CSS Nite LP54

Coder's High 2017

2017.11.4

CSSを
ちゃんと
書くためには?

CSS Nite LP54「Coder's High 2017」

まぼろし
MABOROSHI Co., Ltd.

久保 知己 ・ 伊藤 由暁

まぼろし
MABOROSHI Co., Ltd.

久保 知己

CSS: 9年目 / Sass: 5年目

@kojika17

伊藤 由暁

CSS: 10年目 / Sass: 4年目

@o_ti

まぼろし
MABOROSHI Co., Ltd.

ペペロンチーノ

15年以上

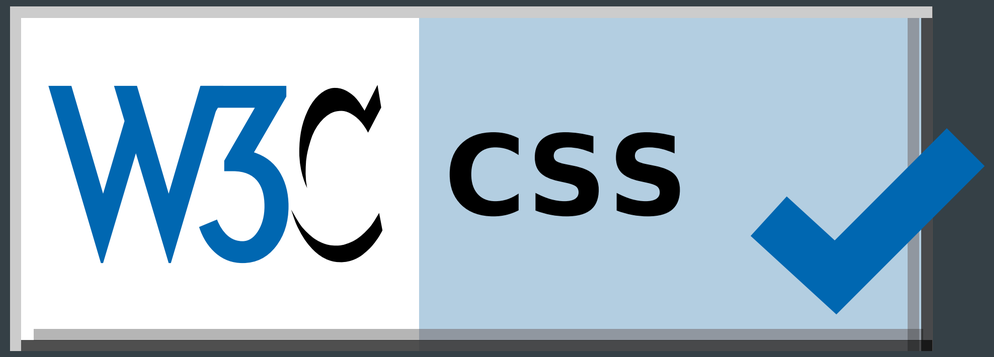
kojika17

たまごかけごはん

20年以上

otiext

CSSをちゃんと書く



CSS

webサイトの
デザインの壊しかた

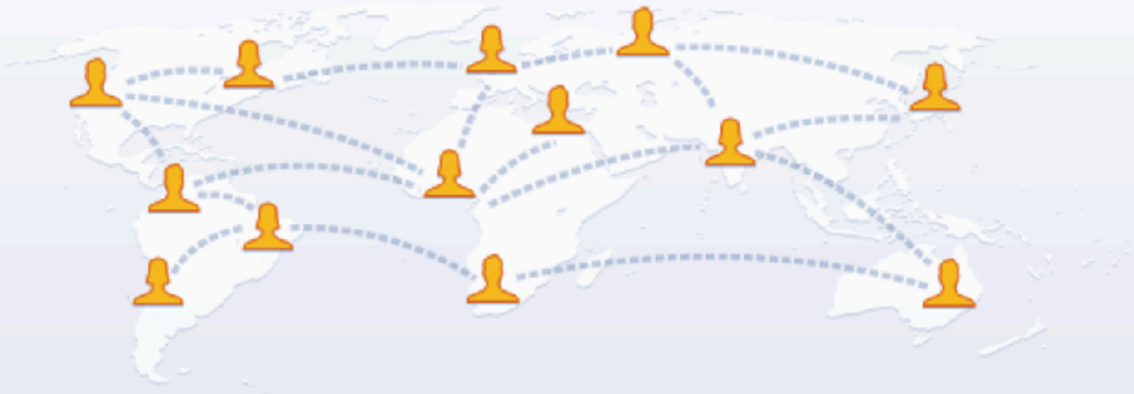
facebook

メールアドレスまたは携帯番号　パスワード

ログイン

アカウントを忘れた場合

Facebookを使うと、友達や同僚、同級生、仲間たちとつながりを深められます。ケータイ、スマートフォンからもアクセスできます。



アカウント登録

情報のプライバシーは設定で管理できるので安心です。

姓　名

携帯番号またはメールアドレス

パスワード

誕生日

1999 9月 16　生年月日を入力していただく理由

女性　男性

[アカウントを作成]をクリックすることで、利用規約に同意し、Cookieポリシーに関する情報を含むデータに関するポリシーを読んだものとします。サービスに関連してFacebookからSMS通知が届くことがありますが、これはいつでもオプトアウトできます。

アカウントを作成

https://www.facebook.com/

セレクタ

```
div {

}
```

プロパティ

```
div {
   width: ;
}
```

値

```
div {
  width: 100%;
}
```

宣言

```
div {
  width: 100%;
}
```

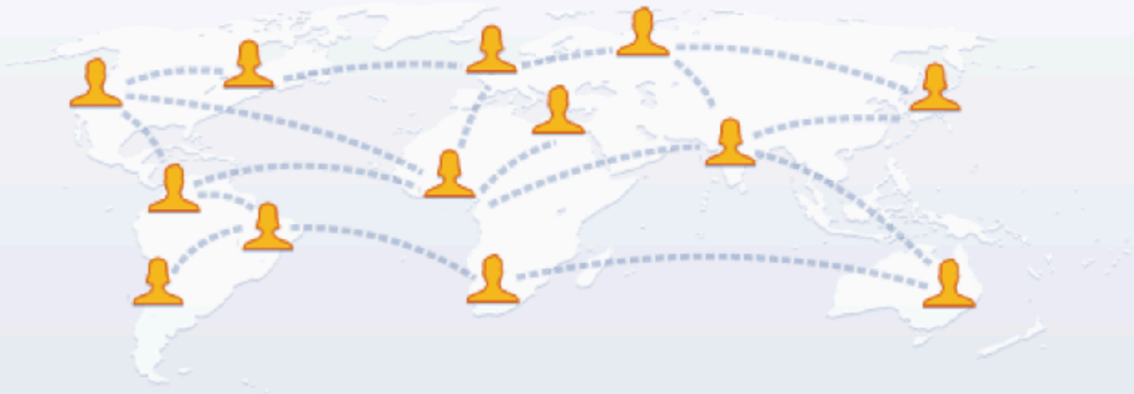
facebook

メールアドレスまたは携帯番号　パスワード

ログイン

アカウントを忘れた場合

Facebookを使うと、友達や同僚、同級生、仲間たちとつながりを深められます。ケータイ、スマートフォンからもアクセスできます。



アカウント登録

情報のプライバシーは設定で管理できるので安心です。

姓　名

携帯番号またはメールアドレス

パスワード

誕生日

1999 9月 16　生年月日を入力していただく理由

女性　男性

[アカウントを作成]をクリックすることで、利用規約に同意し、Cookieポリシーに関する情報を含むデータに関するポリシーを読んだものとします。サービスに関連してFacebookからSMS通知が届くことがありますが、これはいつでもオプトアウトできます。

