーディングに困るデザインデータ

01 背景として実装するなら「元素材」が必要

- 画像を使った背景プロパティでは「元の素材」を一緒に渡す
- 繰り返しのパターンをbackgroundプロパティで実装する場合
- リキッド画像をbackgroundプロパティで実装する場合

02 不揃いなグラデーションはコードを煩雑にする

- 「ちょっと違うグラデーション」を作らない
- Figmaのスタイル機能、バリアブル機能を活用する

03 本文に「字切り」の改行は入れない

- Webの「字切り」は万人の環境で同一の表示ができない

デザイナーの意図する改行

Web サイトの特性としては、閲覧者側
の要因によってデバイステキスト部分の
フォントサイズが変化します。

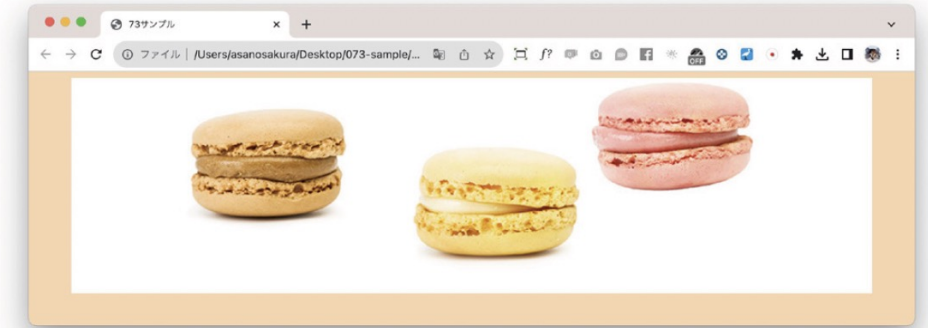
環境によって変化する場合も

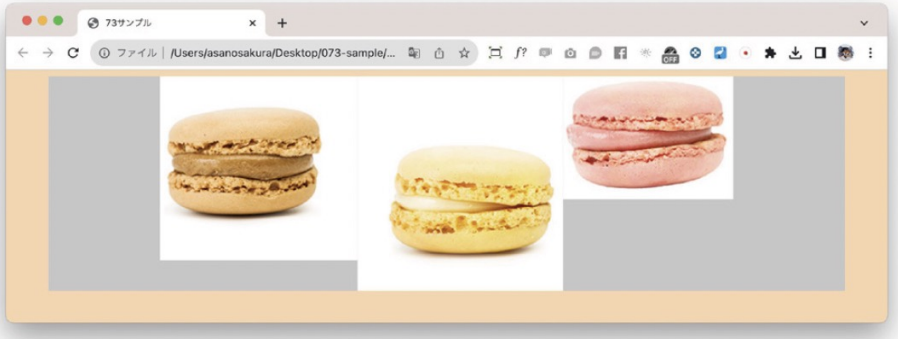
Web サイトの特性としては、閲覧者側
の要因によってデバイステキスト部分
の
フォントサイズが変化します。

- 実務におけるレスポンシブウェブデザインでの改行対応
- 字切りをしたい場合とその対応

04 謎の余白を作らない

- 「CSSが苦手」で画像に余白を持たせるとCSSが複雑になる





05 必要か不必要かがわからないデータは存在させない

- デザインの案？状態の変化？明確に指示を入れる
- Figmaのページ機能を活用してもOK
- 元の素材データの管理はしっかりと。シェアは場合により

06 更新でデザインがどう変化するかを想定する

- リリース後のコンテンツの増減によって崩れやすい見た目
- 文字数の問題、数の問題

07 「動かしたい、でもどう動くかわからない」はNG！

- インタラクションをつける理由と注意点
  - インタラクションはユーザに行動を喚起させる
- 動かすことのデメリットも考慮する
- インタラクションの伝え方
  - 参考例を見せる / アプリで共有する / 完全にお任せする