

コンポーネント指向と 余白の設計

CSS Talk vol.02 Manabu Yasuda

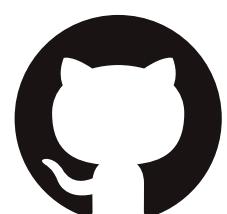
INTRODUCTION

自己紹介

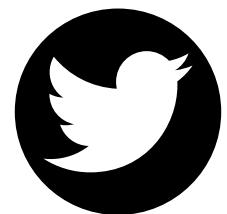


安田 学 (Yasuda Manabu)

株式会社 TAM
マークアップエンジニア



<https://github.com/manabuyasuda>



@Gaku0318

アジェンダ

01. コンポーネント指向とはなにか
02. コンポーネントの粒度を整理する
03. コンポーネント同士をレイアウトする
04. 余白設計のパターンを考える

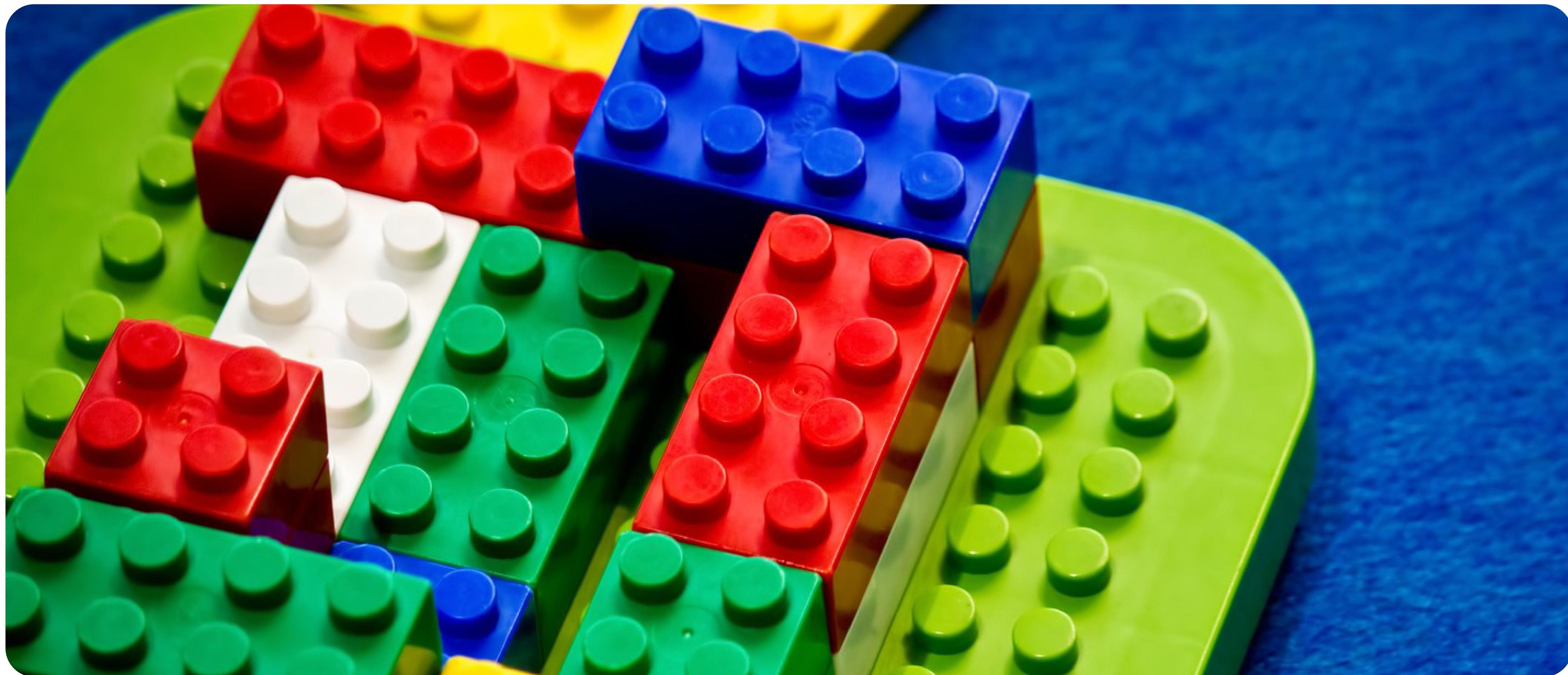
01

コンポーネント指向とはなにか

コンポーネント指向とはなにか

- ・コンポーネント = UI を部品化した状態
- ・コンポーネントの組み合わせ = ページ

コンポーネントのイメージ



コンポーネントのイメージ



コーディング面でのメリット

- ・ 小さな単位でコードを考えることができる
- ・ 理解しやすく保守しやすいコードになる

デザイン面でのメリット

- ・ デザインの再利用ができる
- ・ デザインの手離れがはやい

02

コンポーネントの粒度を整理する

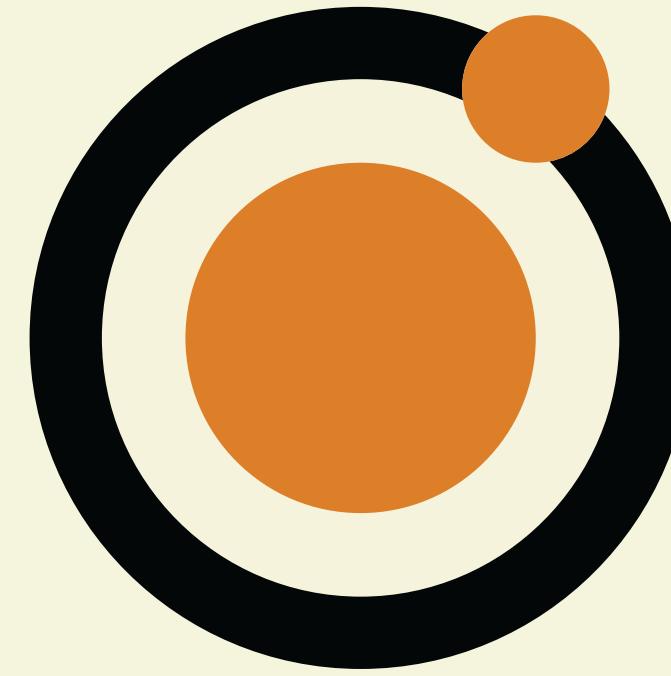
コンポーネントの粒度の指針

Atomic Design (アトミックデザイン)