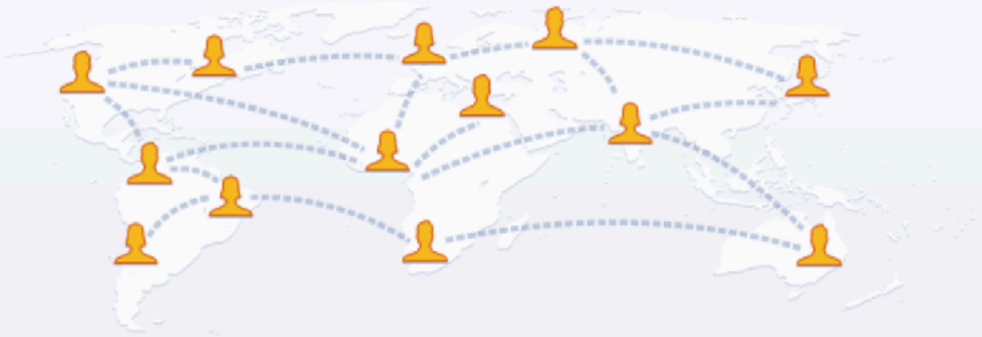
アカウントを作成

https://www.facebook.com/

facebook

メールアドレスまたは携帯番号　パスワード

Facebookを使うと、友達や同僚、同級生、仲間たちとつながりを深められます。ケータイ、スマートフォンからもアクセスできます。



アカウント登録

情報のプライバシーは設定で管理できるので安心です。

姓

名

携帯番号またはメールアドレス

https://www.facebook.com/

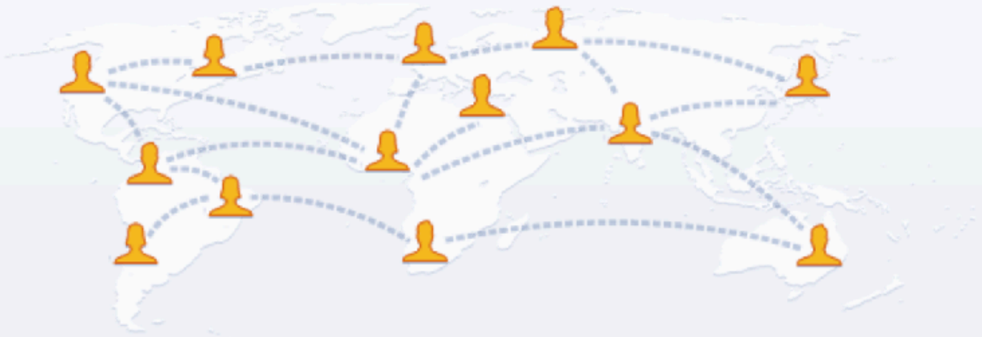
宣言に !important 規則を追加すれば完璧

```
div {
  width: 100% !important;
}
```

facebook

メールアドレスまたは携帯番号　パスワード

Facebookを使うと、友達や同僚、同級生、仲間たちとつながりを深められます。ケータイ、スマートフォンからもアクセスできます。



アカウント登録

情報のプライバシーは設定で管理できるので安心です。

姓

名

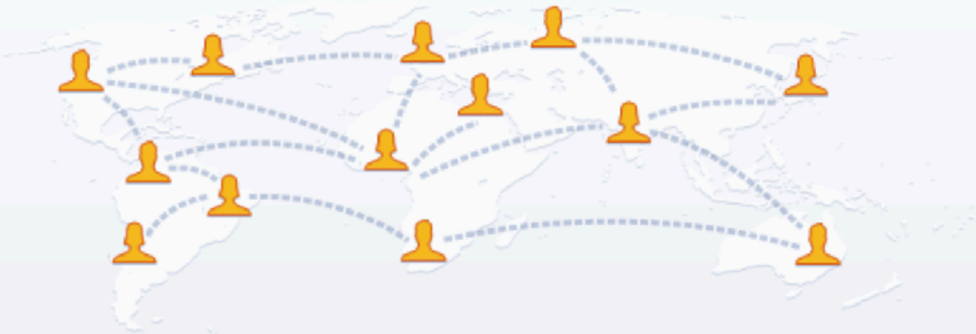
携帯番号またはメールアドレス

https://www.facebook.com/

facebook

メールアドレスまたは携帯番号　パスワード

Facebookを使うと、友達や同僚、同級生、仲間たちとつながりを深められます。ケータイ、スマートフォンからもアクセスできます。



アカウント登録

情報のプライバシーは設定で管理できるので安心です。

姓

名

携帯番号またはメールアドレス

パスワード

https://www.facebook.com/

丁寧に作ったCSS設計も
宣言(プロパティと値)で簡単に壊せる

知らず知らずに宣言で
CSSを壊している可能性も？

本当にあった怖いCSS

```
a {
  color: #000 !important;
}
```

意図せずにCSSを壊すことも？

・宣言をとにかく詰め込む

・宣言の選び方が下手

CSSを理解しよう

CSSの仕様

- レイアウトモード
- 視覚整形モデル
- ボックスモデル
- CSS 構文
- @-規則
- コメント
- 優先度
- 初期値
- 指定値
- 継承
- 算出値
- 使用値
- 実効値
- 解決値
- 置換要素
- 値定義構文
- 短縮プロパティ
- マージンの相殺

