

本当に役に立つ!!

UIデザインの法則50 ユーザビリティ チェックリスト

総集編

Contents

01 システムステータスの可視化
Visibility of system status

02 システムと現実世界の一致
Match between system and the real world

03 ユーザーのコントロールと自由度
User control and freedom

04 一貫性と標準化
Consistency and standards

05 エラー防止
Error prevention

06 想起よりも認識
Recognition rather than recall

07 使用の柔軟性と効率性
Flexibility and efficiency of use

08 美的でミニマルなデザイン
Aesthetic and minimalist design

09 エラー時のユーザーによる認識・診断・回復の補助
Help users recognize, diagnose, and recover from errors

10 ヘルプとドキュメント
Help and documentation

01

システムステータスの可視化

Visibility of system status



1

システムステータスの可視化

入力項目が多い時はステップを分ける

The diagram illustrates the transformation of a form with many input items into a multi-step process. On the left, labeled 'Before', is a single form with sections for 'お届け先' (Address) and 'お支払い方法' (Payment Method). The 'お届け先' section includes fields for '名前' (Name: 田中太郎), 'フリガナ' (Kana: タナカタロウ), '郵便番号' (Postal Code: 1230045), and '住所' (Address: 東京都渋谷区1-2-5). The 'お支払い方法' section has radio buttons for '現金' (Cash) and 'クレジットカード' (Credit Card). A '住所自動入力' (Address Auto-fill) button is also present. On the right, labeled 'After', is a three-step process. Step 1, 'お届け先', contains the name and kana fields. Step 2, 'お支払い方法', contains the postal code and address fields. Step 3, '注文内容の確認' (Order Confirmation), contains the payment method field. A '住所自動入力' button is also present in step 2. An arrow points from the 'Before' form to the 'After' process.

フォームの入力項目が多い場合は、項目をグルーピングして画面を分割しましょう。

フォームが長すぎると、ユーザーは入力を途中で辞めてページから離脱してしまうかもしれません。

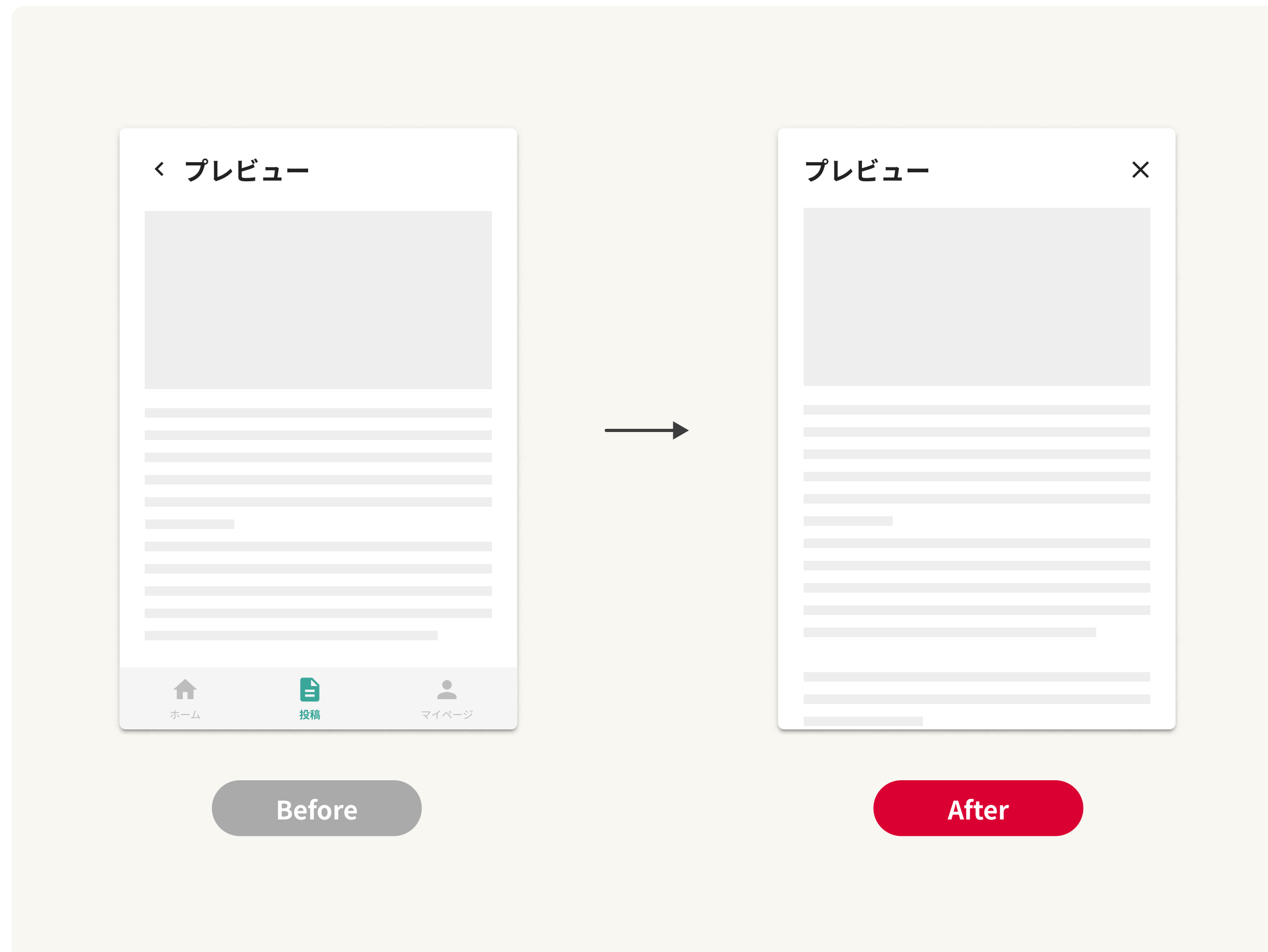
その上で、ステッパーを設置して現在の進捗状況を明示しましょう。

ステッパーがあることで進捗状況を常に確認することができ、完了までの道のりがどれくらいあるかわかるためモチベーションの向上にもつながります。

2

システムステータスの可視化

モードを意識させる



モードに入っているときには、それ以外の操作が制限されています。そのため、今の状態がモードなのか否かをユーザーに明示する必要があります。

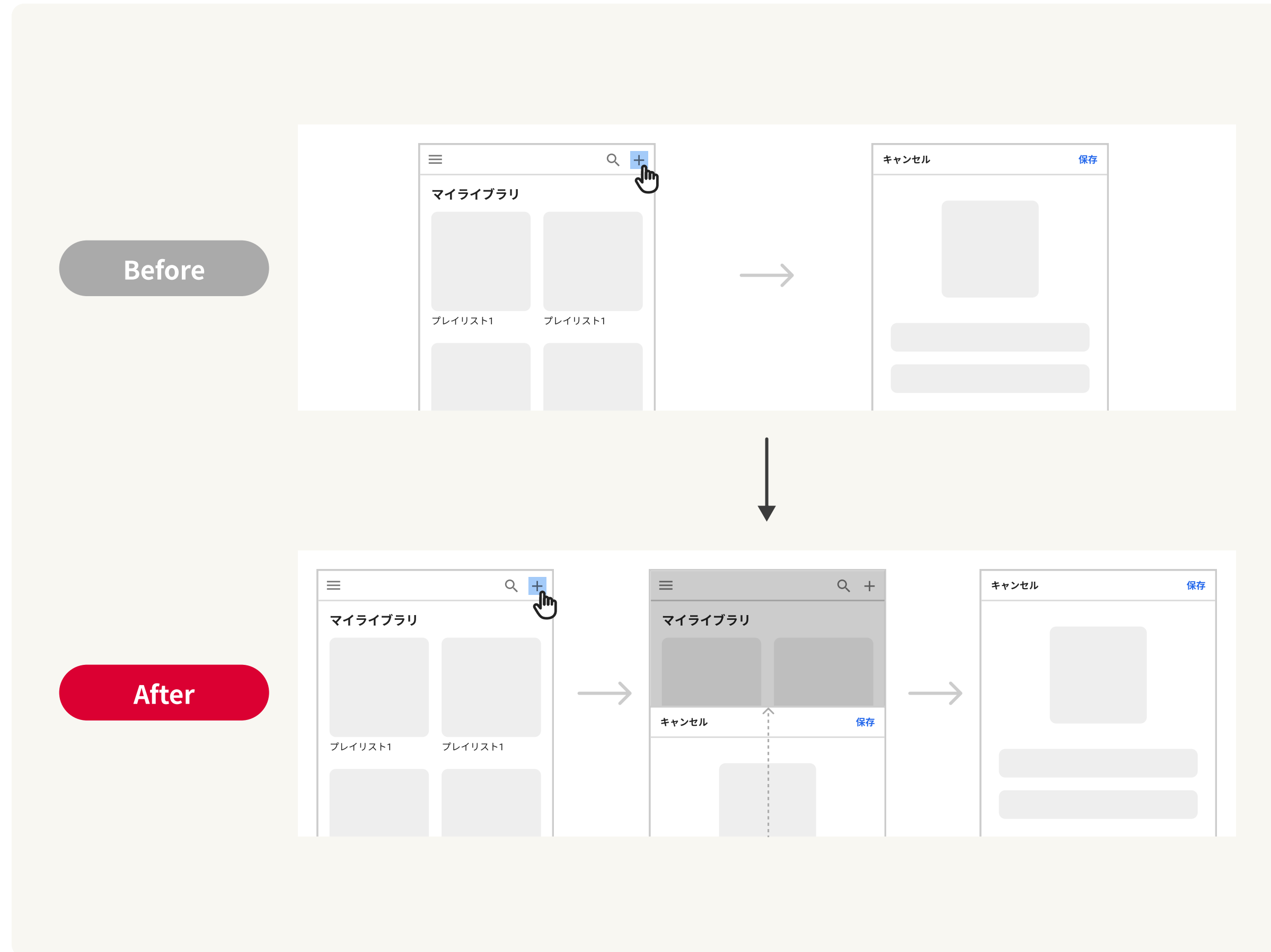
モードが認識されていない暗黙的なモードでは、ユーザーがモードから離脱しようとした時に「中断しますがよろしいですか？」などと引き止めることになります。ユーザーは、アラートの表示によって初めてモードに入っていることを認識することになります。（当然、意図しないアラートが出ることはユーザーにとって好ましくありません）

iOSアプリであれば、モードに入った画面ではTab barのような離脱する動線を配置しないことや、画面上部に「〇〇中」といったバーを出すなど”モード中”であることを認識できるように明示するなどの配慮が必要です。

3

システムステータスの可視化

画面の変化をアニメーションで表す

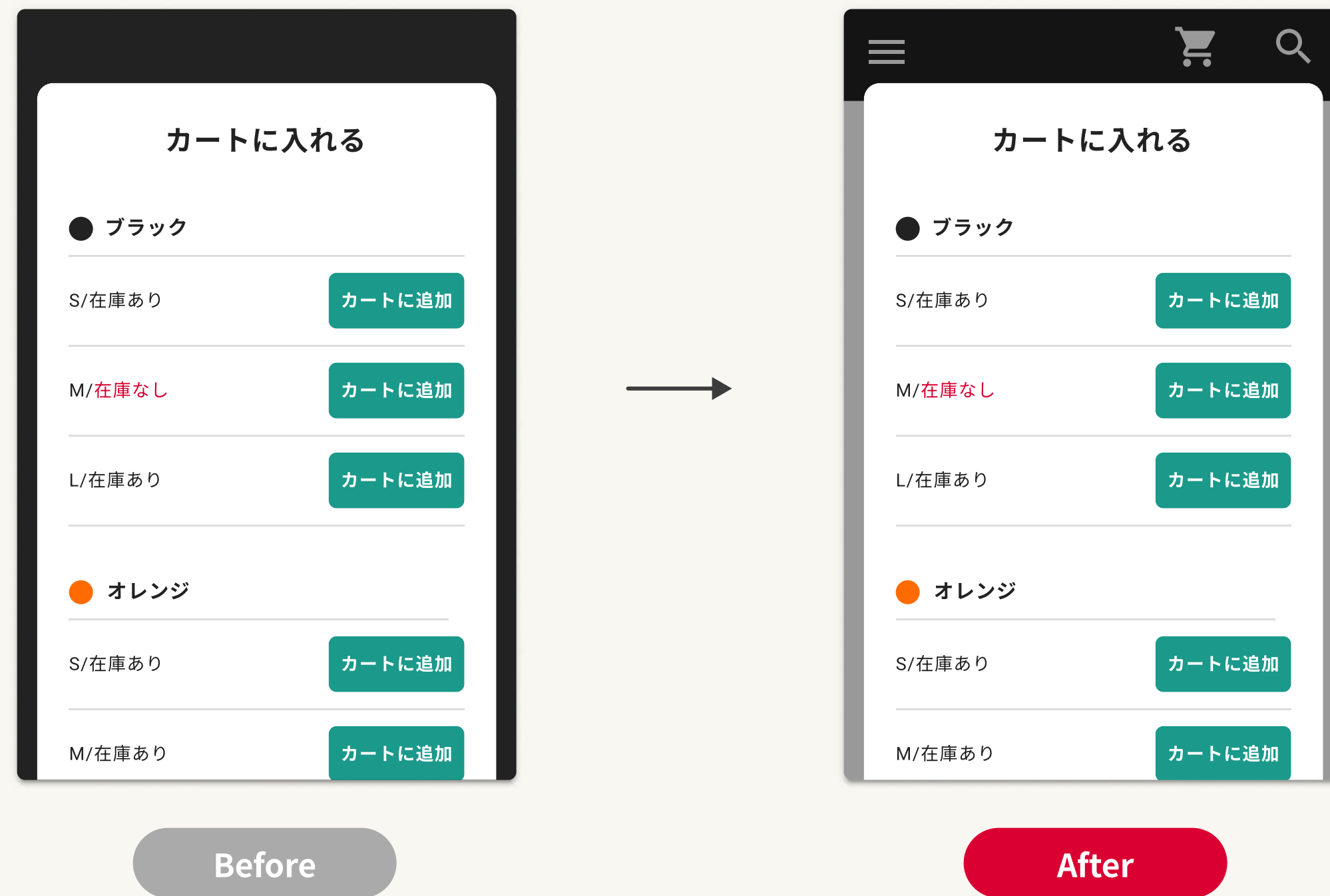


0→1で一瞬で画面が切り替わると状態遷移の連続性をユーザーが把握できないためシステムの概念的構造をイメージすることができません。

アニメーションをつけることでどのような過程を経て変化したのか確認することができるため、ユーザーは無意識のうちにシステムの概念的構造を想起することができます。

特にモバイルのような表示領域が狭い空間では、画面を分けたり、操作によって表示/格納を切り替えるUIが頻出しますが、こうした仮想の情報空間における構造や位置関係のルールを作ることで、ユーザーは情報へアクセスしやすくなります。