- ① Atoms (アトム・原子)
- ② Molecules (モルキュー・分子)
- ③ Organisms (オルガニズム・生物)
- ④ Templates (テンプレート)
- ⑤ Pages (ページ)

Atomic Design (アトミックデザイン)

① Atoms (アトム・原子)



機能的にこれ以上分割できないコンポーネント

Atomic Design (アトミックデザイン)

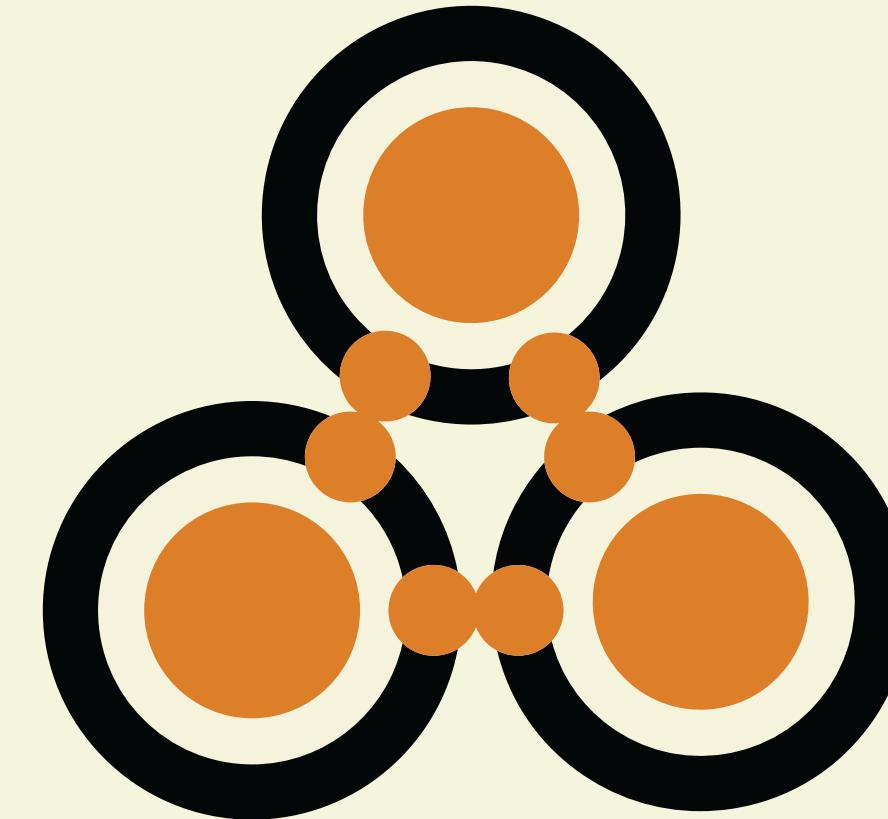
Search



入力フィールドとボタン

Atomic Design (アトミックデザイン)

② Molecules (モルキュー・分子)



Atoms を組み合わせた比較的シンプルなコンポーネント

Atomic Design (アトミックデザイン)

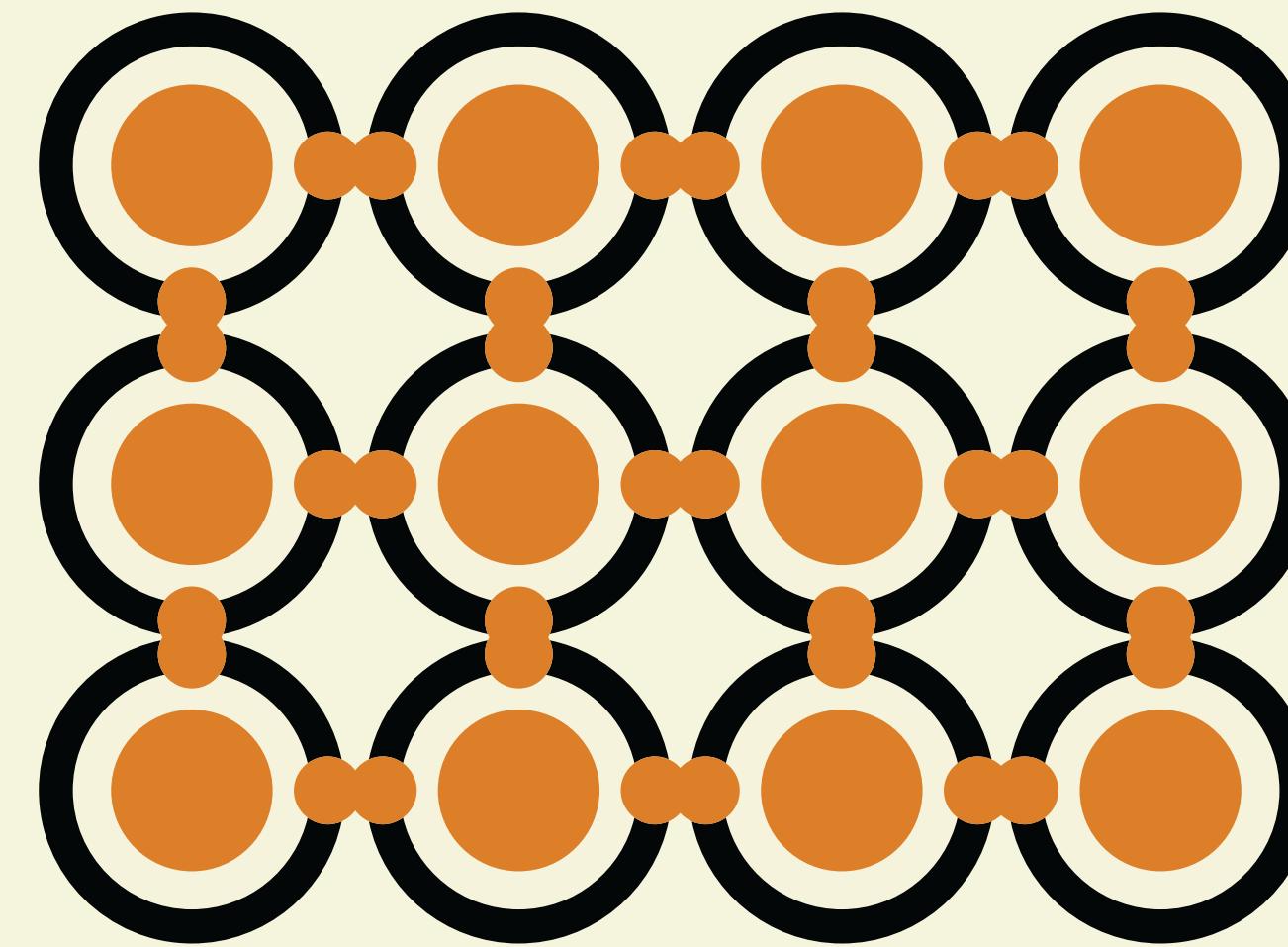
Search



検索フォーム

Atomic Design (アトミックデザイン)