CSSの仕様

- レイアウトモード
- 視覚整形モデル
- ボックスモデル
- CSS 構文
- @-規則
- コメント
- 優先度
- 初期値
- 指定値
- 継承
- 算出値
- 使用値
- 実効値
- 解決値
- 置換要素
- 値定義構文
- 短縮プロパティ
- マージンの相殺

視覚整形モデル

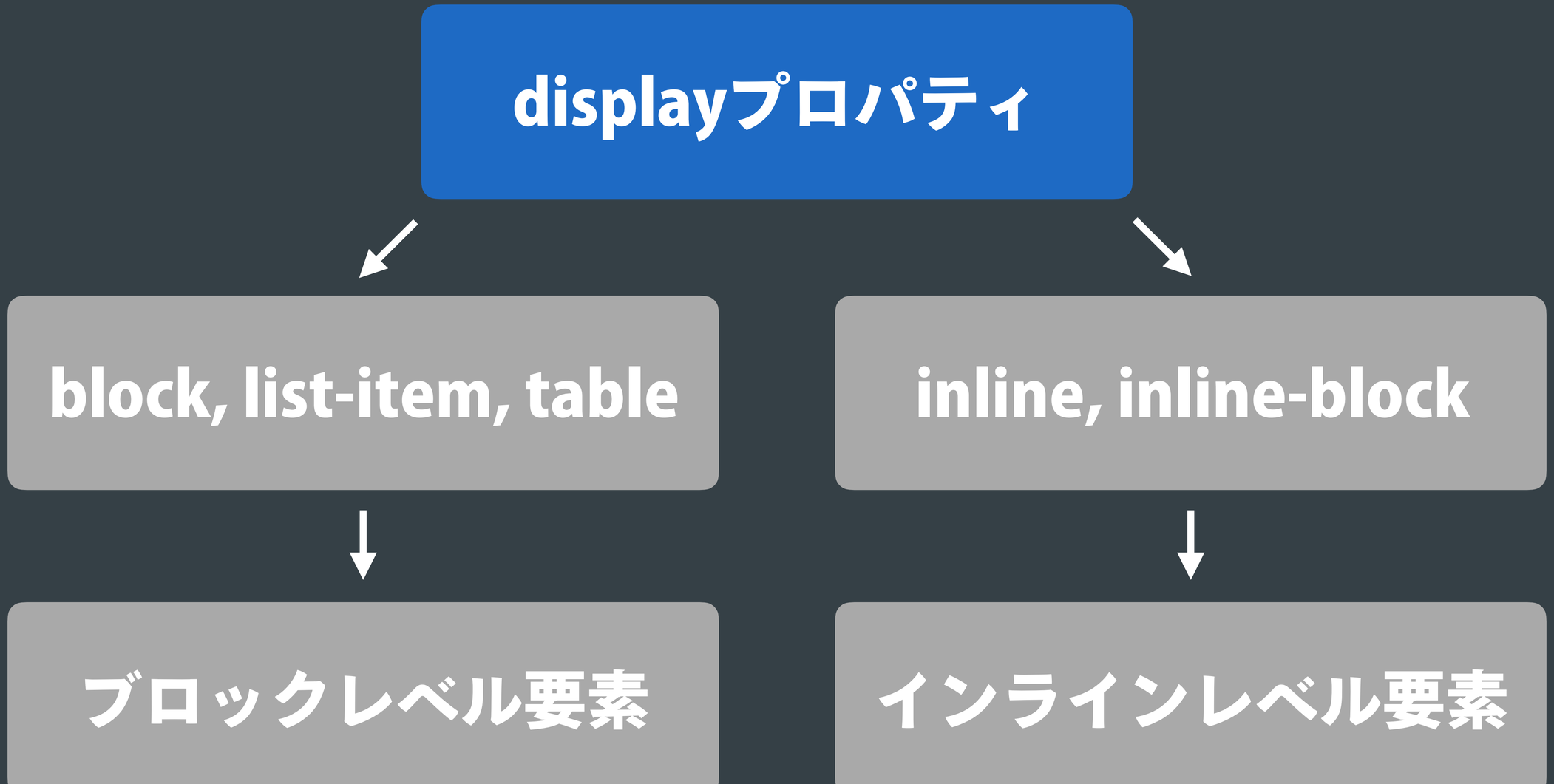
視覚整形モデル

- 要素の種類
- ボックスの生成
- 位置決定スキーム
  (通常フロー、フロート、絶対位置指定)