オーバーレイは背景を透過させる



ボタンを押してオーバーレイしたときに背景が透過されていないとユーザーは新しいページに移動したと誤解してしまう可能性があります。

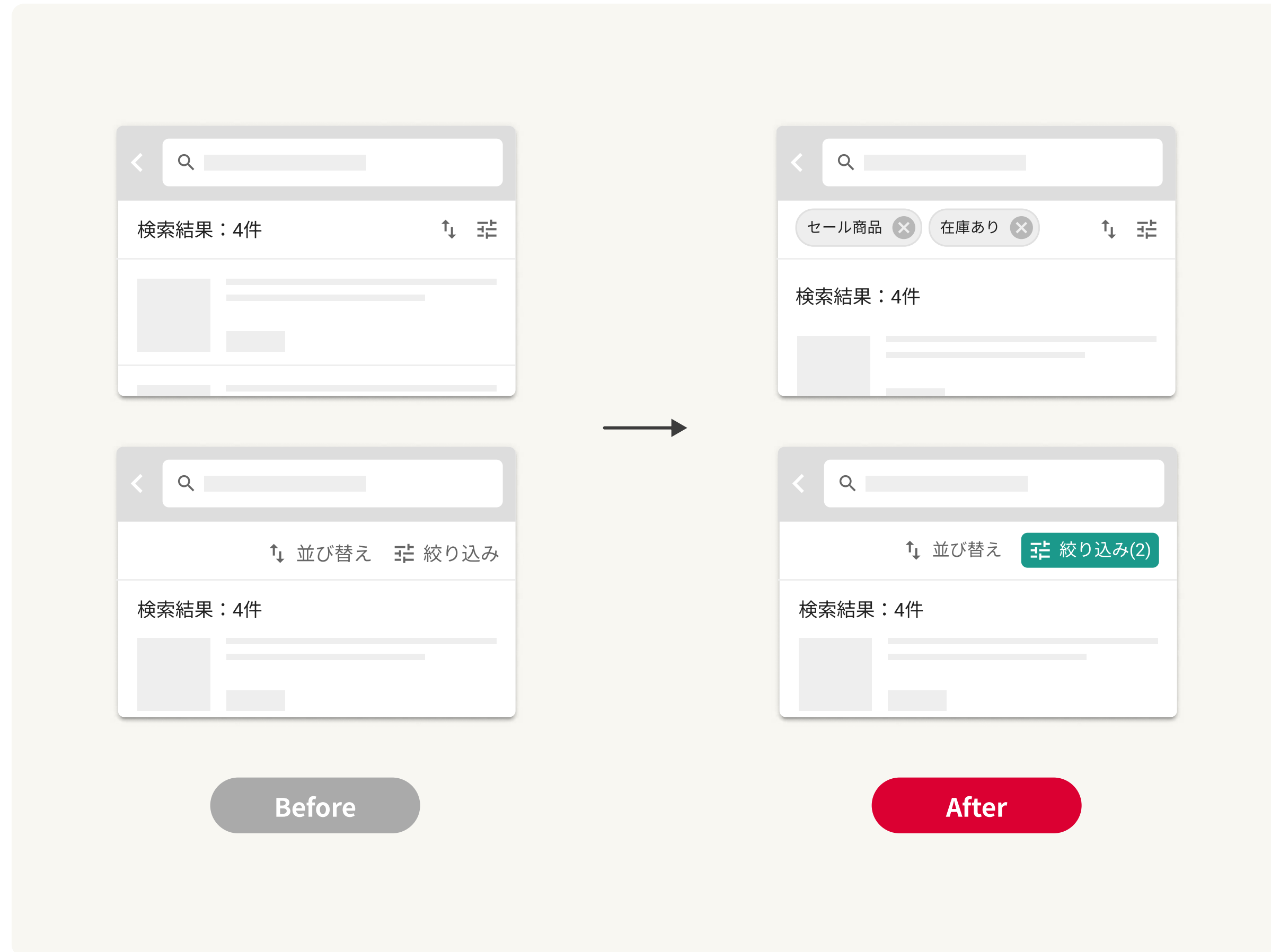
ユーザーは自分の現在地や状態がわからなくなると混乱し、元の状態に戻ろうとしてブラウザバックをしたりしますが、思っていた画面より一つ前の画面に移動してしまい不要な手間を要します。

モーダルなどでオーバーレイを行う際は、背景を透過してユーザーに現在の状態や位置を伝えましょう。

5

システムステータスの可視化

絞り込み中の状態を明示する



検索結果を表示する画面では、絞り込み条件が設定されているかどうかを明確に示しましょう。

絞り込み中の表現として

- 絞り込み条件の数を表示する
- 絞り込みボタンの色が変わる
- 絞り込み条件をChipsとして表示する
- 絞り込み条件をテキストとして表示する

などが考えられます。

エンプティステートを用意する



アプリやサイトを初めて利用した際に、まだデータやコンテンツが入っていない空の状態を”エンプティステート(Empty State)”と言います。

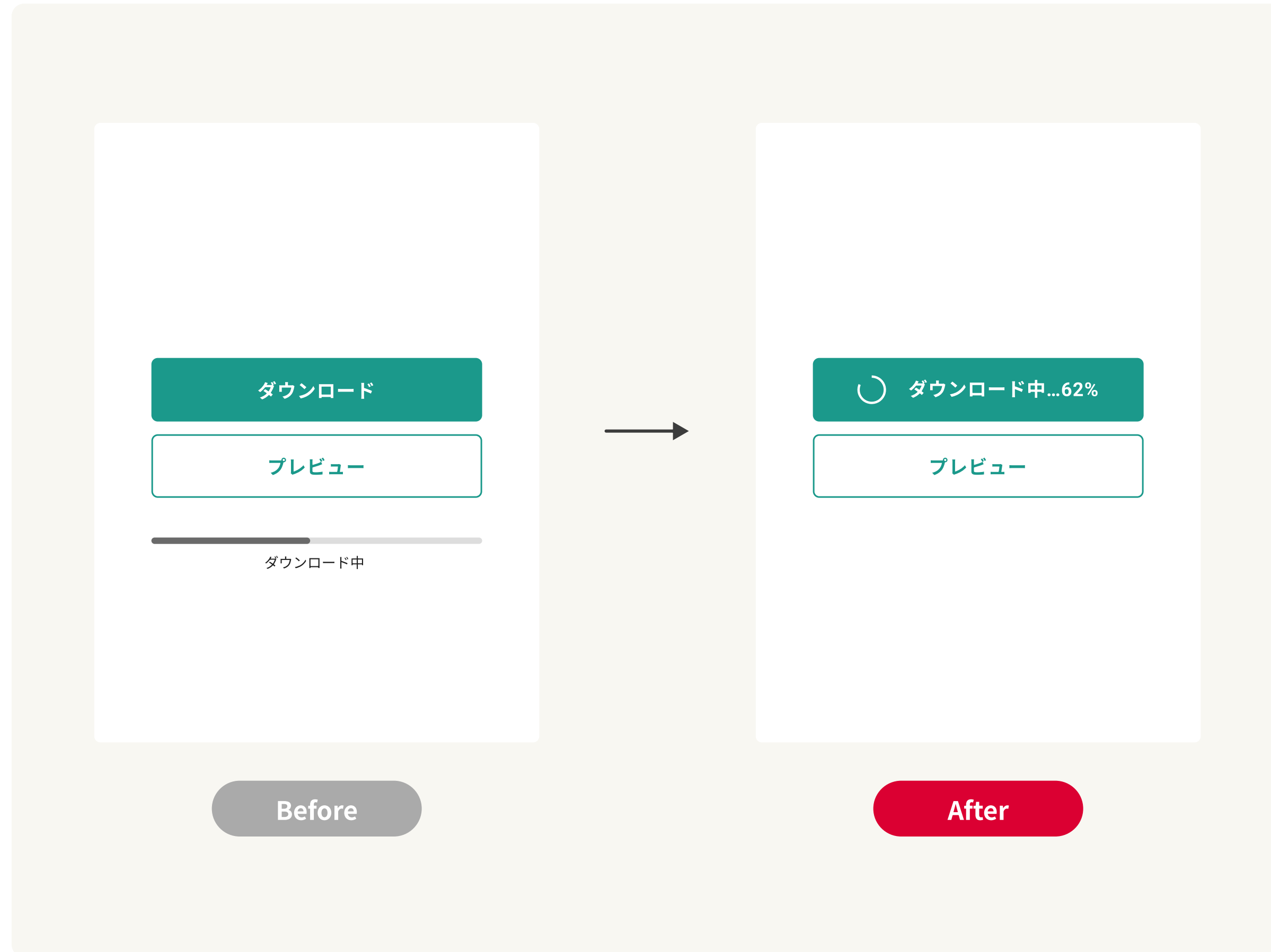
何も表示されない空の画面だけが存在していると、ユーザーは何をして良いかわからず、混乱してしまうでしょう。