③ Organisms (オルガニズム・生物)



Atoms や Molecules を組み合わせた比較的複雑な
コンポーネント

Atomic Design (アトミックデザイン)



グローバルヘッダー (左からサイトロゴ、ナビゲーション、検索フォーム)

Atomic Designを導入するメリット

- ・コンポーネントの大きさを基準にするというコンセプトが分りやすい
- ・コンポーネントは具体的な見た目を持っている
- ・コンポーネントの粒度が適度に細かい

03

コンポーネント同士をレイアウトする

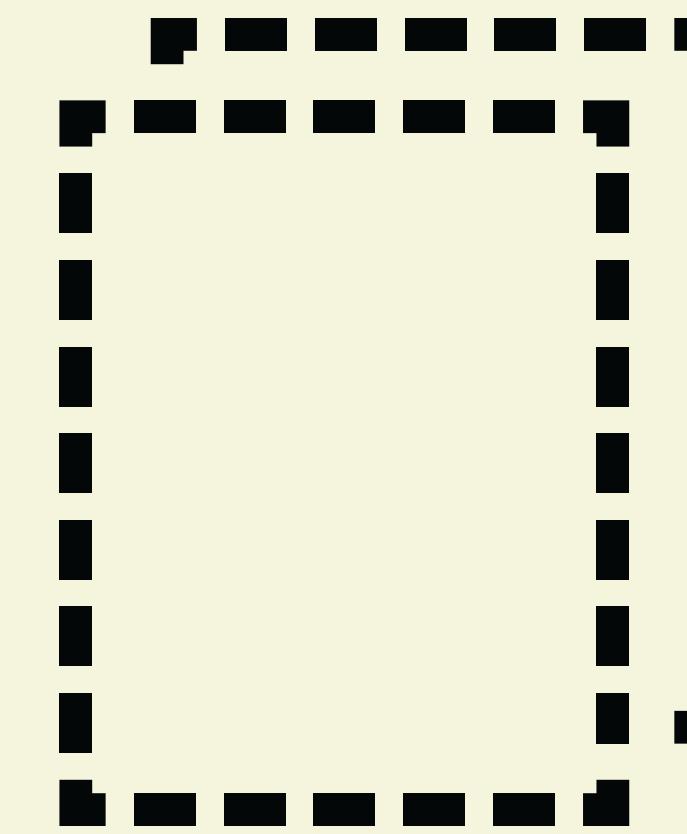
コンポーネントは、 内包するコンポーネントに影響を与えられる

- Atomsは内包するコンポーネントを持たない
- MoleculesはAtomsを上書きできる
- OrganismsはMoleculesとAtomsを上書きできる

コンポーネント同士は
Templates (テンプレート) でレイアウトする

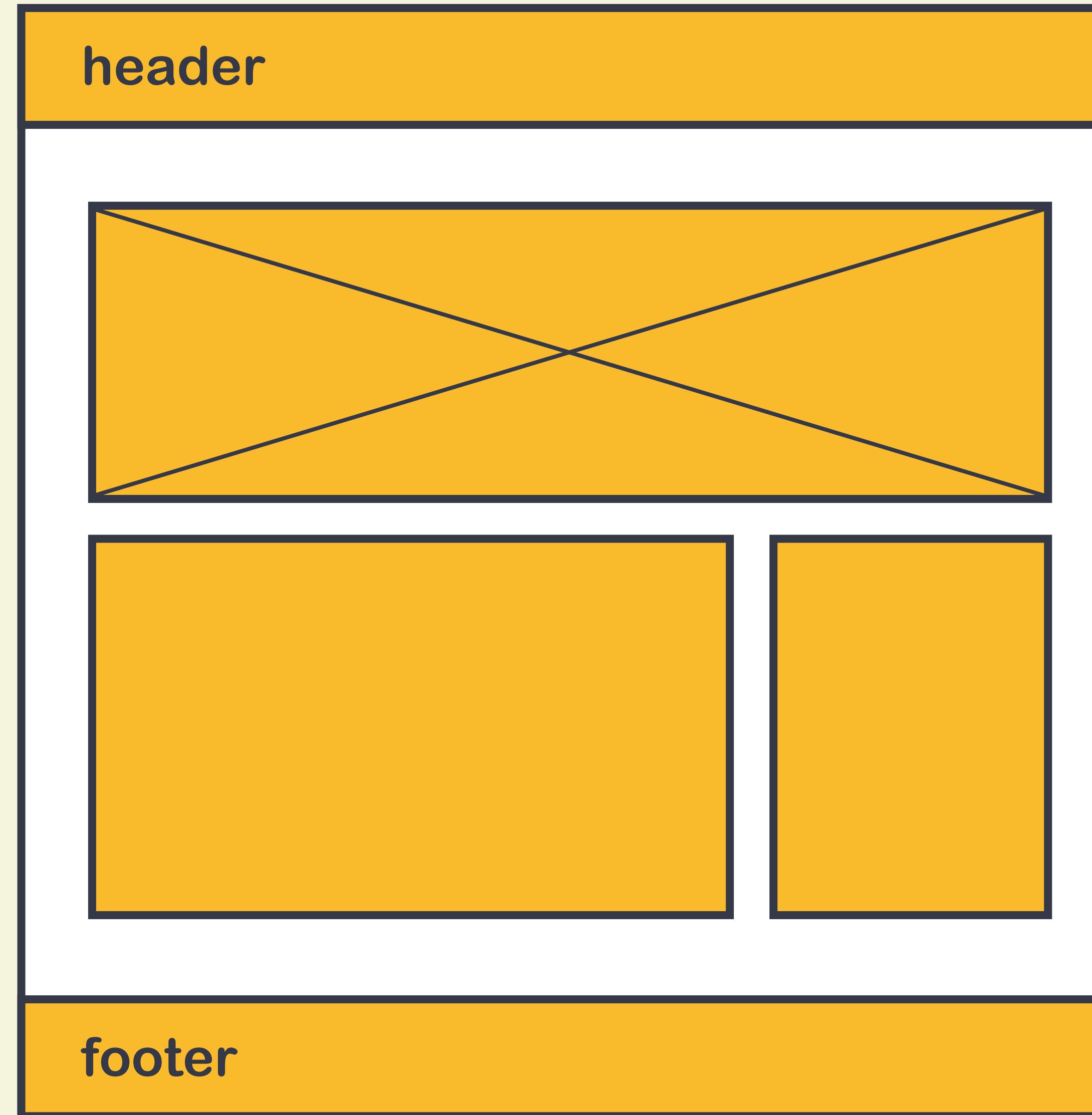
Atomic Design (アトミックデザイン)