ボックスの生成

視覚整形モデル

視覚整形モデル: ボックスの生成



視覚整形モデル: ボックスの生成

block, list-item, table

↓

ブロックレベル要素

↓

縦に積まれる

inline, inline-block

↓

インラインレベル要素

↓

行内に配置

視覚整形モデル: ボックスの生成

ブロックレベル要素

↓

縦に積まれる

インラインレベル要素

↓

行内に配置

Point

言葉が似てるけど、全く異なる

HTML 4.01, XHTML

- ブロック要素
- インライン要素

CSS

- ブロックレベル要素
- インラインレベル要素

視覚整形モデル: ボックスの生成

ブロックレベル要素

↓

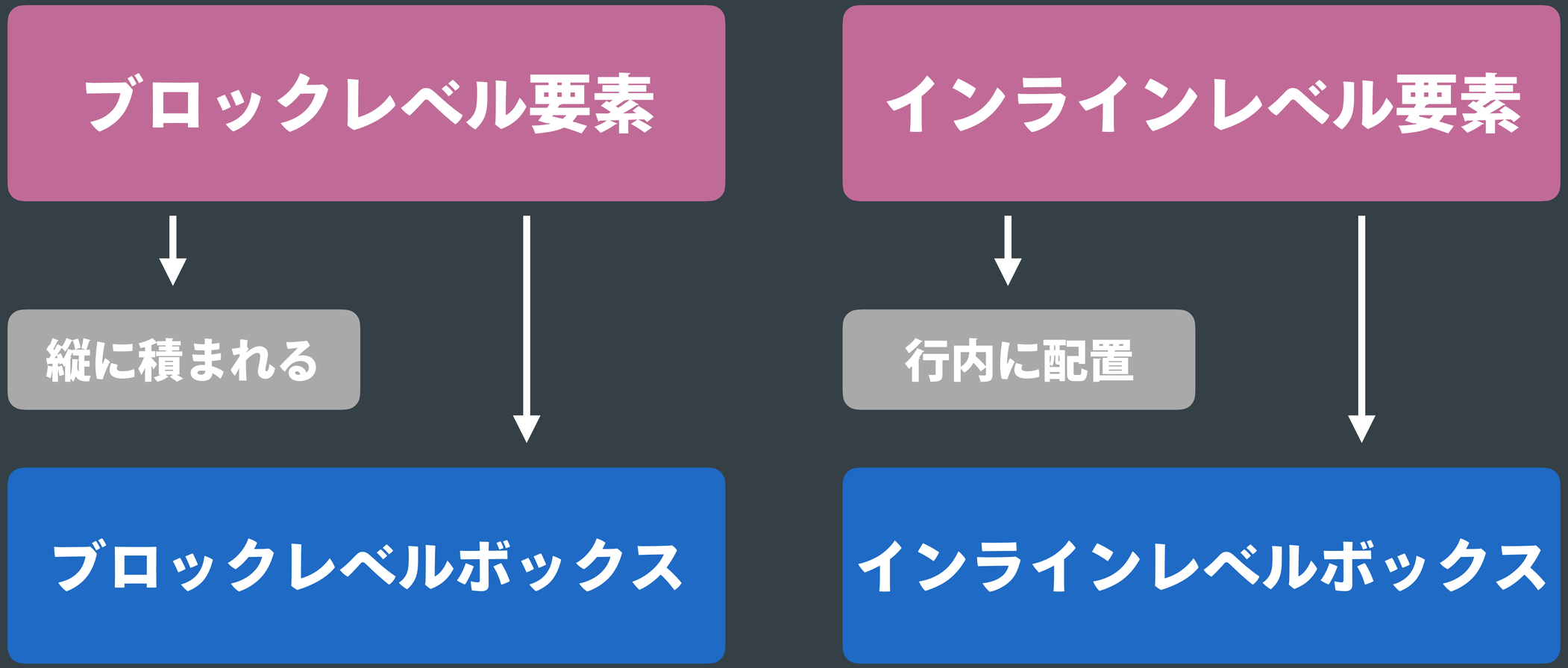
縦に積まれる

インラインレベル要素

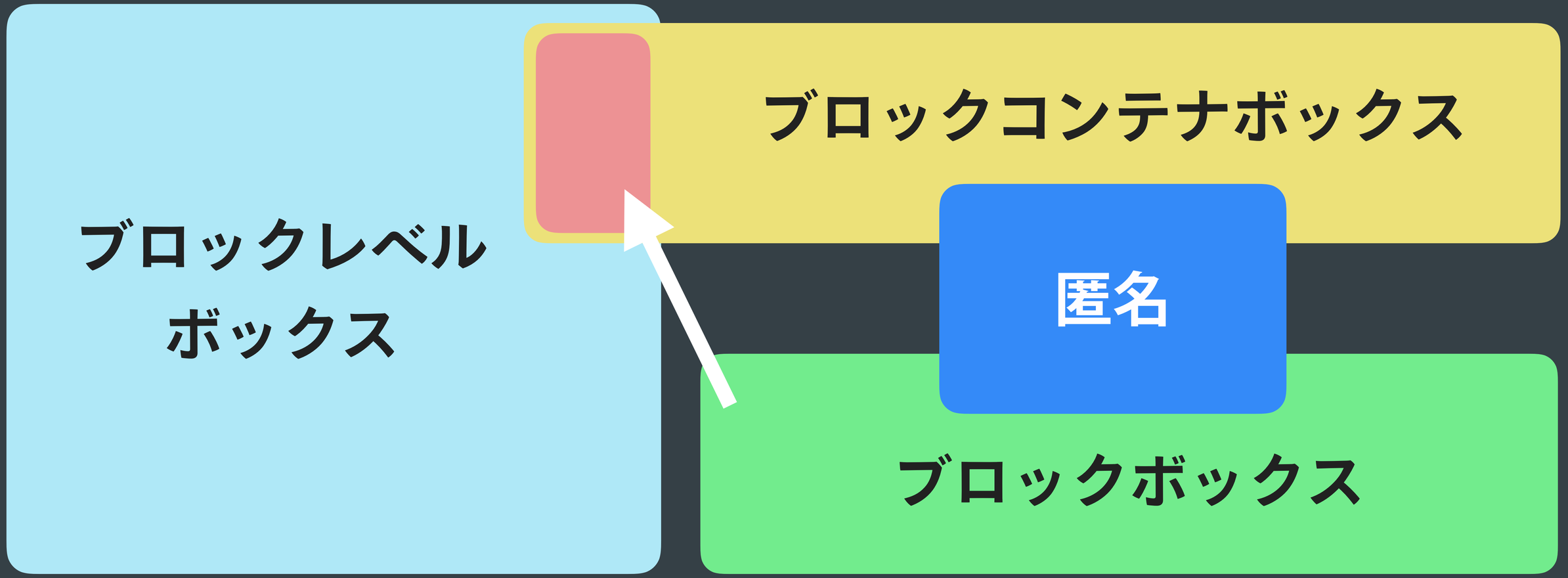
↓

行内に配置

視覚整形モデル: ボックスの生成



視覚整形モデル: ボックスの生成



この場で理解するのは
難しいと思うので
続きはwebで

視覚整形モデル

この文書は翻訳中です。他国語のままの部分などがあるのはその為です。
是非お気軽に MDN に登録して翻訳に参加し、私たちの手助けをして下さい！

CSS の 視覚整形モデル (visual formatting model) は、文書の処理と視覚メディアへの表示に使われるアルゴリズムで、CSS の基本概念です。視覚整形モデルは文書の各要素を変換して、CSS ボックスモデルに従う 0、1、またはいくつかのボックスを生成します。各ボックスの配置は、次のいずれかにより定められます:

- ボックスの大きさ: 精密な定義、制約あり、制約なし
- ボックスのタイプ: インライン、インラインレベル、不可分なインラインレベル、ブロックボックス
- 位置決定スキーム: 通常フロー、フロート、絶対位置指定
- 文書木にある他の要素: 子要素や隣接要素
- ビューポートの大きさや位置
- 含まれる画像の持つ大きさ
- その他外部からの情報

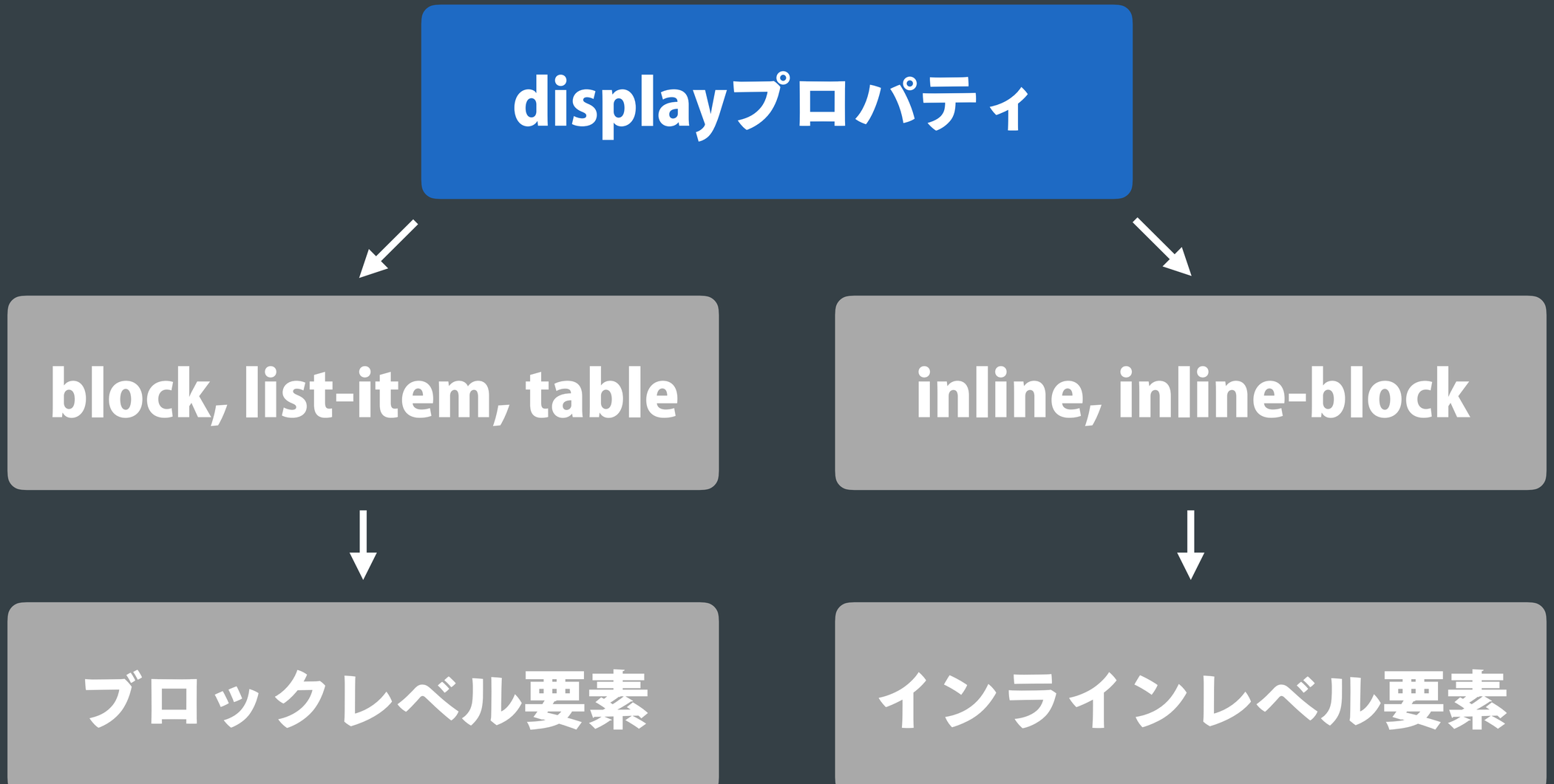
ボックスはその包含ブロックの辺を基準に描かれます。通常、ボックスはその子孫要素のために包含ブロックを作ります。ボックスはそれ自

この記事内

ボックスの生成
ブロックレベル要素とブロックボックス
匿名ブロックボックス
インラインレベル要素とインラインボックス
匿名インラインボックス
その他のボックスのタイプ
行ボックス (line boxes)
ランインボックス (run-in boxes)
モデル起因のボックス (model-induced boxes)
位置決定スキーム (positioning schemes)
通常フロー

https://developer.mozilla.org/ja/docs/Web/CSS/Visual_formatting_model

視覚整形モデル: ボックスの生成



視覚整形モデル: ボックスの生成

displayプロパティを意識する

displayプロパティ

視覚整形モデル: ボックスの生成

displayプロパティや状態を意識する

displayプロパティ

positionプロパティ

floatプロパティ

displayプロパティや状態によって利用できるプロパティが異なる

視覚整形モデル: 状態を意識する

Property ＼ display	block	inline	table	flex
border-spacing	×	×	○	×
width, height	○	×	○	○
vertical-align	×	○	×	×
align-items	×	×	×	○