この画面が何のための画面で、コンテンツが存在していない状態を明示し、どうすればコンテンツが表示されるのかをナビゲートしましょう。

7

システムステータスの可視化

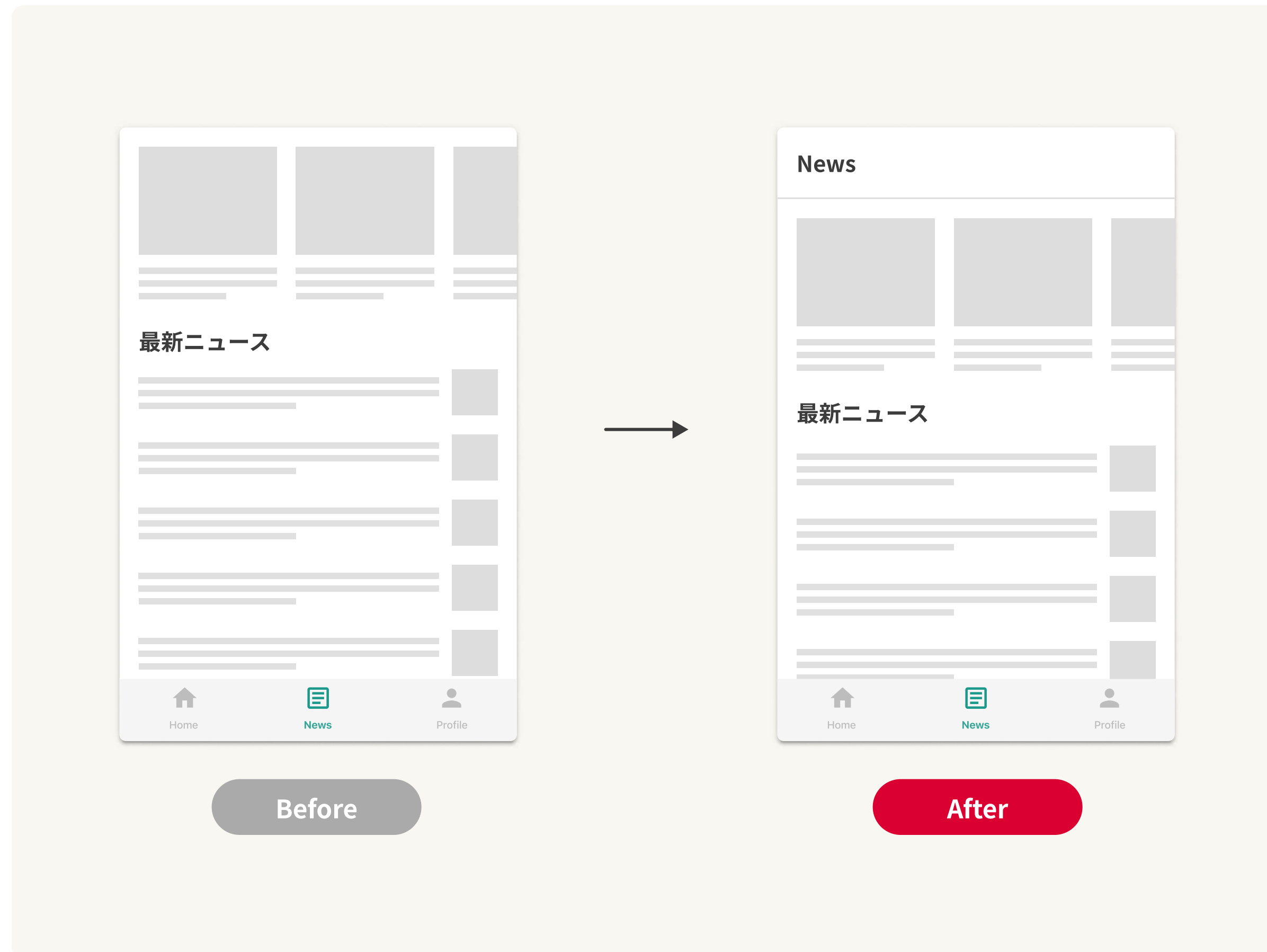
状態が変化するときにはアクションの近くで行う



アクションに対するフィードバックが離れた箇所に表示されるとユーザーに気づかれない可能性があります。

ダウンロードの開始やアップロードの完了のように状態が変化するときには、アクションの近くでフィードバックを表示しましょう。

サイト・アプリ上の現在地を把握できるようにする

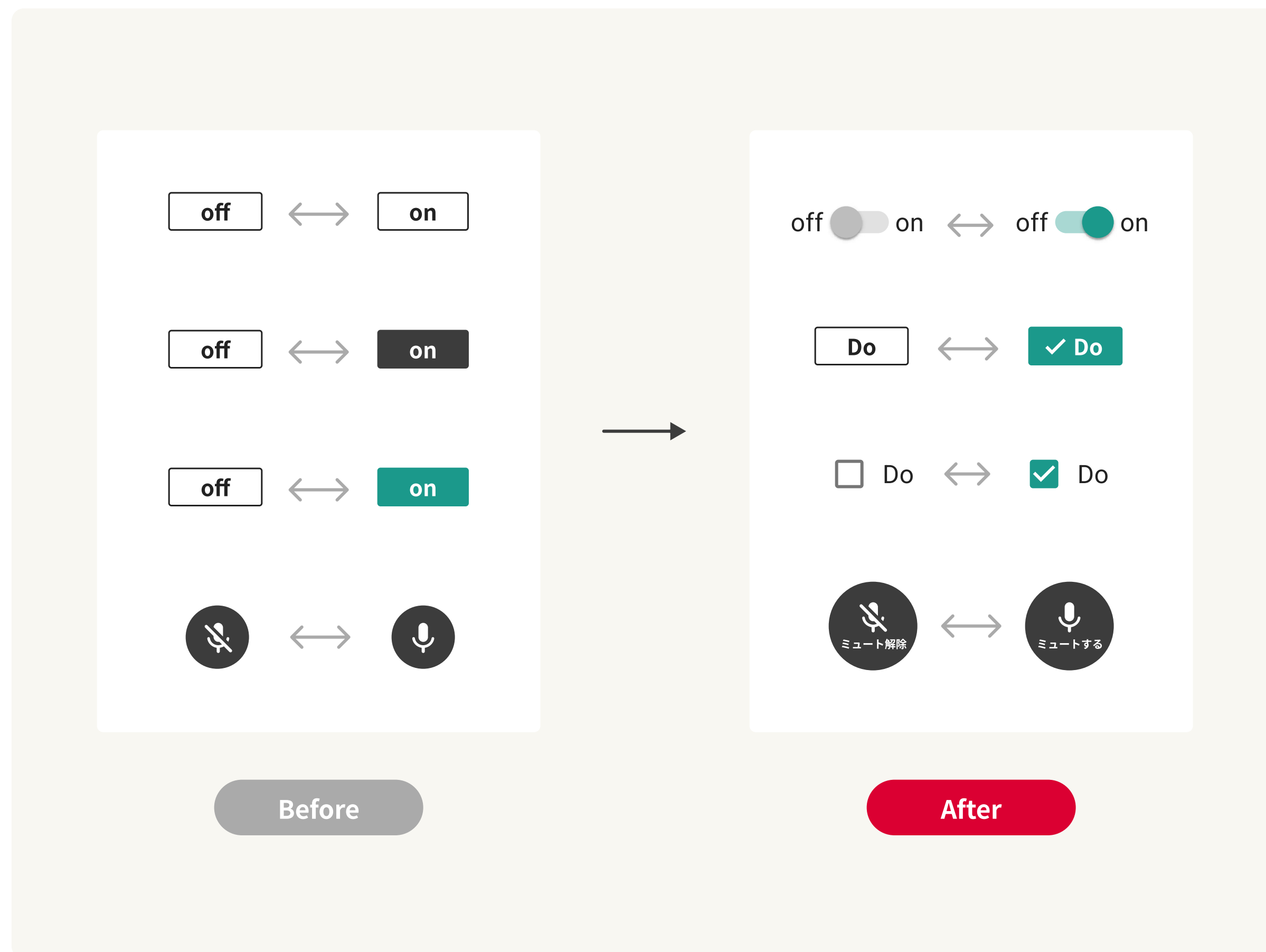


今いる位置がどこであるかや、ユーザーが何をすべき箇所なのかを常に明示する必要があります。

ユーザーはページ内で迷った時、現在地を知ろうとします。しかし、現在地を把握する手段がないと、目的の場所に行く前に意図しない画面遷移やエラーを引き起こす可能性が高まります。