④ Templates (テンプレート)



ページレベルのオブジェクトで、
コンポーネント (Atoms・Molecules・Organisms) を配置して
ページの構成を整理する

Atomic Design (アトミックデザイン)



ワイヤーフレームのような
レイアウト

04

余白設計のパターンを考える

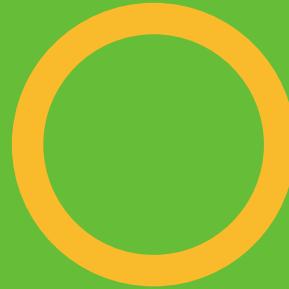
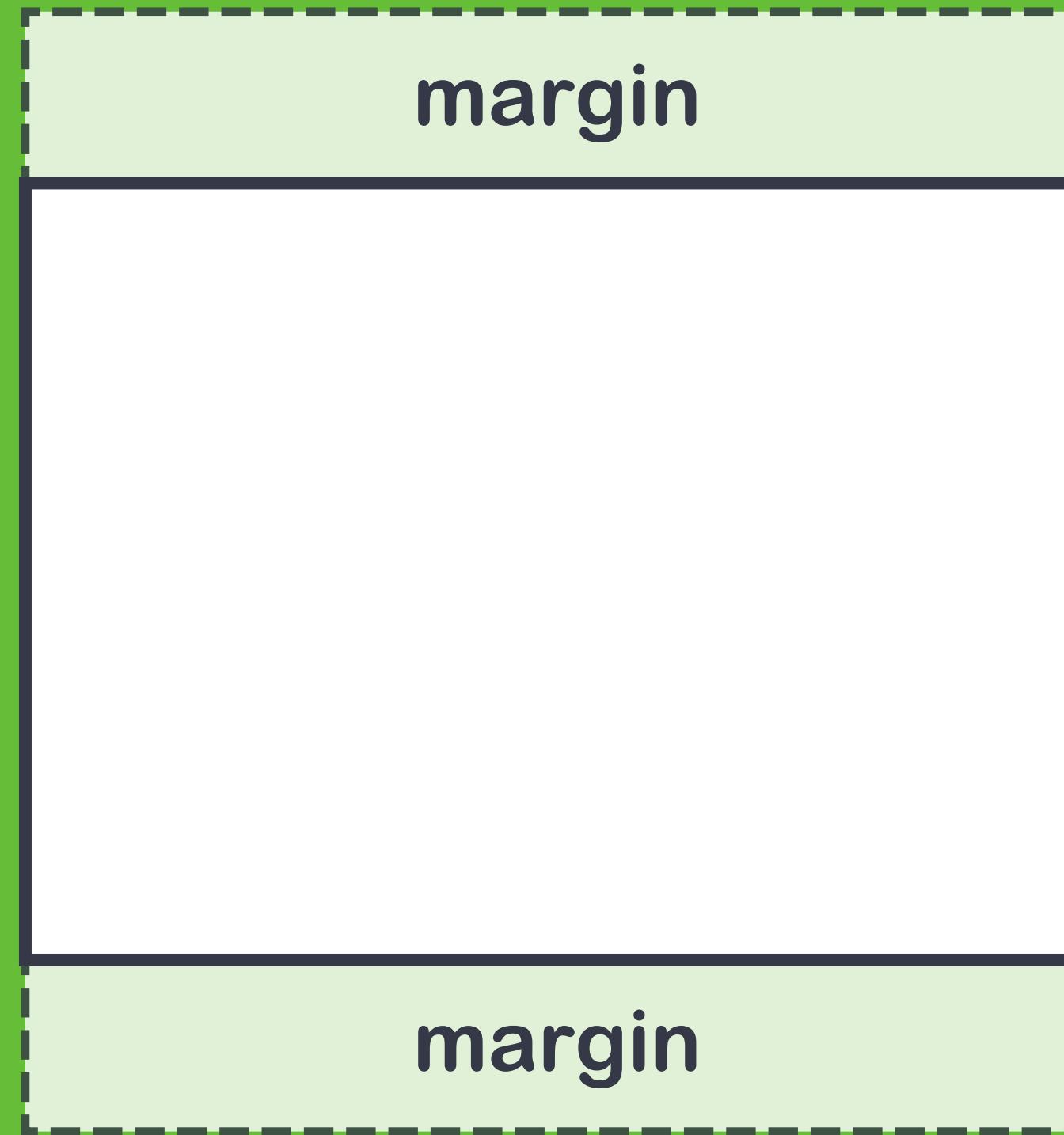
余白設計のパターンを考える

- ① margin-bottom パターン
- ② last-child パターン
- ③ Global パターン
- ④ Section パターン
- ⑤ Inset パターン
- ⑥ Grid パターン
- ⑦ Component パターン
- ⑧ Body パターン