視覚整形モデル: 状態を意識する

Property ＼ display	block	置換要素	table	flex
border-spacing	×	×	○	×
width, height	○	○	○	○
vertical-align	×	○	×	×
align-items	×	×	×	○

9.5.2 Controlling flow next to floats: the 'clear' property

Name:	***clear***
Value:	none | left | right | both | inherit
Initial:	none
Applies to:	block-level elements
Inherited:	no
Percentages:	N/A
Media:	visual
Computed value:	as specified

This property indicates which sides of an element's box(es) may *not* be adjacent to an earlier floating box. The 'clear' property does not consider floats inside the element itself or in other block for-

https://www.w3.org/TR/CSS22/visuren.html#propdef-clear

CSSの仕様書の「Applies to」を見ればわかる

意図せずにCSSを壊すことも？

- 宣言をとにかく詰め込む
- 宣言の選び方が下手

意図せずにCSSを壊すことも？

- 宣言をとにかく詰め込む
- 宣言の選び方が下手

継承と前景

宣言を設計する

宣言を設計する: 継承と前景

```
.box {
  padding: 20px;
  border: 4px solid #666;
  h1 {
    color: #666;
  }
  p {
    color: #666;
  }
  ul {
    color: #666;
  }
}
```

宣言を設計する: 継承と前景

```
.box {
  padding: 20px;
  border: 4px solid #666;
  color: #666;
}
```

宣言を設計する:継承と前景

```
.box {
  padding: 20px;
  border: 4px solid;
  color: #666;
}
```

宣言を設計する:継承と前景

```
.box {
  padding: 20px;
  border: 4px solid;
  color: #666;
  &.is-strawberry {
    color: #FF2F92;
  }
}
```

widthプロパティ

宣言を設計する

宣言を設計する：widthプロパティ

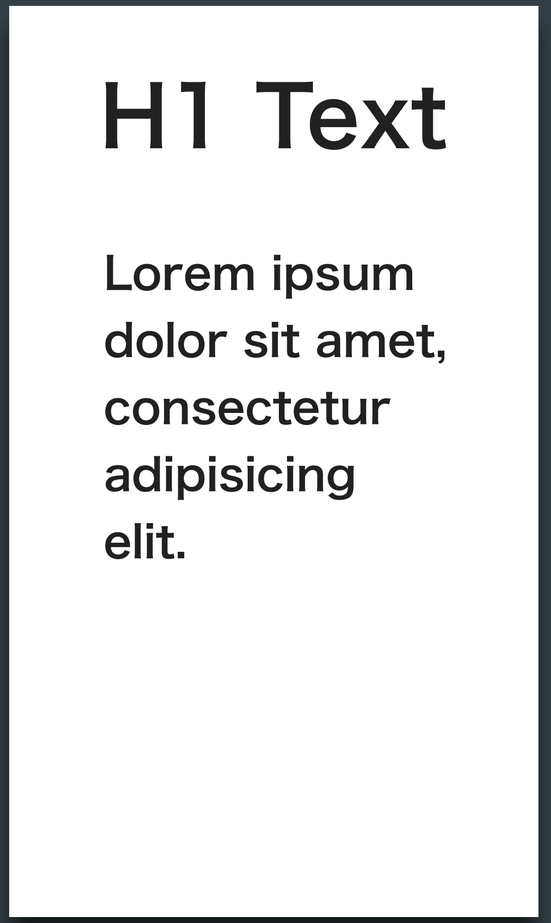
```
.contents {
  width: 980px;
  margin: auto;
}
.box {
  width: 920px;
  padding: 0 30px;
}
```

宣言を設計する: widthプロパティ

```
.contents {
  width: 980px;
  margin: auto;
}
.box {
  width: 920px;
  padding: 0 30px;
}
```

宣言を設計する: widthプロパティ

```
.contents {
  width: 980px;
  margin: auto;
}
.box {
  width: 920px;
  padding: 0 30px;
}
@media screen (max-width: 979px) {
  .contents { width: 100%; }
  .box {
    box-sizing: border-box;
    width: 100%;
  }
}
```



widthプロパティを
無駄に書いて肥大化させている

宣言を設計する：widthプロパティ

```
.contents {
  width: 980px;
  margin: auto;
}
.box {
  width: 920px;
  padding: 0 30px;
}
```

宣言を設計する: widthプロパティ

```
.contents {
  width: 980px;
  margin: auto;
}
.box {
  padding: 0 30px;
}
```

宣言を設計する: widthプロパティ

```
.contents {
  max-width: 980px;
  margin: auto;
}
.box {
  padding: 0 30px;
}
```

宣言を設計する: widthプロパティ

```
.contents {
  max-width: 980px;
  margin: auto;
}
.box {
  padding: 0 30px;
}
```

```
.contents {
  width: 980px;
  margin: auto;
}
.box {
  width: 920px;
  padding: 0 30px;
}
@media screen (max-width: 979px) {
  .contents { width: 100%; }
  .box {
    box-sizing: border-box;
    width: 100%;
  }
}
```

h1 Text

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Iure eius natus ducimus ullam, nisi est, molestias eveniet commodi tenetur, laborum aspernatur voluptas sunt perferendis nobis iusto repellendus illo soluta, distinctio?

Adipisci totam dignissimos veniam nisi illo quis, deleniti dolore. Quibusdam ab qui laborum obcaecati voluptatibus maxime in, beatae suscipit illum, quaerat at provident tempora hic nobis explicabo earum, vel tenetur!

Corporis non quasi quidem, assumenda ut molestias similique dolore hic aut tenetur quos, quibusdam veritatis aliquid minima natus a quo delectus repudiandae labore, culpa in nemo soluta ullam enim aspernatur.