タイトルやパンくずリストを設置するなどしてユーザーが現在地や状況を把握できるようにしましょう。

フリップ・フロップ問題を避ける



ひとつのボタンでオンとオフを切り替える場合、ラベルが現在の状態を表すのか、押した後の状態を表すのかわかりづらいことをフリップフロップ問題と言います。

デザインにおいて、ラベルが「現在の状態」を示しているのか「操作の効果」を示しているのかわからないことはよく起こります。

この問題を回避するには、オフ状態のデザインに要素をひとつ追加した表現をオン状態のデザインとします。

例えばチェックボックスの「レ」や、スイッチャーの「緑色」、アイコンボタンであればラベルで現在の状態を補足することでユーザーは自分の状態を把握することができます。

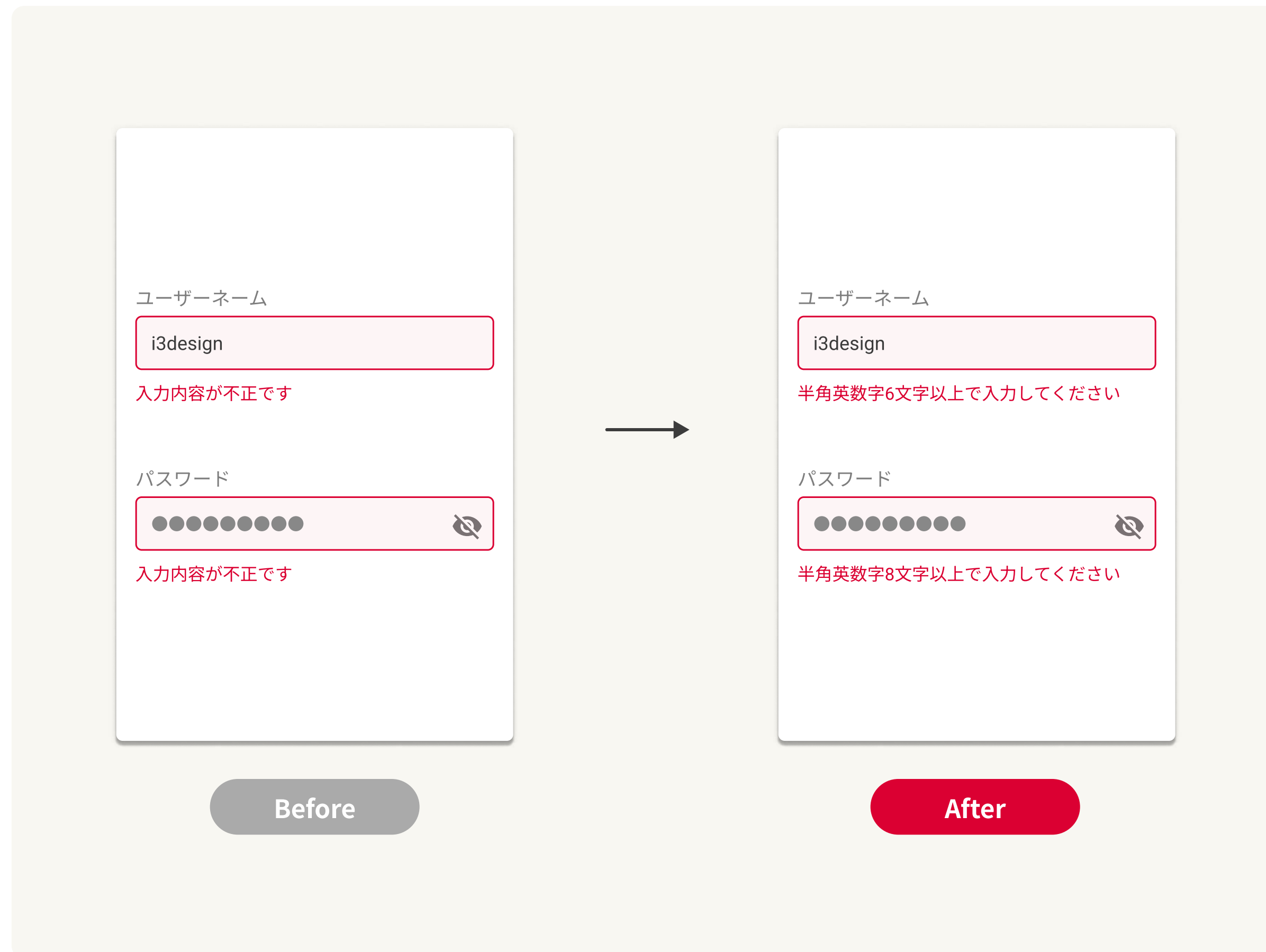
02

システムと現実世界の一致

Match between system and the real world



ユーザーが分かる言葉を使う

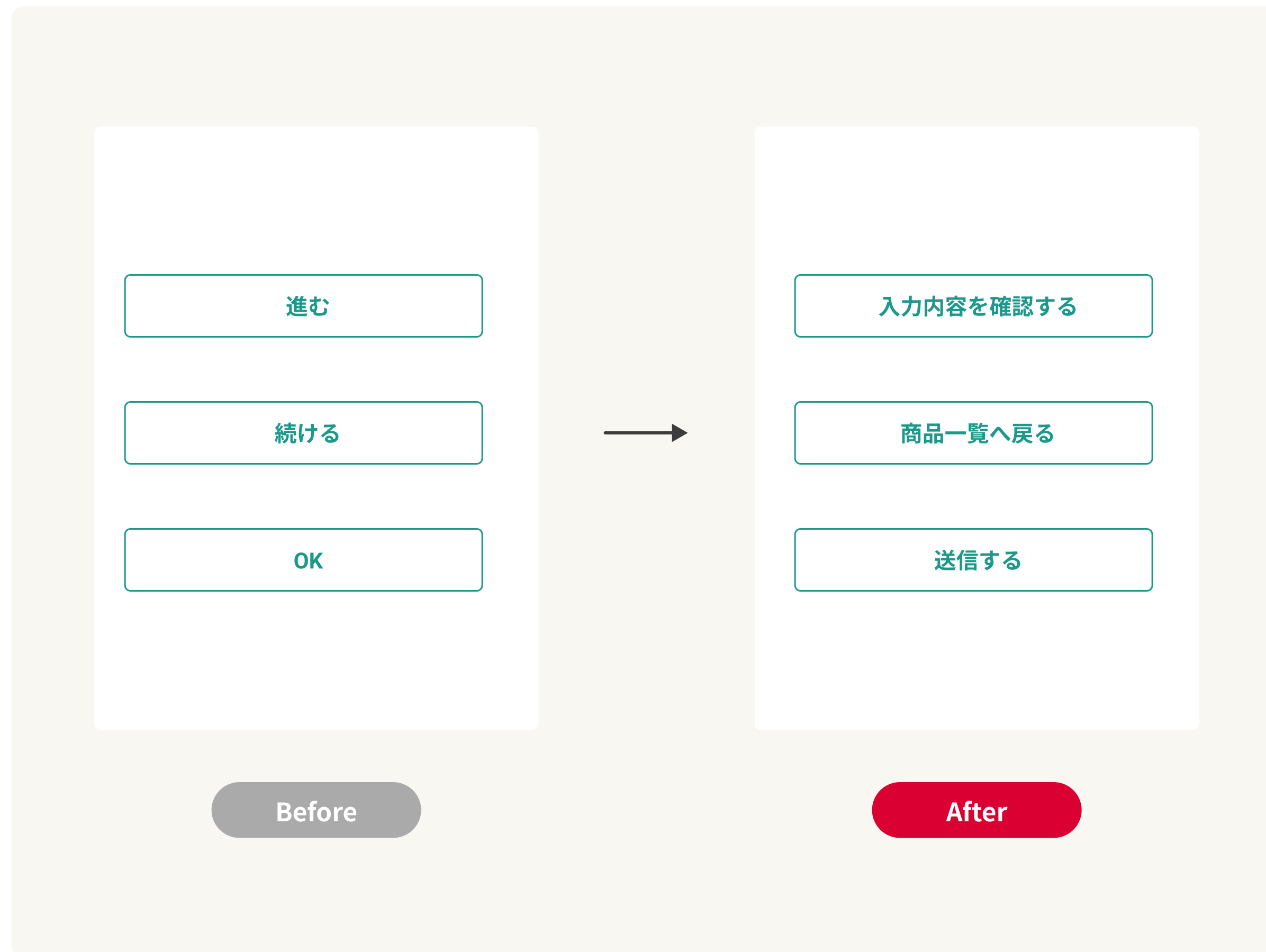


サービスやプロダクト内で使用する言葉は、プログラム目線ではなく、ユーザーが読んで分かる言葉を使いましょう。

「入力内容が不正です」「認証に失敗しました」「情報の取得に失敗しました」といった言葉はプログラム目線で書かれているためユーザーにとって有用な情報とは言えません。

「半角英数字6文字以上で入力してください」「パスワードは半角英数字8文字以上で入力してください」のように、ユーザーが見て内容が理解できる言葉を使いましょう。

ボタン/リンクの文言は「押すと何が起きるのか」を明示する

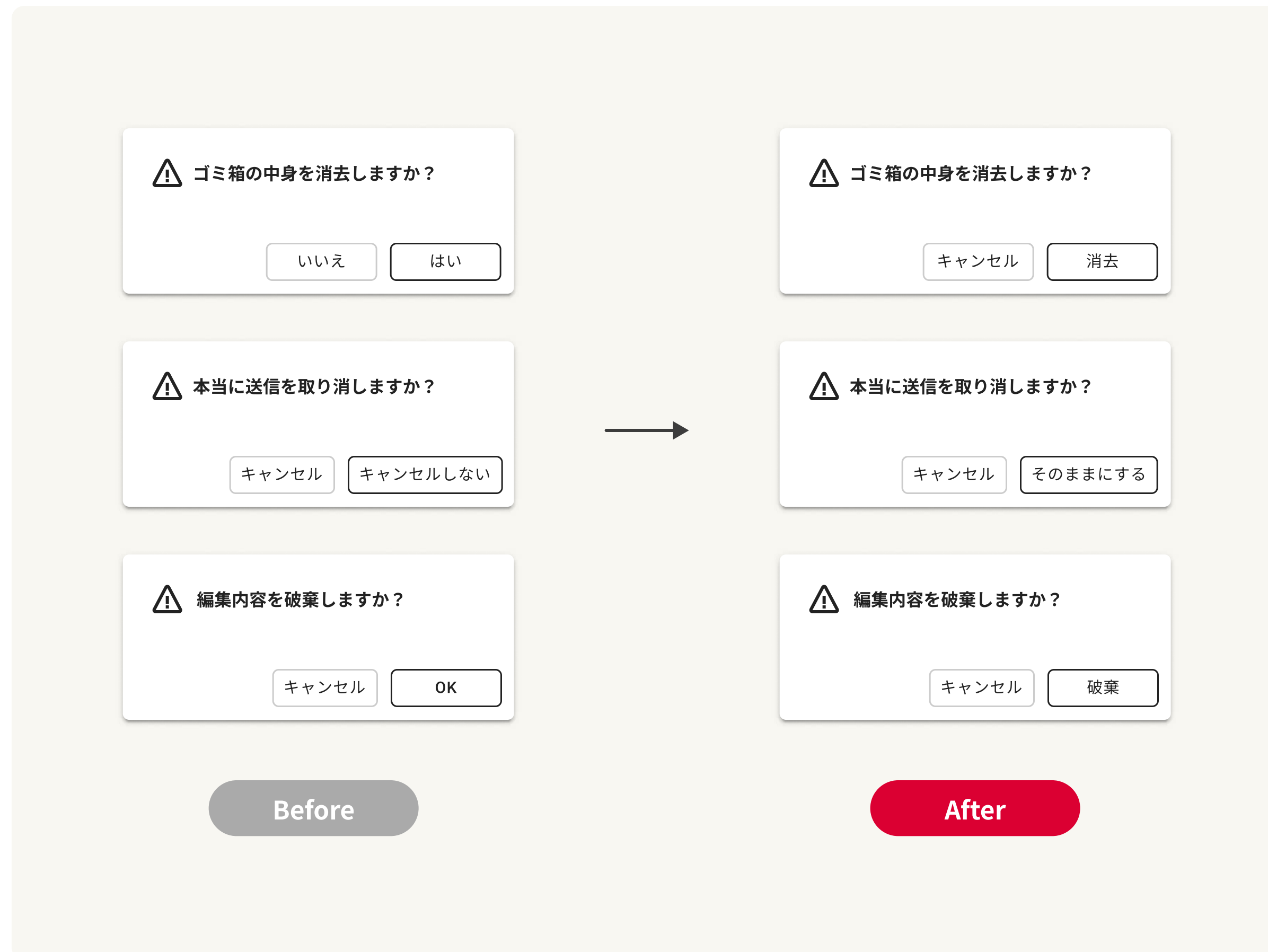


「進む」「続ける」「OK」のような文言は、そのボタンを押した後に何が起きるのか明示的ではありません。

ボタンを押すとどうなるか想起できないため、ユーザーがボタンを押すことを躊躇してしまう可能性があります。

「入力内容を確認する」「送信する」のように実際にボタンを押すことで起こるアクションを文言に使用し、ユーザーにボタンを押すと何が起きるのかを明示的にしましょう。

ボタンの文言を動詞的表現にする



モーダルなどのボタンのラベルは「OK」や「はい」などではなく、「保存」や「消去」のようにユーザーにアクションを促し、そのボタンを押すことで行われるアクションを表すラベルを用います。

また、「～をキャンセルしてもよろしいでしょうか？」のように、モーダルの説明文に「キャンセル」という言葉は使わないようにしましょう。キャンセルは否定的ボタンによく使われるため、混同するような問いかけは避けましょう。

日付と分かる表記にする

2020-10-10

2020.5.16

2020_7_12

2020/9/2

Before



2020年10月10日

2020年5月16日 (金)

7月12日 (金) 19:00

2020/09/02

After

数字を見た時にそれが日付であるということが即座に理解できる必要があります。

「-」 (ハイフン) や「.」 (ピリオド)、「_」 (アンダーバー)などは、日付を表す記号として一般的でないため避けましょう。

また、業務システムなどで一覧として日付表記が複数並ぶ場合には、ゼロパディング (数字の頭にゼロを付けること) を用い、桁数によって並びがずれないようにしましょう。