① margin-bottom パターン - 余白設計の考え方

- ・マージンの相殺をできるだけ避けるため
上下両方の余白は避ける
- ・ページは上から下の方向で読むのが自然なので
下方向に余白をつけたほうが理解しやすい

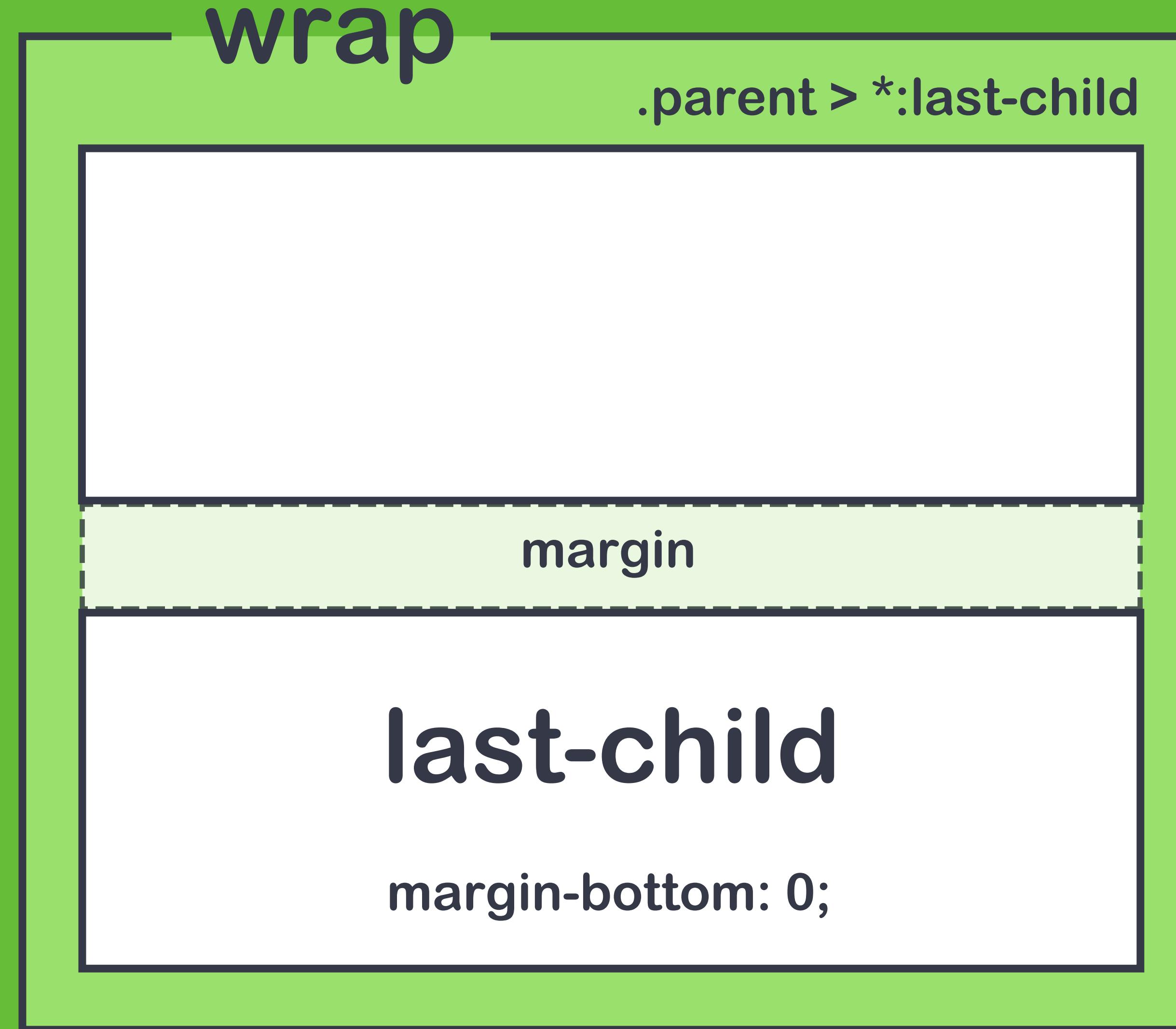
① margin-bottom パターン - 余白設計の考え方



② last-childパターン - 余白設計の考え方

- ・コンポーネントには必ずラップ要素を用意する
- ・ラップ要素の直下にある最後の要素のmargin-bottomを0にする

② last-child パターン - 余白設計の考え方

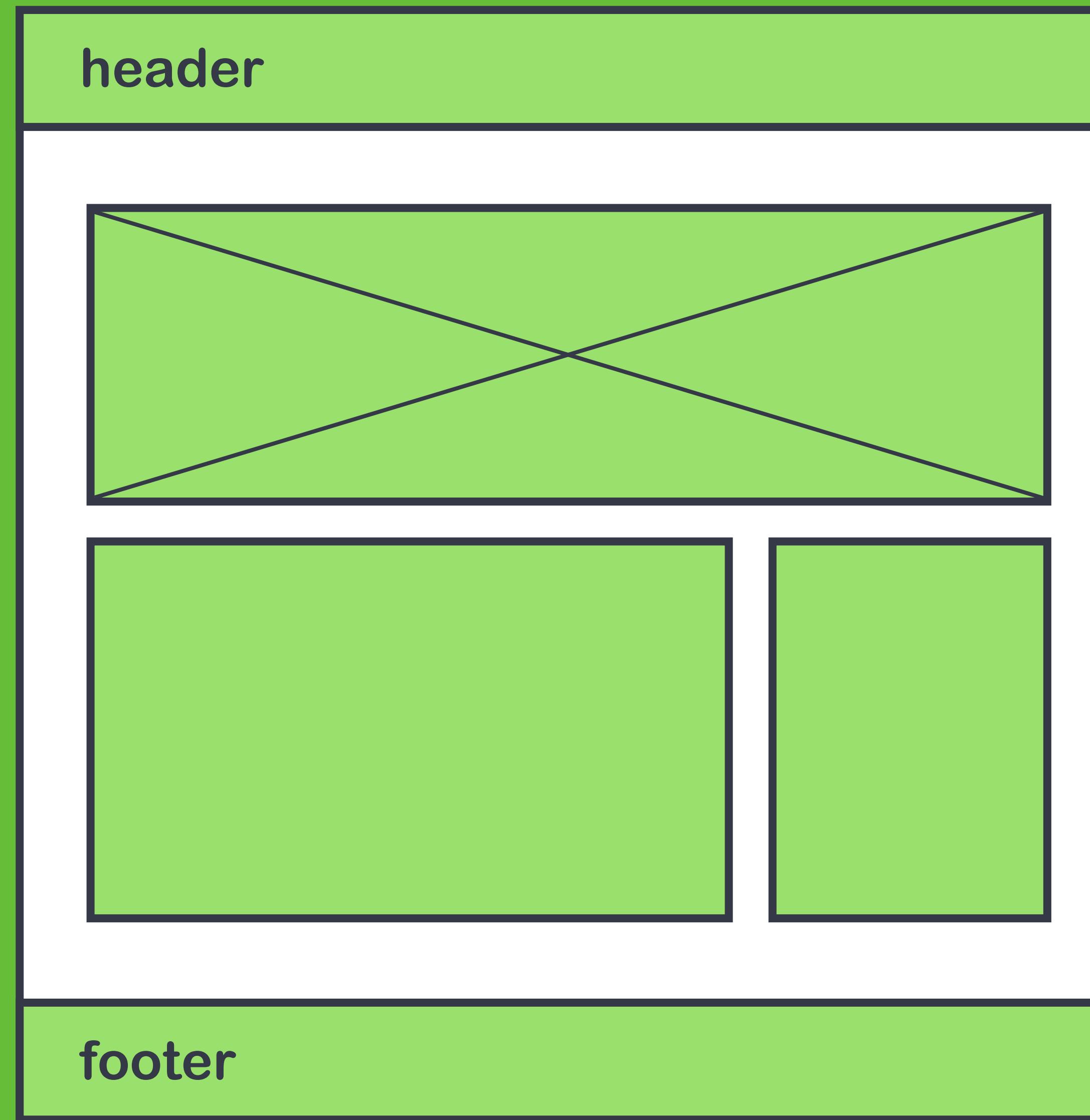