記述の少ないCSSで
スマートに管理

宣言もCSS設計の重要な要素

Sass

CSSメタ言語「Sass」ってなぁに

- CSSを拡張した記述が使えるプリプロセッサーのひとつ
- コンパイルしてCSS形式に変換
- Bootstrap v4でも使われている
- Sass記法とSCSS記法がある

Sassがコンパイルできるツール

Editor


Sublime Text


Brackets

Dreamweaver


Atom


VS Code

WebStorm

GUI


Prepros


Koala


CodeKit

Platform

Node.js

Ruby on Rails

Sassでちゃんとcss

Sassの主な拡張機能

- 変数
- ネスティング
- @import
- @mixin
- @extend
- @function
- プレースホルダーセレクタ
- インターポレーション
- 演算
- &
- @at-root
- 組み込み関数

Sassの主な拡張機能

- 変数
- ネスティング
- @import
- @mixin
- @extend
- @function
- プレースホルダーセレクタ
- インターポレーション
- 演算
- &
- @at-root
- 組み込み関数

変数

変数

- $name: value; の形で宣言
- 大きさ、キーワード、文字列などを変数化できる
  - 10px、50%などの値
  - #000、rgba(0,0,0, 0), redなどのカラー、カラーネーム
  - sans-self、Helveticaなどのフォント名
  - “/path/to/icon.svg” などの文字列

なんでもかんでも変数にしない

```
$main-margin:  30px;
$sub-margin:   15px;
$small-margin: 10px;
$main-padding: 10px;
```

```
$main-box-border:  1px solid #000;
$sub-box-border:   1px solid #999;
$small-box-border: 1px solid #aaa;
$main-box-border2: 2px solid #000;
```

汎用性の高い変数にする

SCSS

```scss
$box-space: 30px;

.box      { margin: $box-space auto }
.box--sub { margin: $box-space/2 auto }
```

演算される

CSS

```css
.box      { margin: 30px auto }
.box--sub { margin: 15px auto }
```

同じカテゴリの変数は「データマップ」でまとめる

SCSS

```scss
$colors: (
  main: #000,
  sub:  #999,
  small: #aaa
);

.box {
  color: map-get($colors, main);
}
.box--sub {
  color: map-get($colors, sub);
}
.box--small {
  color: map-get($colors, small);
}
```

CSS

```css
.box {
  color: #000;
}

.box--sub {
  color: #999;
}

.box--small {
  color: #aaa;
}
```

ネスティング

宣言ブロックを入れ子で記述できる

SCSS

```scss
.gnav {
  .gnav__list {
    .gnav__item {
      color: gold;
    }
  }
}
```

HTML

```html
<nav class="gnav">
  <ul class="gnav__list">
    <li class="gnav__item"><a href...
    <li class="gnav__item"><a href...
    <li class="gnav__item"><a href...
  </ul>
</nav>
```

CSS

```css
.gnav .gnav__list .gnav__item {
  color: gold;
}
```

```
.gnav .gnav__list .gnav__item {
  color: gold;
}
```

詳細度0.3.0

「&」記号を使用

SCSS

```
.gnav {
  &__item {
    color: gold;
  }
}
```

& = .gnav

& = .gnav__item

さっきのSCSS

```
.gnav {
  .gnav__list {
    .gnav__item {
      color: gold;
    }
  }
}
```