03

ユーザーのコントロールと自由度

User control and freedom



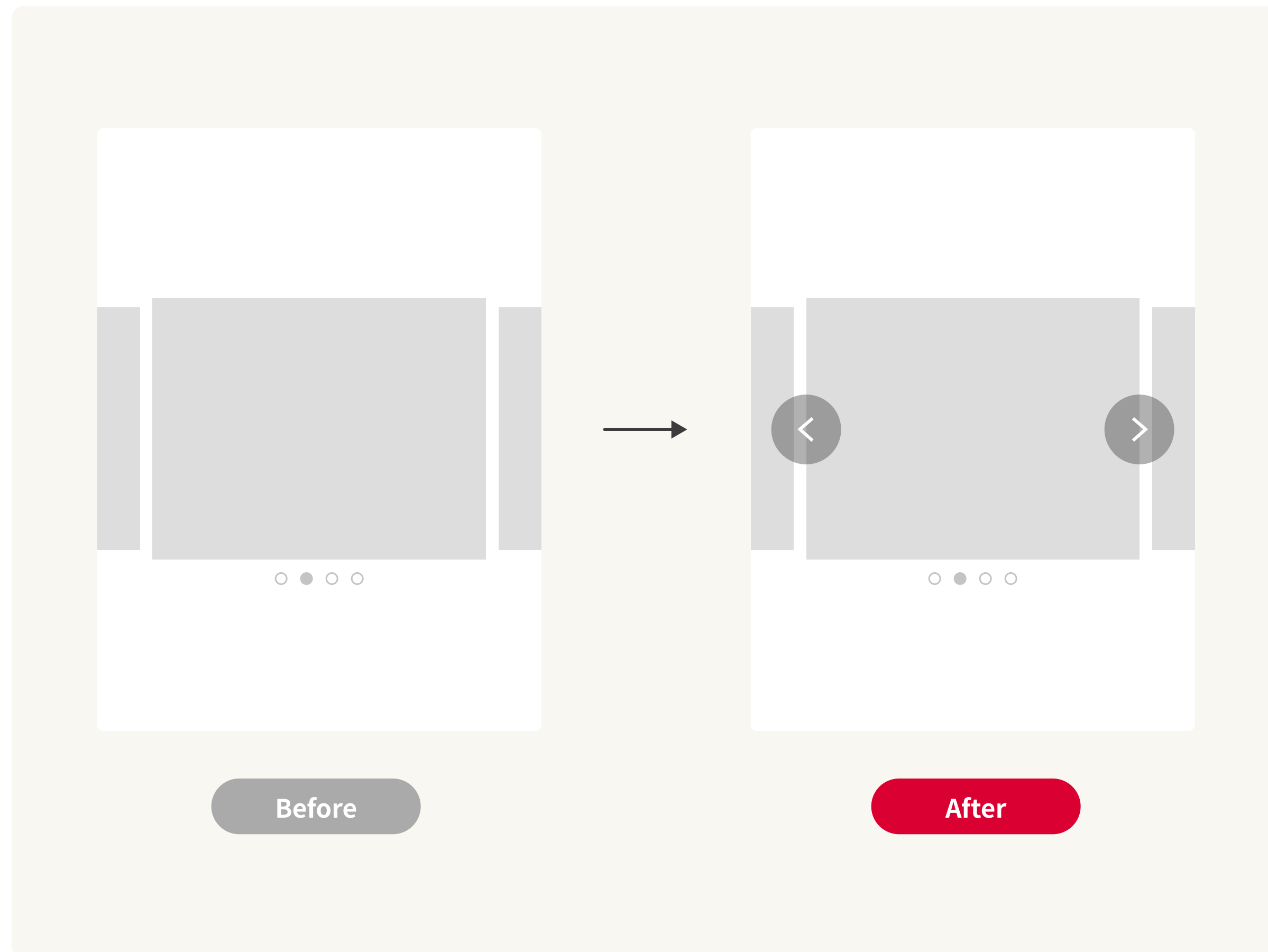
常にユーザーがコントロールできるようにする



ユーザーは、間違えたり、気が変わったり、興味本位でボタンを押したりします。

そのためシステムは、ユーザーの間違いをすばやく修正したり、行った選択をやり直せるようにして、どのタイミングでもシステムをコントロールできるようにしましょう。

カルーセルはドットインジケータだけでなくボタンをつける



ドットインジケータは現在表示しているコンテンツ位置を示すには有効的ですが、小さくクリックしづらいためコンテンツの切り替えにはあまり向いていません。

カルーセルは、ドットインジケータだけでなく、コンテンツをスライドするためのボタンをつけましょう。

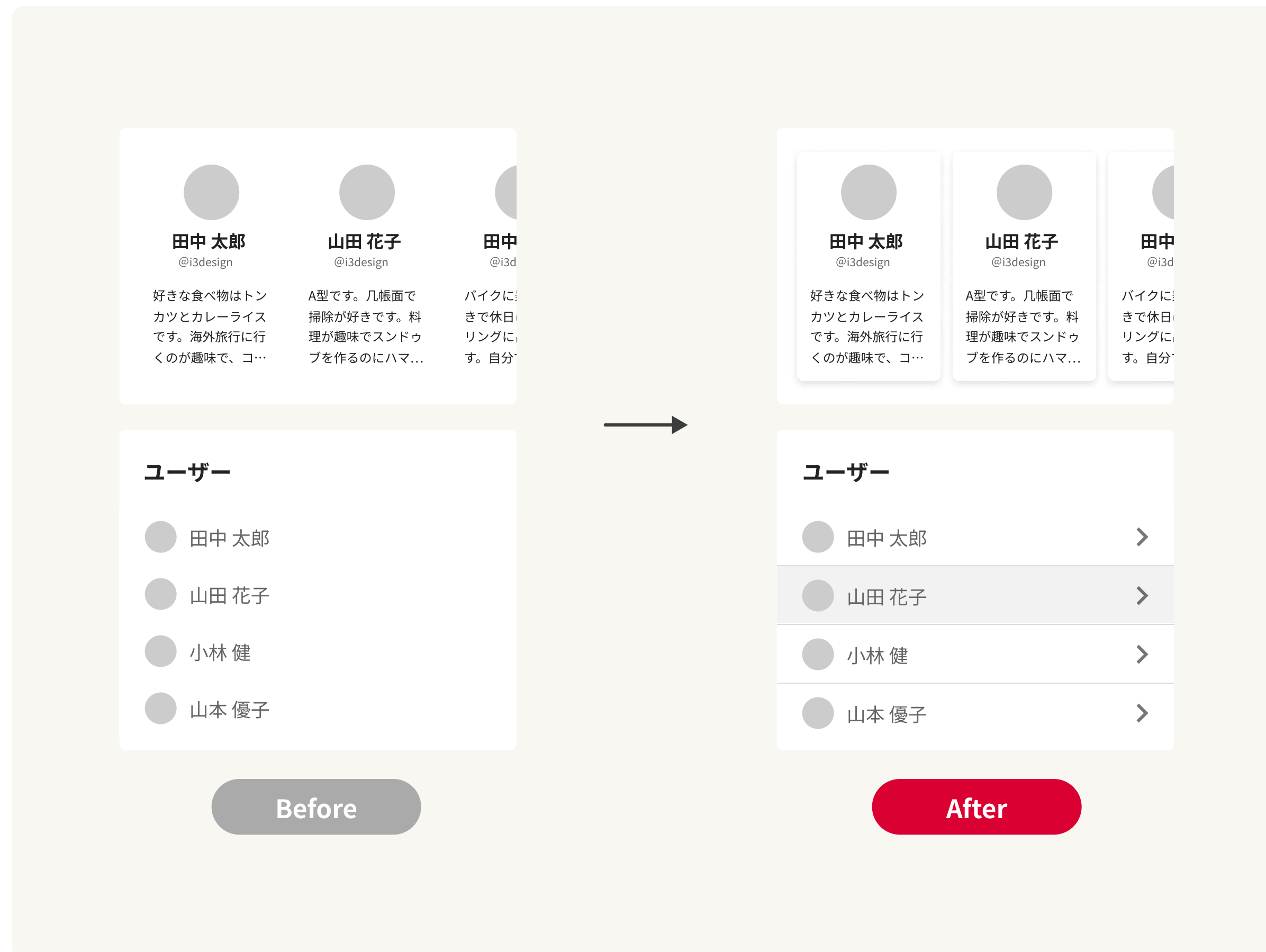
スクロールの先を示唆する



ユーザーにスクロールして見てもらいたい場合、コンテンツの切れ目が画面の端と揃っているとユーザーはスクロールできることに気づかない恐れがあります。

画像や文字などのコンテンツの位置を調整することで、続きがありそうに見せ、その方向にスクロールができることを示唆します。

カード/リストはクリックできることを示唆する



カードやリストもボタンやリンクと同様にクリックできることを示唆しましょう。

クリックできるかどうかを確認するためにクリックさせたりマウスホバーさせることは望ましくありません。

シャドウをつけて奥行きをだしたり、それぞれのリスト項目に矢印アイコンを足すなどして遷移先があるように見えるデザインにしましょう。

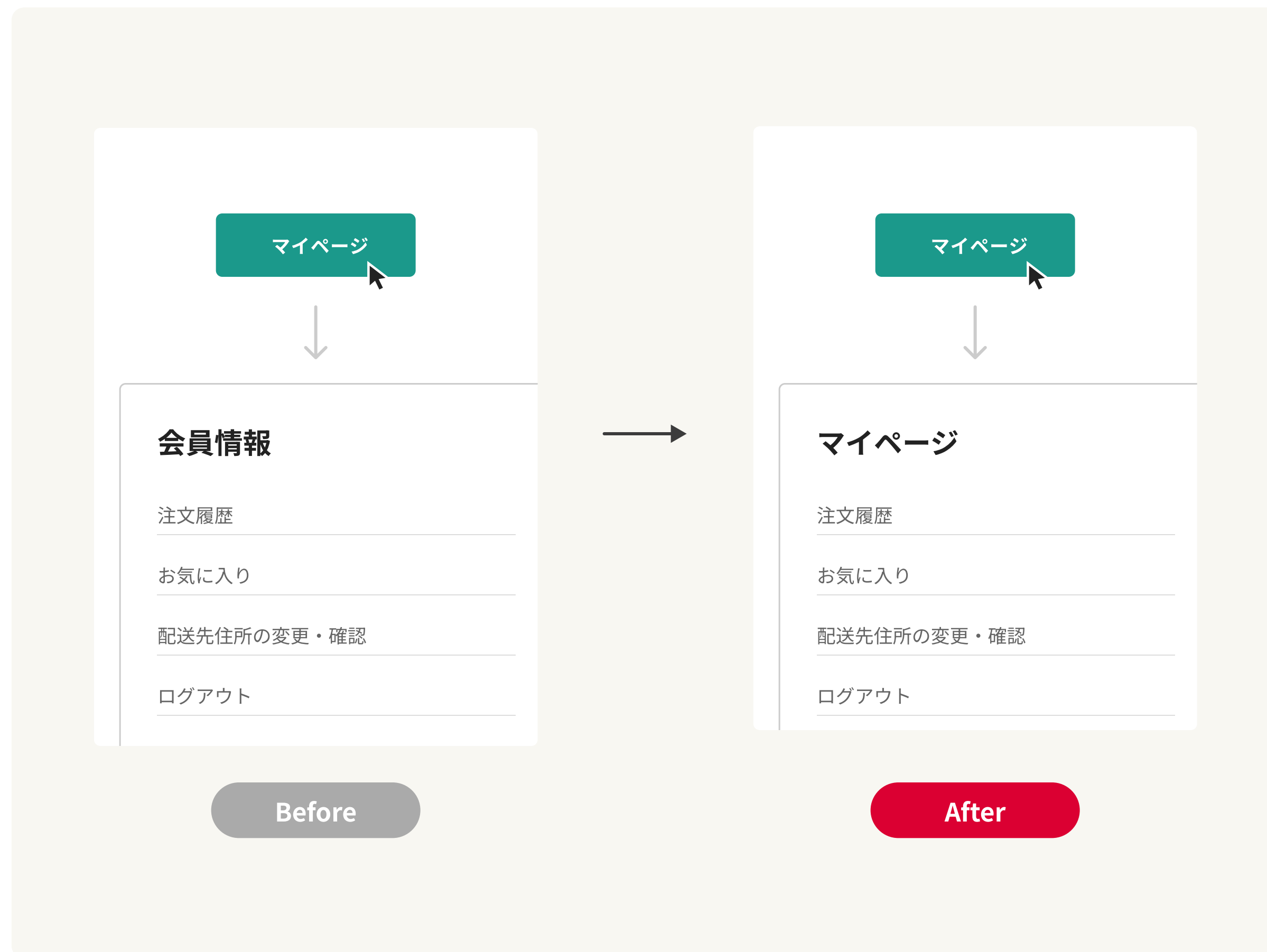
04

一貫性と標準化

Consistency and standards



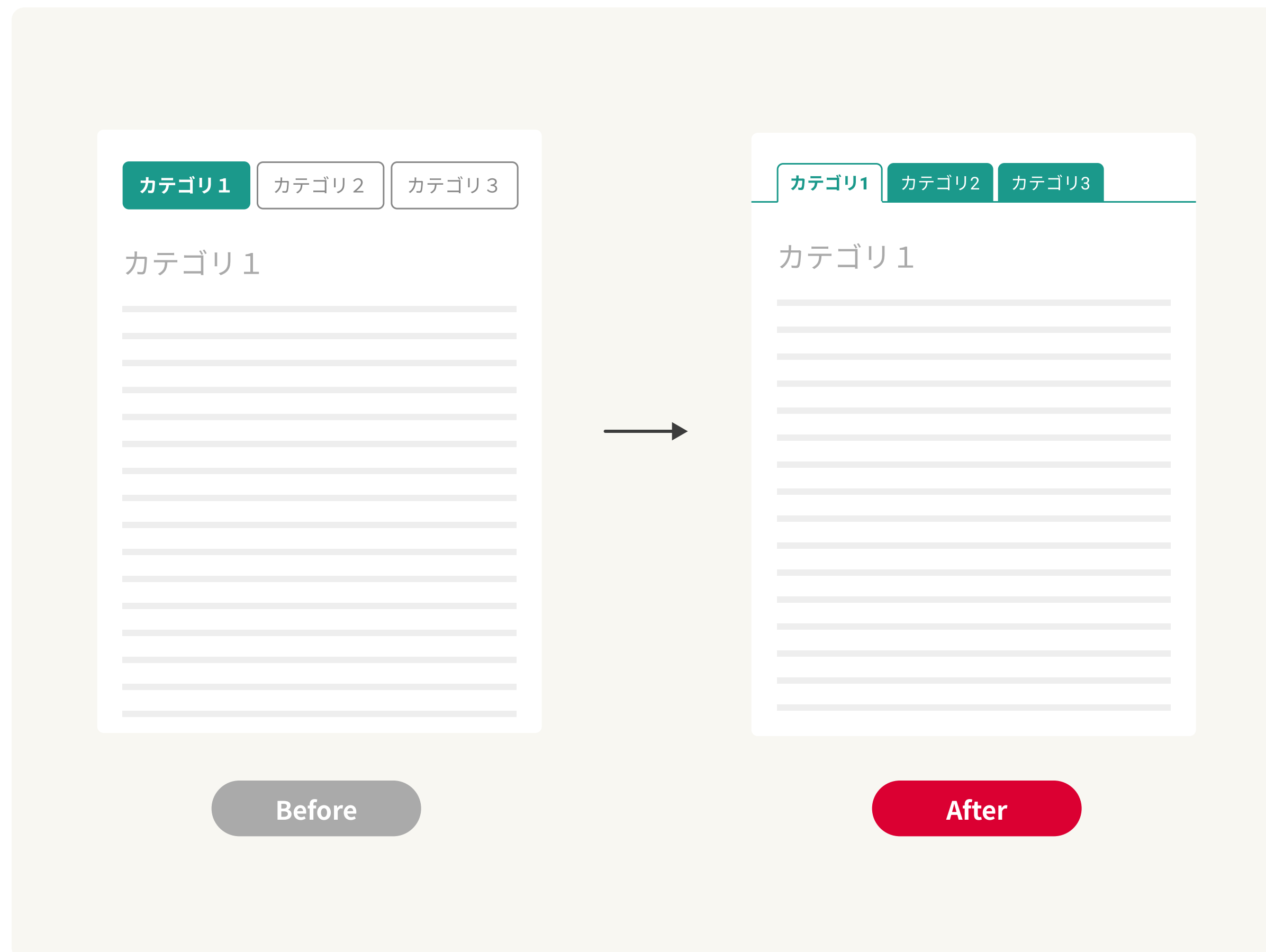
遷移後のページのタイトルを揃える



「マイページ」ボタンを押したはずが、「会員情報」とタイトルのついた画面に遷移するとユーザーは「会員情報ボタンを間違って押したかな？」と誤解するかもしれません。

ボタンの文言と遷移後のページのタイトルを揃えてユーザーが違和感なく使えるように心がけましょう。

タブだとわかる見た目にする



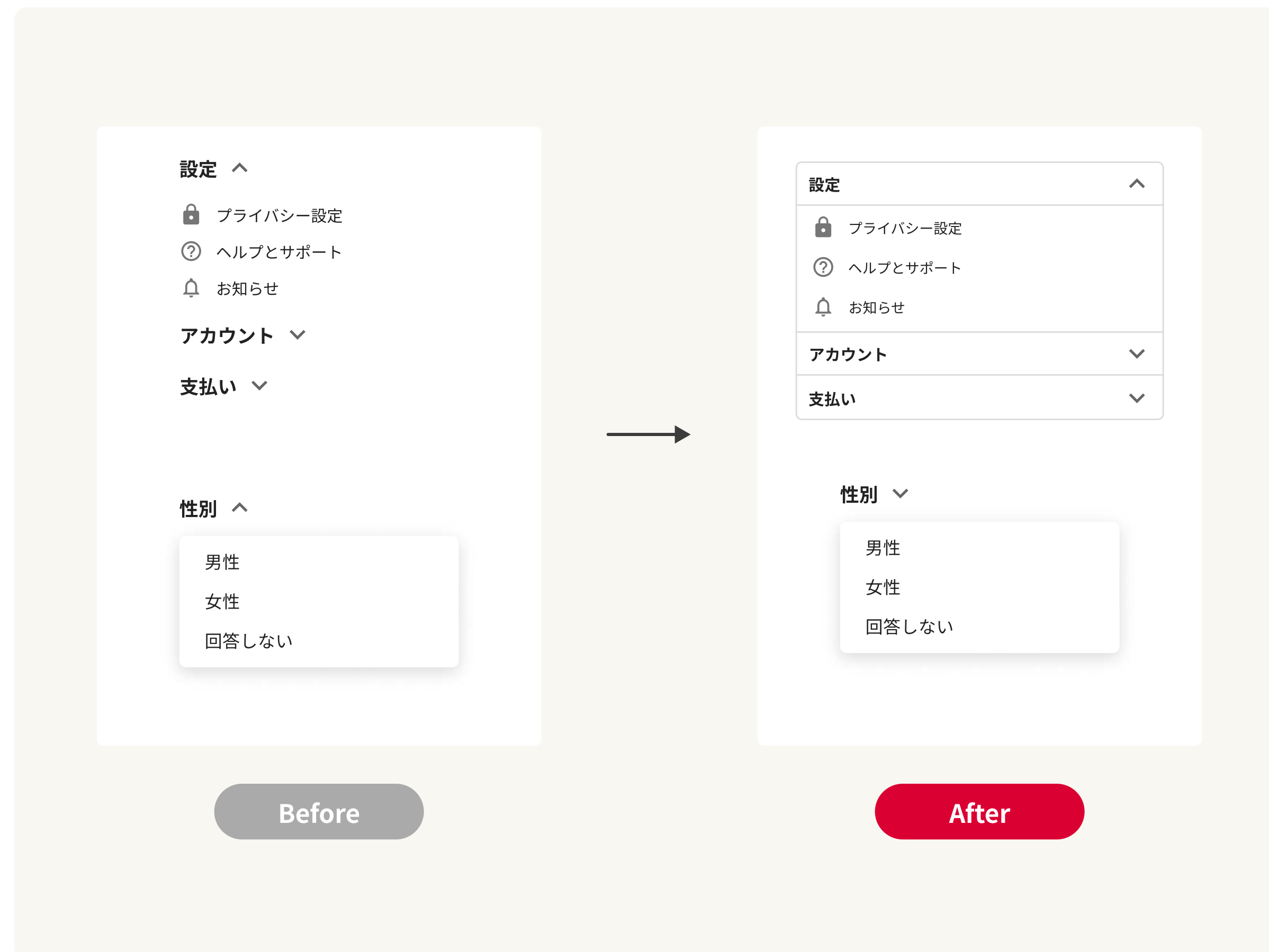
本来タブとはファイルや手帳、ノートなどの端に付け、見たいページを開きやすくするための出っ張った部分を指します。ここから由来して、Webサイトでも「タブ」と呼ばれるパーツがあります。