直下にある最後の要素の
マージンを確実に0にできる

③ Global パターン - 余白設計の考え方

- ・ヘッダーやフッター、コンテンツといった共通部分のレイアウトは基本的に変わらない
- ・確定しているところから組み立てる

③ Global パターン - 余白設計の考え方

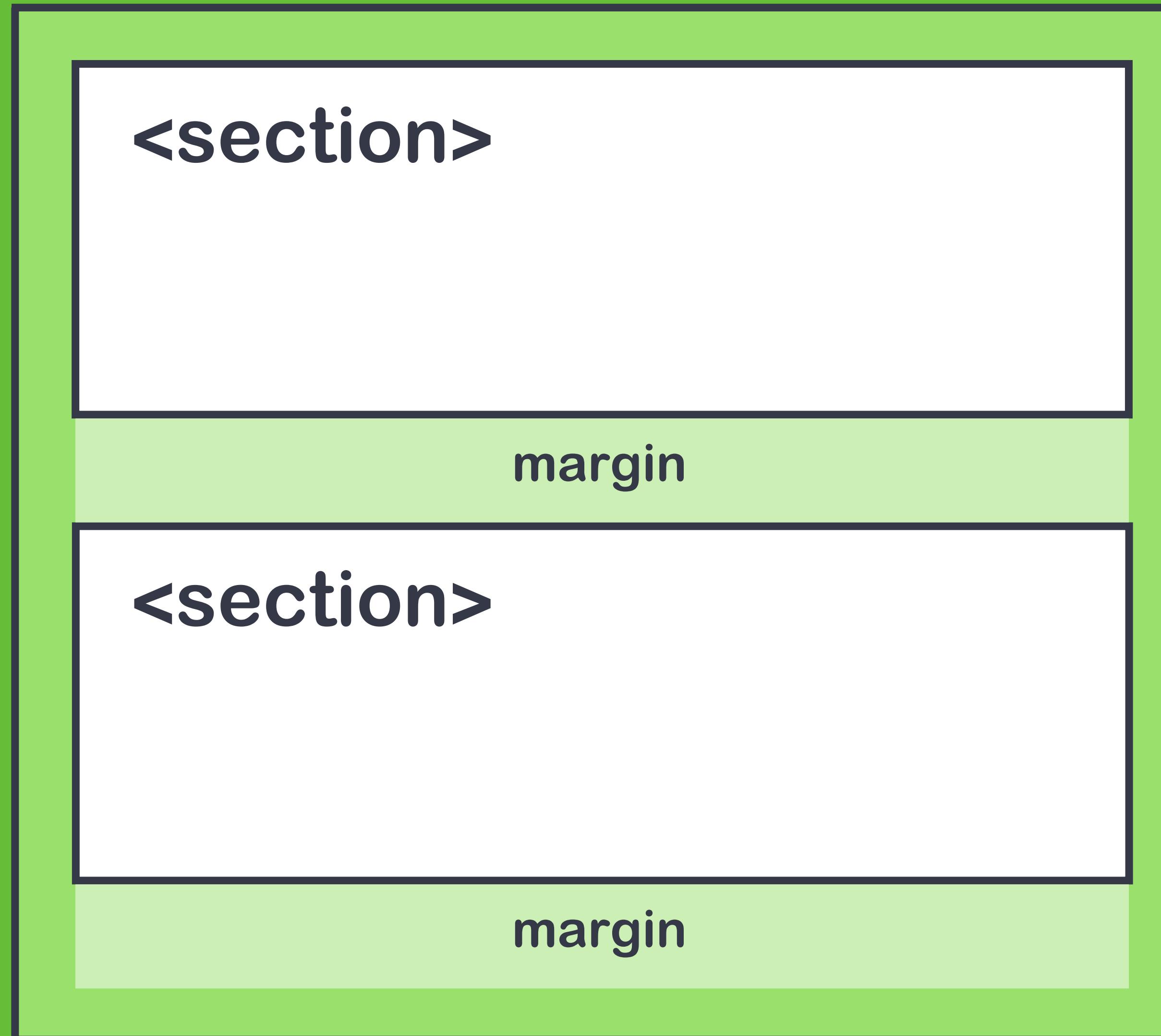


Globalパターンは
ワイヤーフレームのような
大きなレイアウトパターン

④ Section パターン - 余白設計の考え方

- sectionを目安に余白を設計する
- 自然にマシンリーダブルになる

④ Section パターン - 余白設計の考え方

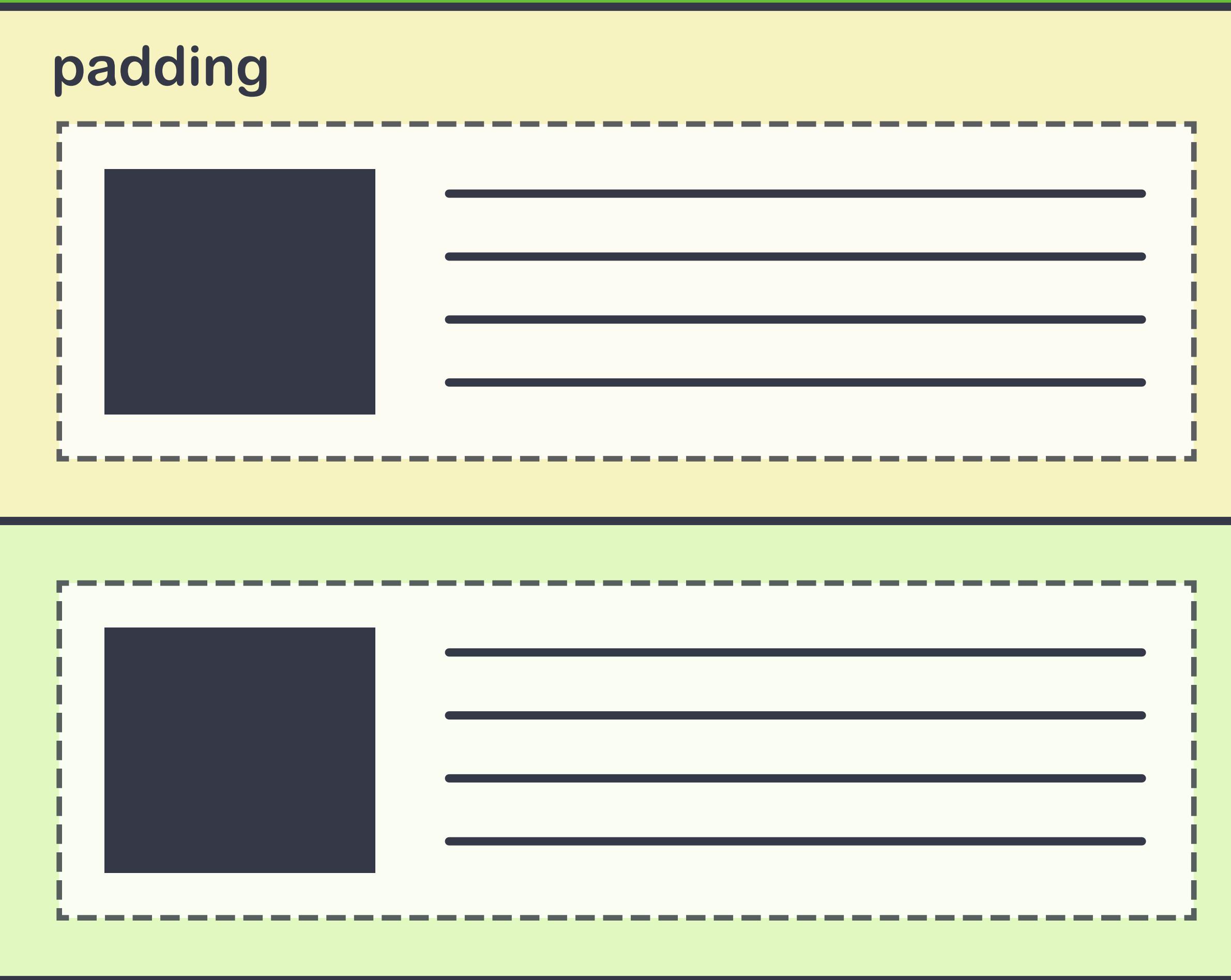