CSS

```
.gnav__item {
  color: gold;
}
```

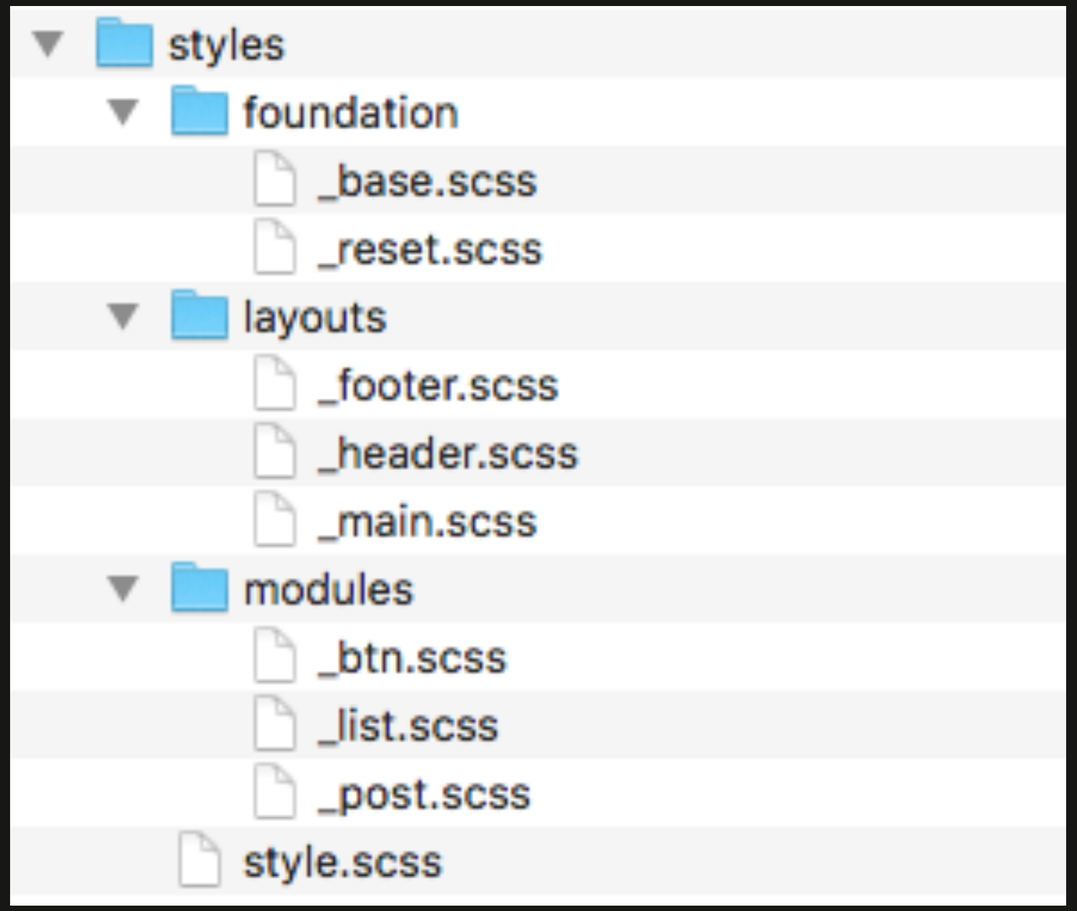
@import

@import

- @import ‘path/to/scss-partial-file’; で宣言
- コンパイル時にひとつに結合される
- パーシャルファイルの頭に _ をつけたファイルは
  個別に出力されない

パーシャルファイルをまとめて@import

File Tree



style.scss

```
@import 'foundation/reset';
@import 'foundation/base';

@import 'layouts/header';
@import 'layouts/footer';
@import 'layouts/main';

@import 'modules/btn';
@import 'modules/list';
@import 'modules/post';
```

@importの良くない例

File Tree

```
modules
  ├── _company.scss
  ├── _contact.scss
  ├── _service.scss
  └── _top.scss
```

_top.scss

```scss
.top {
  &-heading {
    &--sub {...}
  }
  &-contents {
    &__inner {...}
  }
  &-carousel {
    &__item {...}
  }
  &-footer {
    &__inner {...}
  }
}
```

@importの良い例

File Tree

```
modules
    ├── ...
    ├── _top-heading.scss
    ├── _top-contents.scss
    ├── _top-carousel.scss
    └── _top-footer.scss
```

モジュールごとにパーシャルファイルにする

@mixin

@mixin

- @mixin hoge(args) {...} の形で設定
- @include hoge(fuga) {property: value} の形で使用
- 引数を渡して宣言ブロックを新しく展開できる
  - 引数はなくてもいい
  - 使用時の宣言ブロックの中の記述は @content で取得可能
- 冗長な記述やスニペット的な宣言の処理に向いている

マップで冗長になったメディアクエリをmixin化

SCSS

```scss
$breakpoints: (
  xs: 0,
  sm: 576px,
  md: 768px,
  lg: 992px,
  xl: 1200px
);

@mixin minw($bp) {
  @media (min-width: map-get($breakpoints, $bp)+1) {
    @content;
  }
}
```

マップで冗長になったメディアクエリをmixin化

SCSS

```scss
.container {
  padding: 30px;

  @include minw(md) {
    padding: 60px;
  };
}
```

CSS

```css
.container {
  padding: 30px;
}

@media (min-width: 769px) {
  .container {
    padding: 60px;
  }
}
```

@function

@function

- @function hoge(arg) {...} の形で宣言
- hoge(arg) の形で使用
- 値を返す
  - mixinはプロパティーと値のセットを返すが、functionは値しか返せない
  - 何らかの処理をした結果をプロパティーの「値」として使う

@function

SCSS

```scss
@function fugaPiyo($arg) {
  $_var: null;
  @if $arg == “hoge” {
    $_var: “fuga”;
  } @else {
    $_var: “piyo”;
  }
  @return $_var;
}

.hoge::after {
  content: fugaPiyo(hoga);
}
```

CSS

```css
.hoge::after {
  content: “piyo”;
}
```

z-indexを管理

SCSS

```scss
$z-map: (
  header, nav, modal
);

@function z($name) {
  @return index($z-map, $name);
}

.header {
  z-index: z(header);
}
.modal {
  z-index: z(modal);
}
```

CSS

```css
.header {
  z-index: 1;
}

.modal {
  z-index: 3;
}
```

Sassmeister File View Options Login with Github

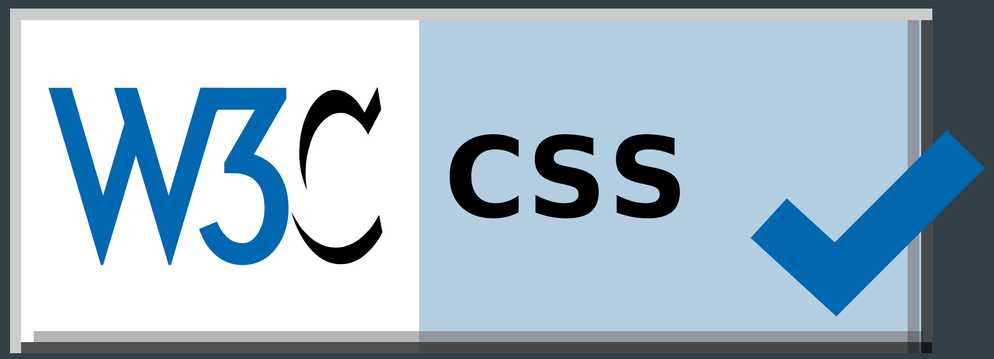
Sass v4.0.0.alpha.1

```scss
$z-maps: (
  header,
  nav,
  modal
);

@function z($name) {
  @return index($z-maps, $name);
}

.header {
  z-index: z(header);
}

.modal {
  z-index: z(modal);
}
```

CSS

```css
.header {
  z-index: 1;
}

.modal {
  z-index: 3;
}
```

CSSをちゃんと書く
まとめ



まとめ: CSS

- **displayプロパティ**により、**要素のタイプ**が分かれる
- **要素の状態**を**意識**して、宣言を行う
- **宣言**もCSS設計の**重要な要素**

まとめ: Sass

- **意味のあるメタ化**
- 機能を活用した**コーディングの効率化**
- **CSSのためのSassを構築**

ありがとうございました