現在では様々タブの表現があり、中にはタブと分からない表現が使用されていることがあります。

また、タブをデザインする上では改めて以下の点に注意してみましょう。

- ・タブメニューとコンテンツエリアが明確な領域で区切られている
- ・タブメニューが並列である
- ・Chipsのような同時に複数選択が可能な見た目にしない

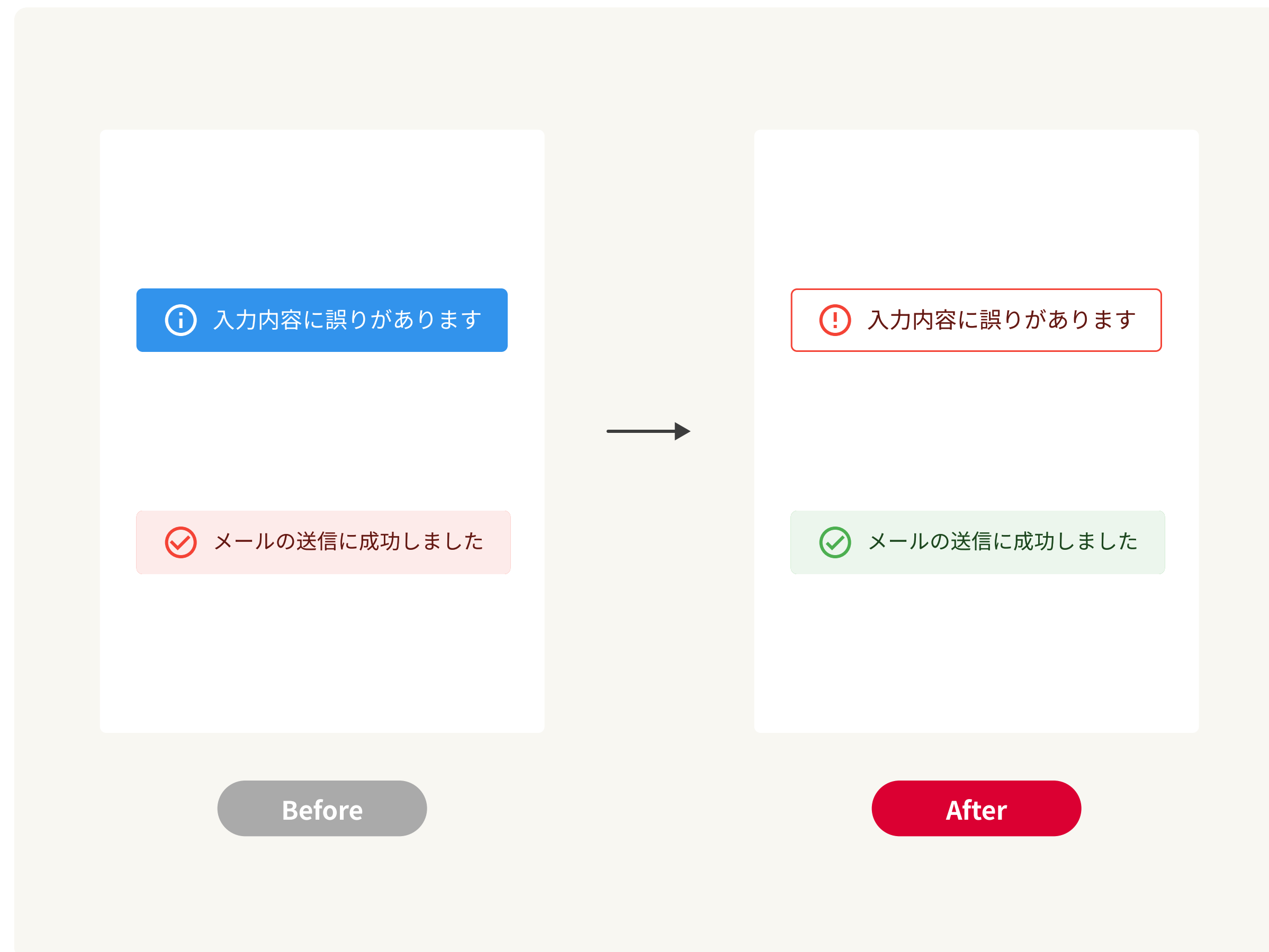
機能が違うものを同じ見た目にしない



ユーザーは1度登場した要素は次も同じ挙動をするはずであると想起するので、要素の見た目が同じで異なる挙動が起きるUIは操作ミスに直結します。

例えば、select要素とアコーディオンメニューは似た外観のデザインが採用されることがありますが、操作前から機能の違いを識別できるように見た目に十分な差をつけておく必要があります。

機能と色を揃える



視覚情報のうち、80%以上が「色彩の情報」と言われています。色を与える影響は大きく、配色1つで好印象にも悪印象にもなり得ます。多少の個人差はあれど、人の持っている色に対してイメージはほぼ同じであるでしょう。例えば、赤は「危険・禁止・エラー」であったり、緑は「安心・正常」などを連想します。

デザインにおいても実際の機能と色が異なってはいけません。「削除」ボタンのような取り返しがつかないようなアクションの色は赤を使うことでユーザーに危険性を訴えています。

リンクとボタンを使い分ける



リンクとボタンは機能や振る舞いが似ているため、混同して使われているサイトやアプリケーションをよく見かけます。

ボタンのような見た目をしていると「押せる」と思わせてしまうため、リンクなのかボタンなのか区別できません。

また、視覚的に「これはリンクだろう」「リンクかもしれない」と思ったものが、ページ遷移しなかったり、テキストが直接選択できないと、ユーザーは「リンクかと思ったけど、これはボタンだったのか」と学習し直す必要があります。

以下の使い分けをしましょう。

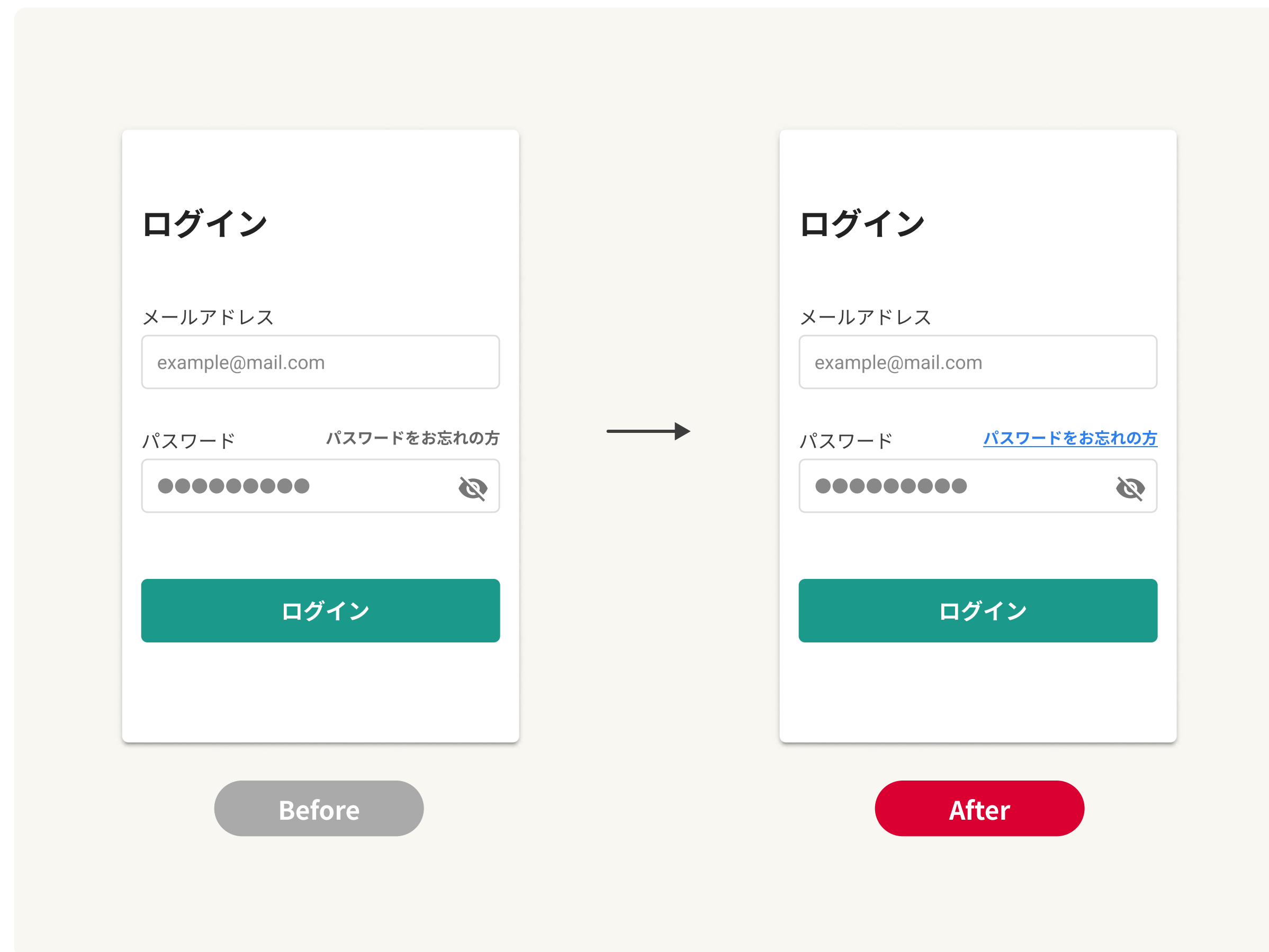
【リンク】

ユーザーが対象に影響を与えないアクション、画面遷移

【ボタン】

ユーザーが対象のバックエンドまたはフロントエンドに影響を与えるアクション、データの操作

テキストリンクであることを下線と色で明示する



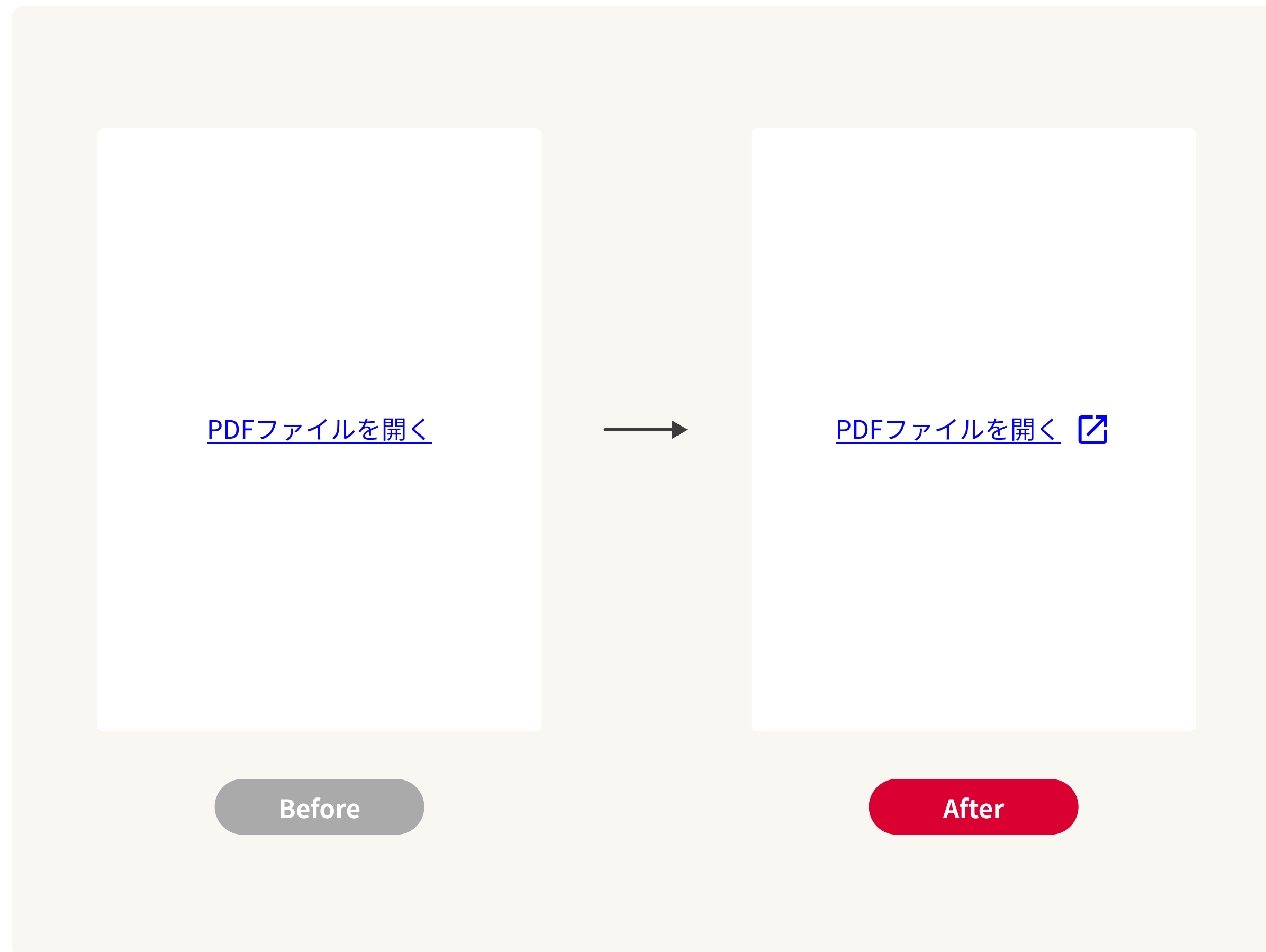
テキストがクリックできるかどうかはすぐわかるようなデザインにする必要があります。