sectionとsectionの間に
大きい余白が入る

⑤ Inset パターン - 余白設計の考え方

- marginではなくpaddingで余白をとる
- 背景色をともなうことが多い

⑤ Inset パターン - 余白設計の考え方

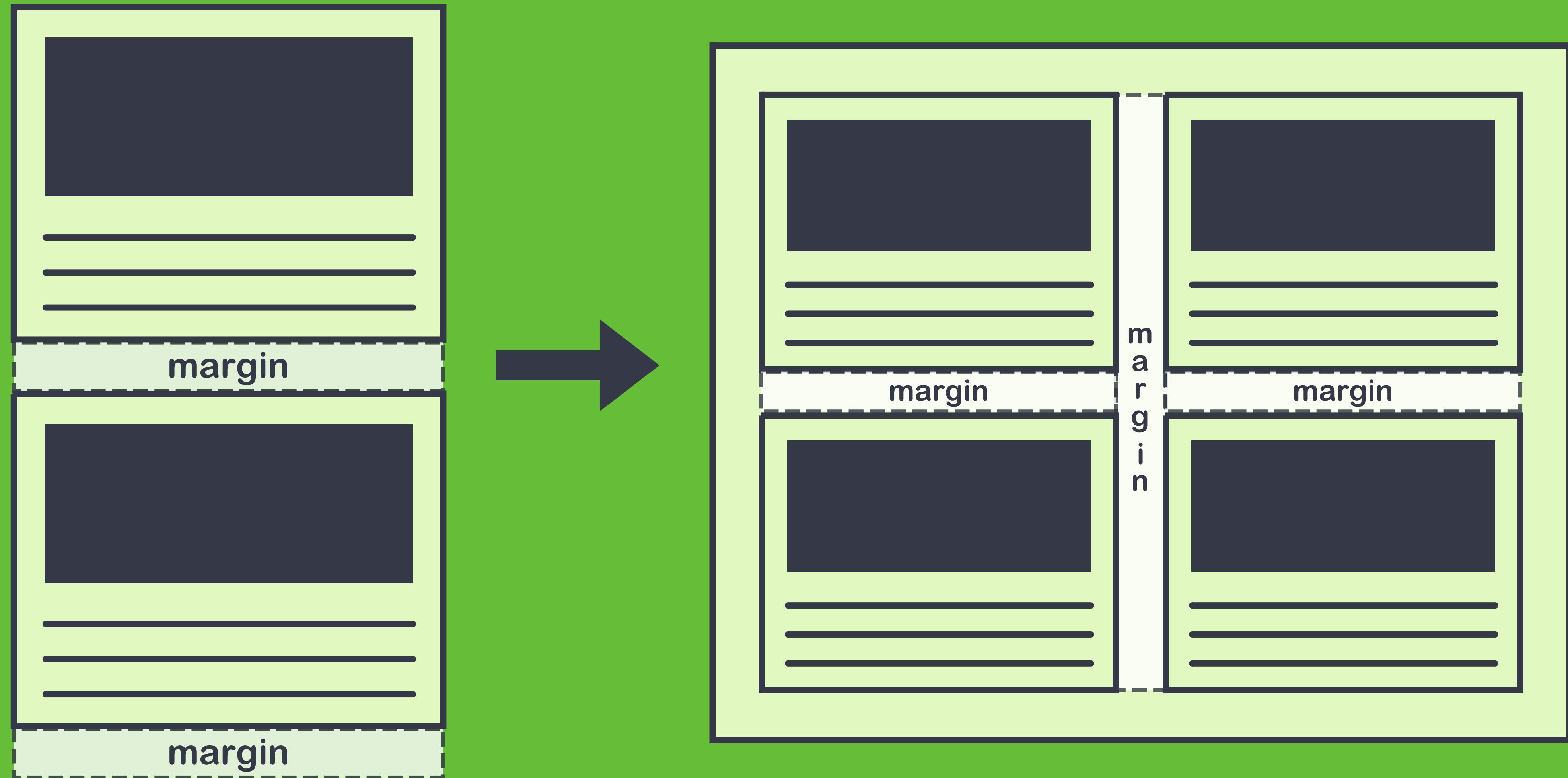


それぞれの sectionごとに
paddingで上下左右の余白を
つける

⑥ Gridパターン - 余白設計の考え方

- ・コンポーネントのパターン化
- ・レスポンシブ対応

⑥ Grid パターン - 余白設計の考え方



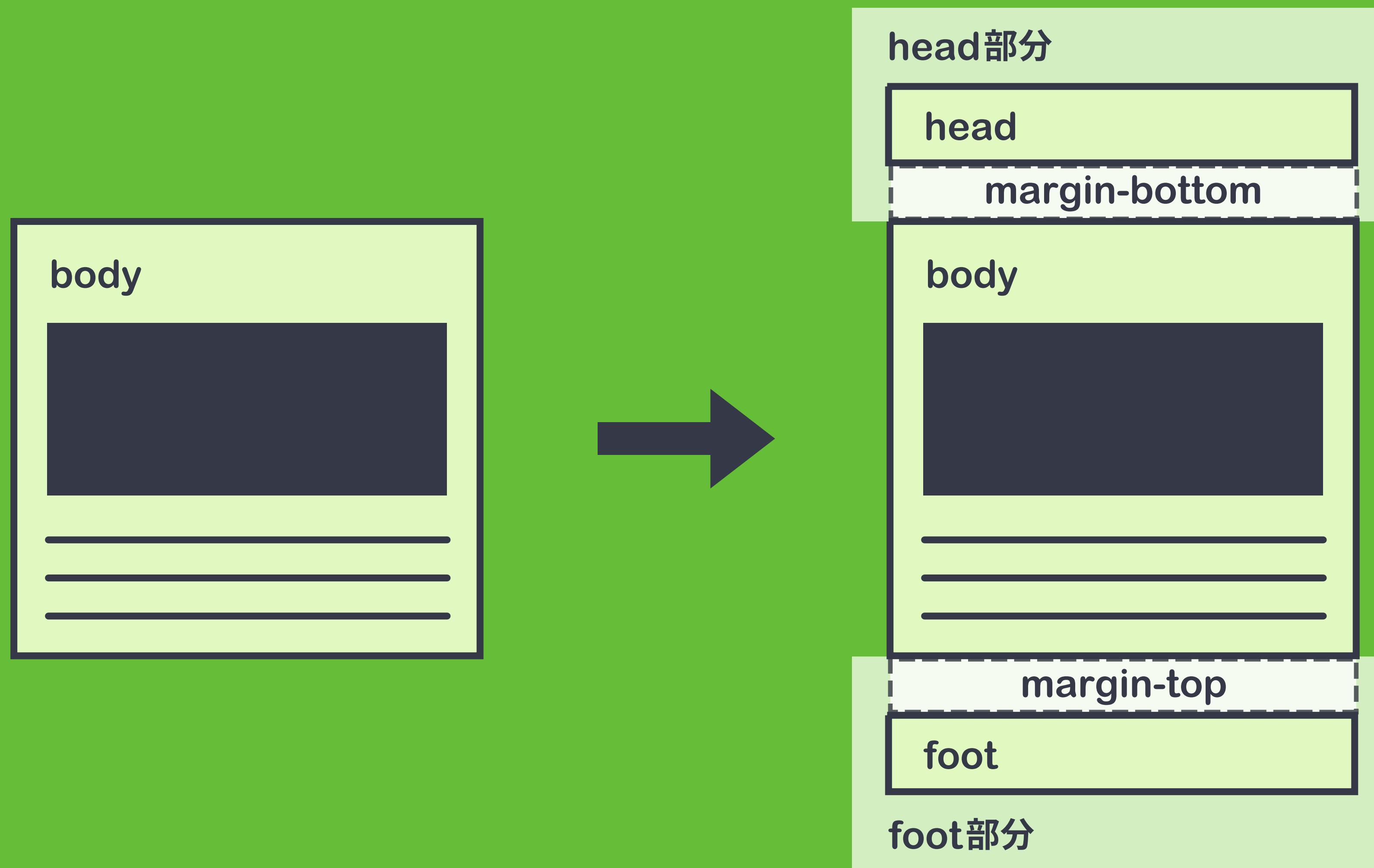
⑦ Componentパターン - 余白設計の考え方

- 汎用的なGrid パターンは適材適所
- コンポーネントごとに
専用の余白を持たせることも検討する

⑧ Bodyパターン - 余白設計の考え方

- ・必ず必要な要素(body)を基準に考える
- ・必ずしも必要ではない要素(head,foot)に余白をつける

⑧ Bodyパターン - 余白設計の考え方



まとめ

- UIを部品化(コンポーネント化)する
- コンポーネントを組み合わせてページを作る
- Atomic Designでコンポーネントの粒度を明確にする
- 8つの余白パターンをベースに余白設計を考える

ありがとうございました。

slide writing : yasuda manabu

slide design : nakajima eri