リンクかどうかを確認するために、テキストをクリックさせたりマウスホバーさせることは望ましくありません。

基本的にテキストリンクには色をつけて、アンダーラインを引きましょう。

色覚障害を持っているユーザーも他の色と区別することができる青色を採用することが多いですが、他の色もリンクが本文と明確に区別することができれば使うことができます。

外部リンクや新規タブを開くリンクにはアイコンをつける



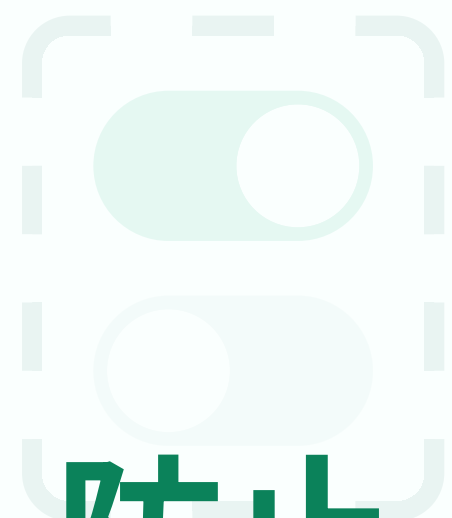
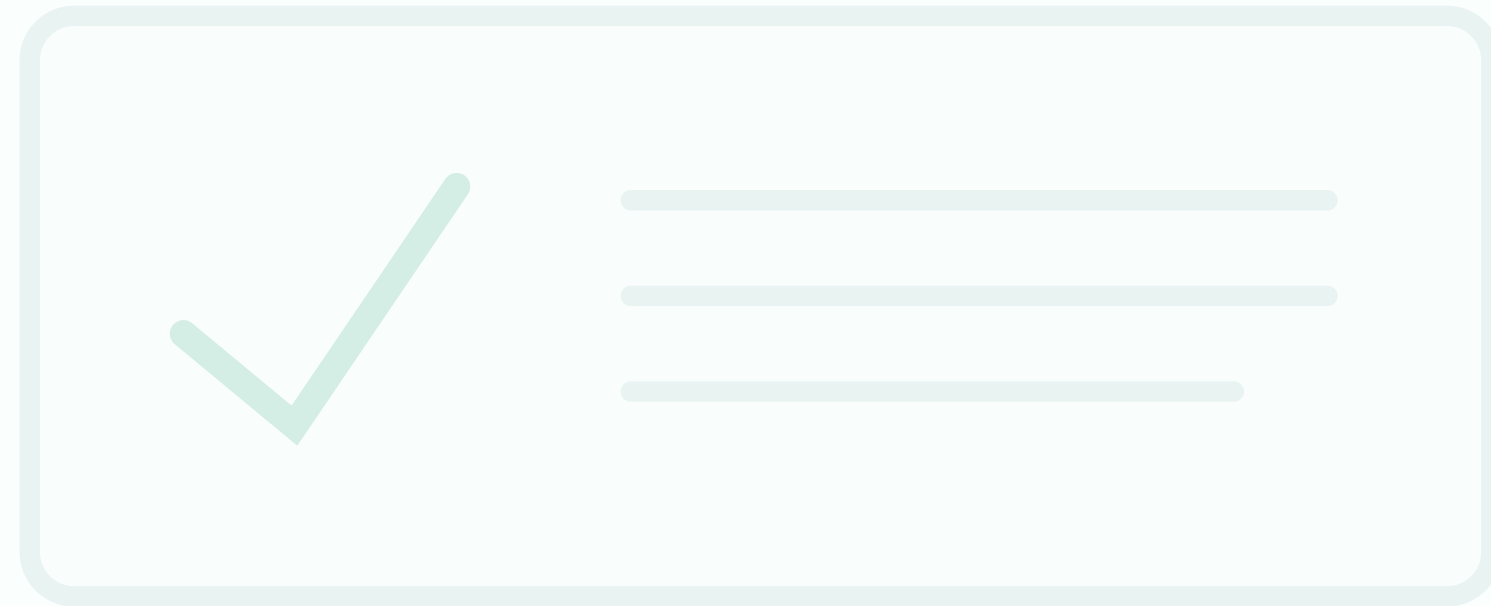
一般的に、サイトを利用する時、ユーザーは今自分が訪れているサイト内で回遊していると考えています。リンクをクリックした時に、アナウンスなしで外部サイトに遷移したり新規タブが開くと、そのまま同サイト内を遷移したと誤解を招いたり、急に遷移することに不審を抱く可能性もあります。

外部リンクにはアイコンを付けることで通常のリンクと明示的に区別することが重要です。

05

エラー防止

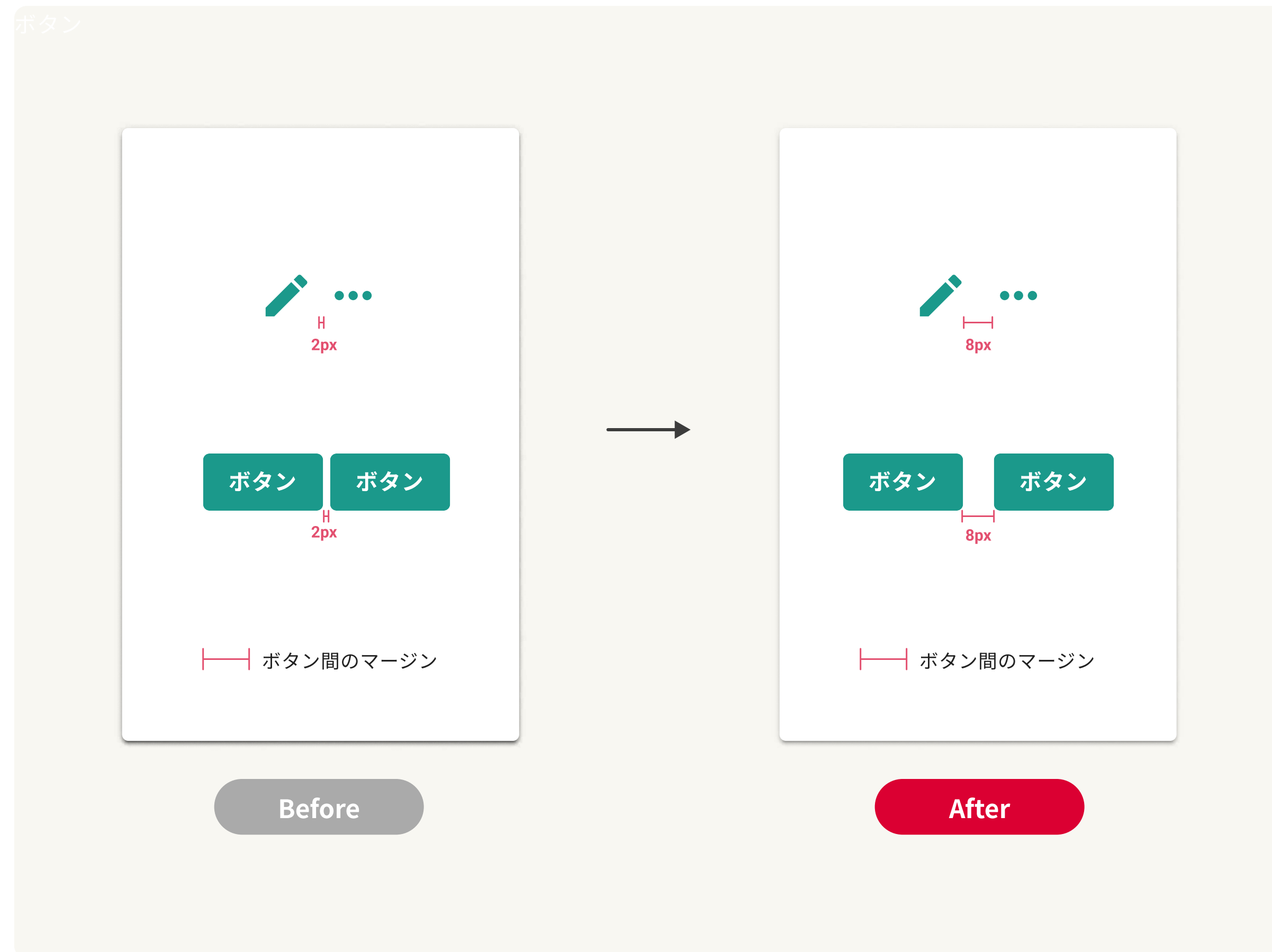
Error prevention



25

エラー防止

リンクアクションは誤タップしないよう十分な間隔を保つ



ボタンサイズが大きい場合はクリックエリアも広いため押しやすいですが、サイズの小さなボタンが並んでいる場合は、ボタンの間隔が近く、意図していないボタンを誤って押してしまう可能性が高くなります。

サイズに合わせてボタン同士の間隔を調節して、押しやすく、誤タップの起きにくいボタンの間隔を保ちましょう。

取り返しがつかないアクションは遠くに置く



ある調査では、「データ損失被害の30%が人的なミスによるもの」だと言われています。削除するつもりがなくても、他のコンテンツとの差異がなく、気づかないうちにボタンを押してしまうことも十分に考えられます。

「削除」などの取り返しのつかないアクションは他のコンテンツと同じ場所に置くことは避けましょう。

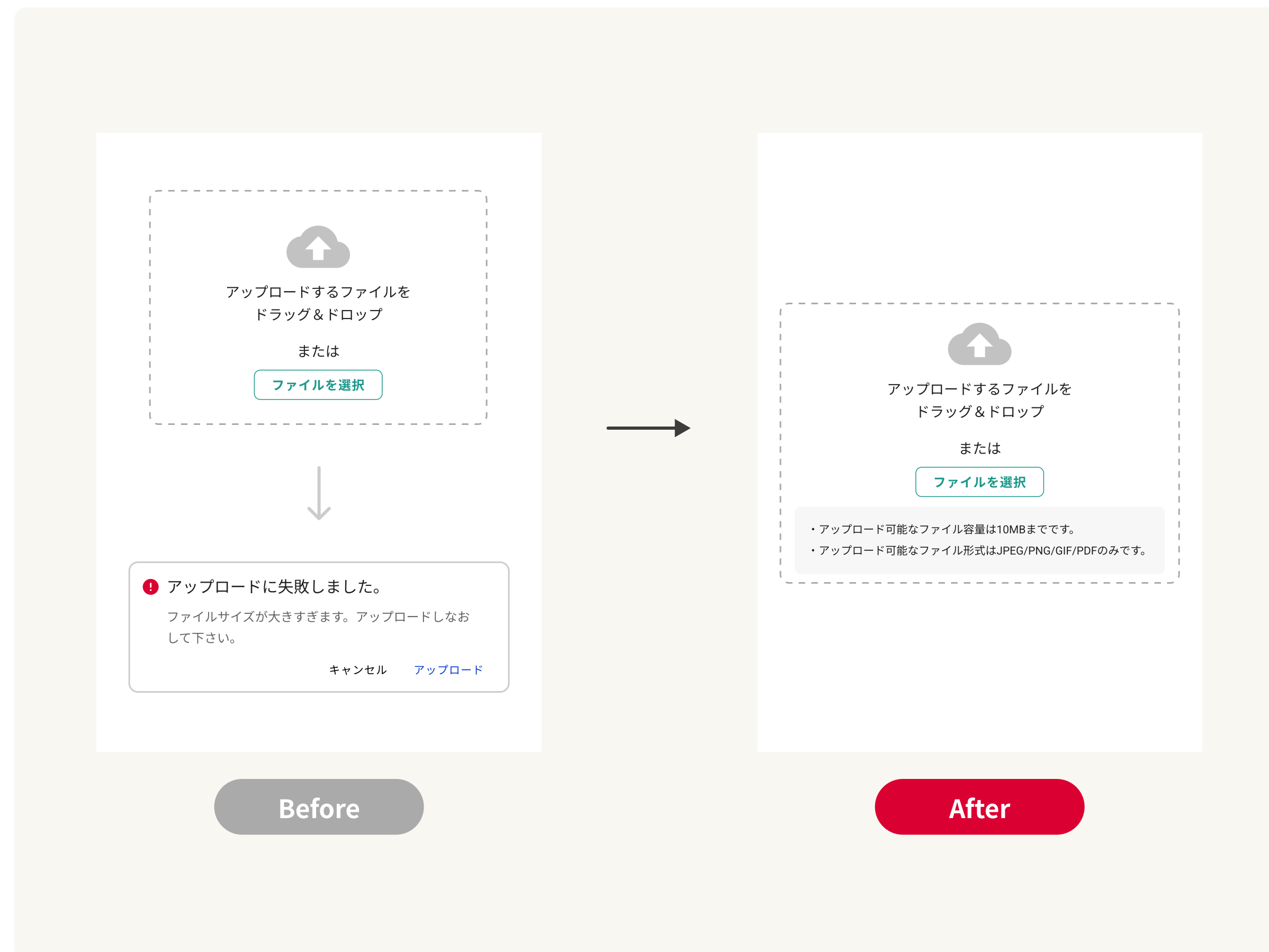
タップ/クリック可能なエリアを広く取る



ユーザーは私たちが思っているほど正確にタップ/クリックしません。押したと思って押せていないことや、本来押せるはずのエリアが反応しないとユーザーはストレスを感じます。

表示されている要素よりも広いエリアをタップ/クリック可能にしましょう。

条件は先に提示する



誰しも、自分のした作業が無駄になることは嫌なはずですが、システムでもそのようなことが起こります。例えばパスワードを決める時の条件や、アップロードするファイルのサイズ制限など、アクションを起こした後にエラーを指摘された経験はないでしょうか？

システム側の条件を後出しでユーザーに提示するのはユーザーにとって非効率な体験となってしまいます。事前に確認しておくべき条件はできる限り早い段階で提示し、ユーザーの作業が無駄になるようなことは避けましょう。

スクロール時に重要な情報が隠れないようにする

Before

史	(不明)	田中 太郎	-	-
るみ	契約中	篠山 達士	2019/10/11~2021/12/15	7,150円
	検討中	小幡 有起哉	2018/11/14~2022/02/13	1249,800円
功	契約不可	北田 由宇	-	-
也	契約中	羽田 薫	2019/01/01~2022/04/24	121,300円
史	(不明)	堀口 康文	-	-

表示件数 5件 < 1 2 3 4 5 >

After

1541	黒川 進	(検討中)	有起哉	2018/11/14~2022/02/13	1249,800円
0131	志田 はるみ	契約中	達士	2019/10/11~2021/12/15	7,150円
1541	黒川 進	検討中	有起哉	2018/11/14~2022/02/13	1249,800円
4242	有賀 雅功	契約不可	由宇	-	-
5511	大野 将也	契約中	薫	2019/01/01~2022/04/24	121,300円
0394	山根 崇史	(不明)	康文	-	-

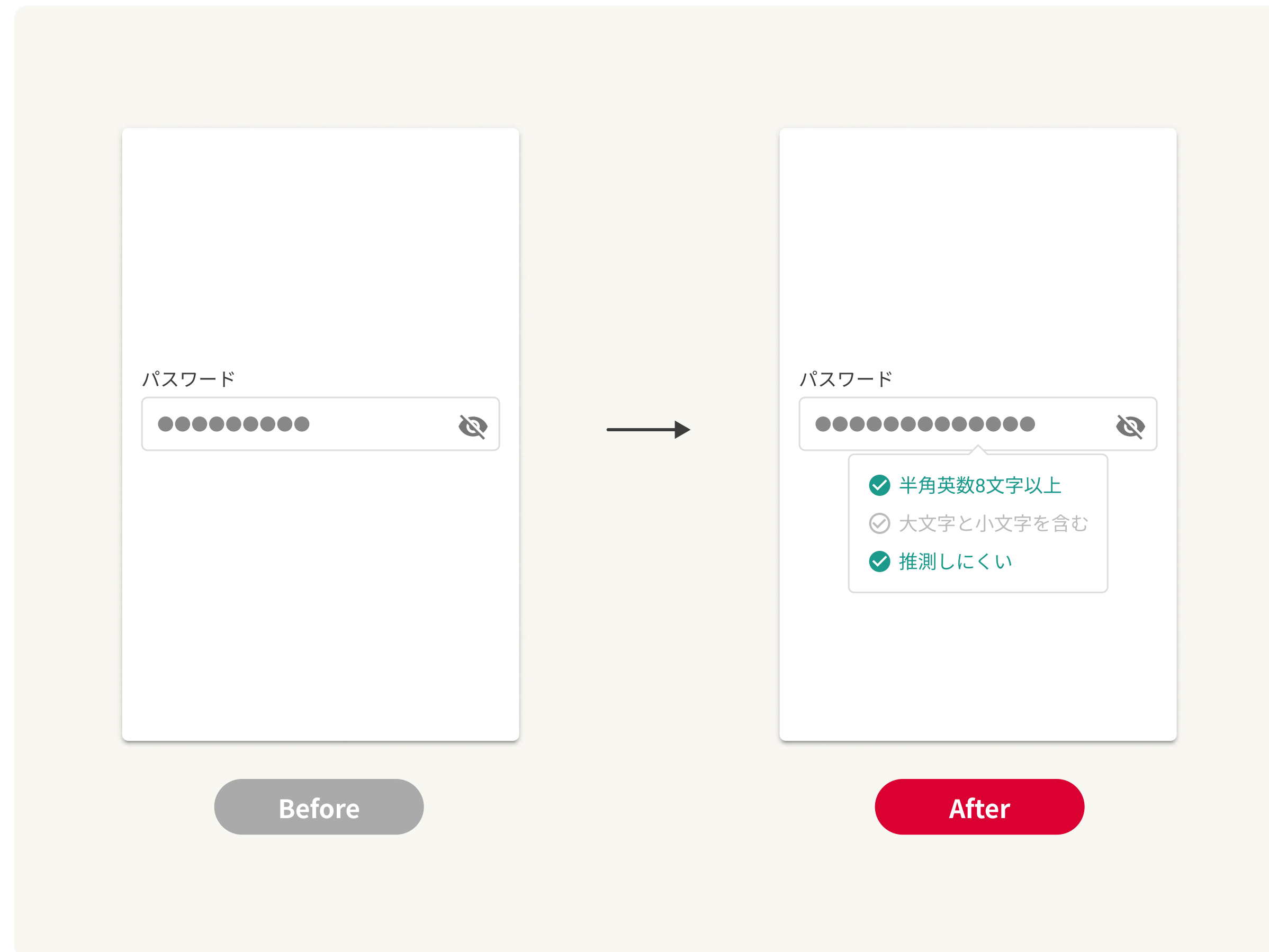
表示件数 5件 < 1 2 3 4 5 >

スクロール前提のテーブルやリストなどは、スクロールした状態で重要な情報が隠れていると、ユーザーエラーを引き起こす可能性があります。

ユーザーがデータを元に判断するとき間違えないために、今見ている要素が何の情報であることを常に確認する必要があります。

そのデータ固有の情報（ユニークであることを判別できる情報）が常に表示されるようにしましょう。

リアルタイムバリデーションで条件の達成状況を明示する

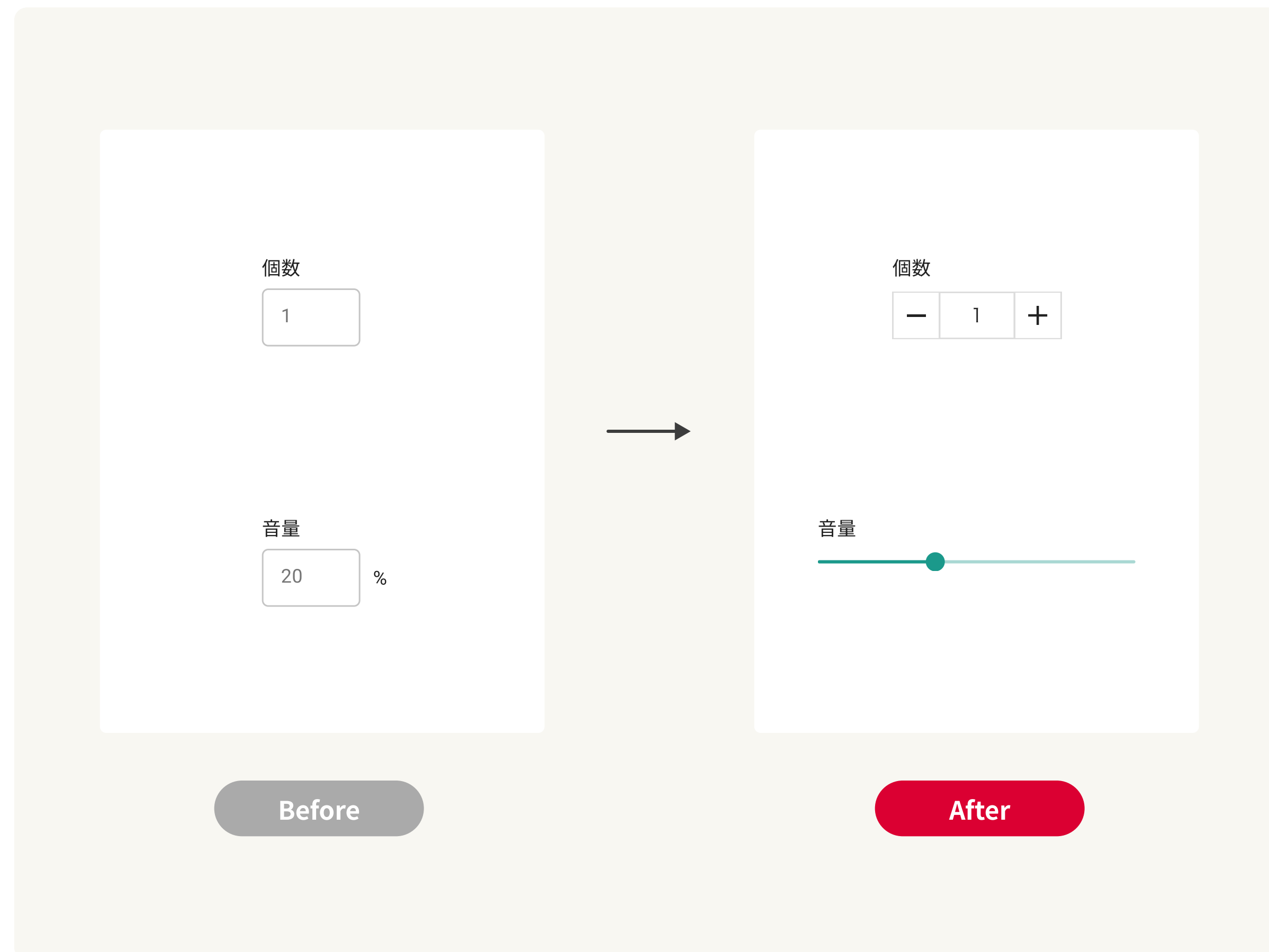


条件があるフォームを入力する時に、エラーを防ぐためのリアルタイムバリデーションを設けましょう。

特に、複数の条件があるフォームは現在どの条件を達成しているかを明示することで入力エラーを未然に防ぐことができます。

「有効なメールアドレスを入力してください」というようなメッセージは具体性がなく、ユーザーは改善方法がわからないので注意しましょう。

手動入力を避ける



個数入力や音量調整、項目入力など入力可能な要素に制限がある場合は、ユーザーによる手動入力を避けましょう。
キーボードを使った手動入力はエラーにつながる可能性があります。

ステッパー、ドロップダウンメニュー、ラジオボタン、スライダー、ピッカーなどを用いて制限をかけることでユーザーが選択可能な値のみを表示しましょう。

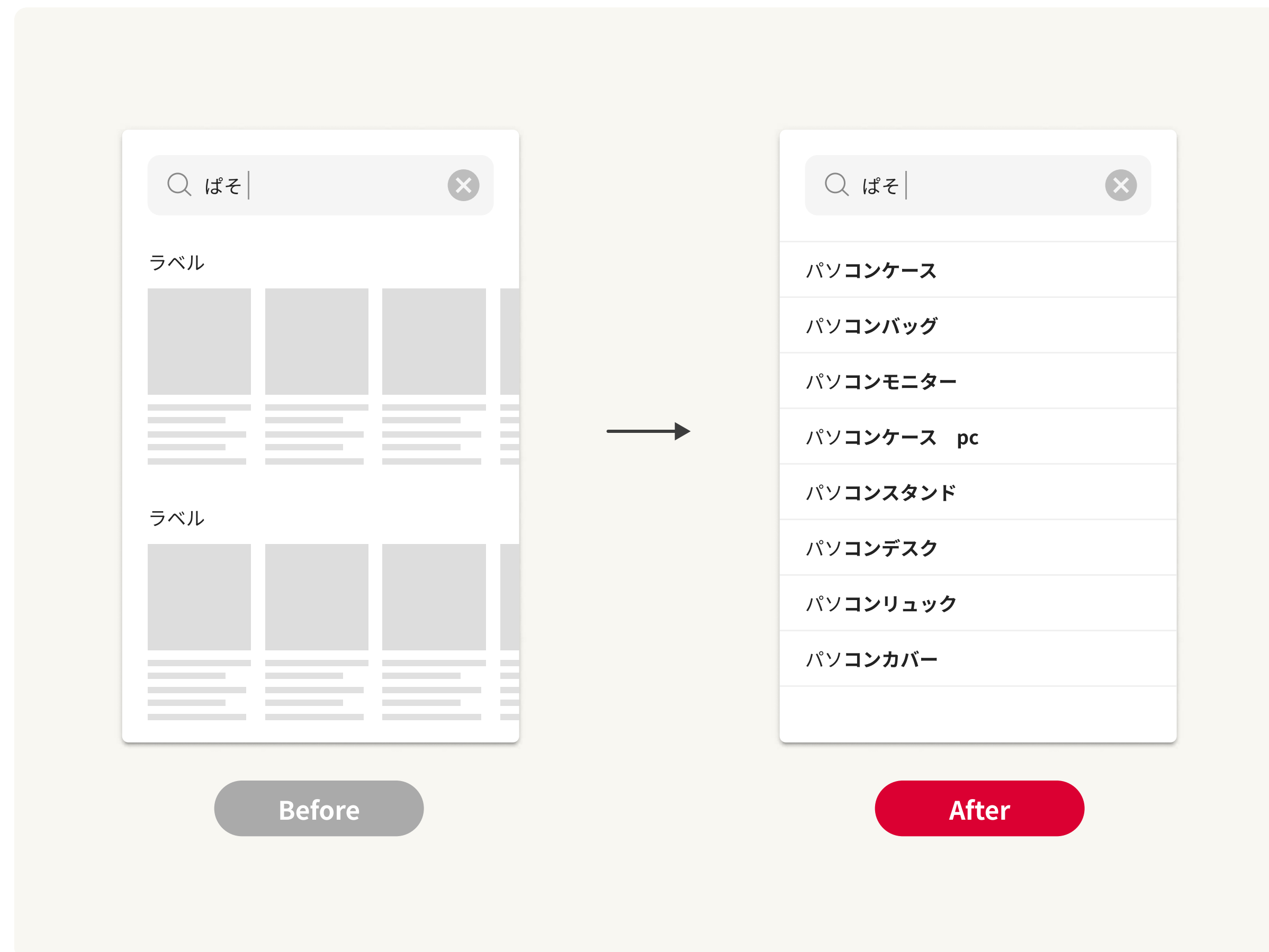
06

想起よりも認識

Recognition rather than recall



入力候補を表示する



タイピングは負担が大きく、誤字脱字などのエラーにつながる可能性が高まります。

テキスト入力の際には、入力中の文字に続く候補を表示しましょう。ユーザーが入力しようとしていたことや関連する候補を表示することで、入力の手間を省くことができ、よりスムーズに検索を行うことができます。

プレースホルダーで記入例を明示する

The diagram illustrates the process of adding placeholder text to form input fields. It is divided into two parts: 'Before' and 'After', connected by a right-pointing arrow.

Before: A form with three input fields. The first field is labeled '氏名' (Name), the second '電話番号' (Phone Number), and the third '住所' (Address). All fields are empty.

After: The same form, but with placeholder text added to each field. The '氏名' field contains '田中太郎' (Tanaka Taro), the '電話番号' field contains 'XXX-XXXX-XXXX', and the '住所' field contains '〇〇県〇〇市×× 1-2-3'.

Below the 'Before' and 'After' sections are two buttons: a grey button labeled 'Before' and a red button labeled 'After'.

そのフォームが何を入力するフォームであるかをラベル以外に判断する手助けとしてプレースホルダーを設けましょう。

注意点として、プレースホルダーをラベルとして使うことは避けましょう。プレースホルダーは入力中に消えてしまうため何のフォームであるかがわからなくなってしまいます。

適切なフォーム幅にする

The diagram illustrates the process of adjusting form width for better usability. It shows two versions of a form: 'Before' and 'After', connected by a right-pointing arrow.

Before: The form fields are wide. The '住所' (Address) field is particularly wide, containing the text '東京都渋谷区1-2-5'. A dark button labeled '住所自動入力' (Address auto-input) is positioned below the address field.

After: The form fields are narrower. The '住所' field now contains the same text but is shorter. The '住所自動入力' button is now positioned to the right of the address field, indicating that the form width is now appropriate for the content and the button placement.

フォーム幅を必要以上に長く取る必要はありません。

適切なフォーム幅は、ユーザーが入力するコンテンツの種類を推測する手助けとなります。

例えば、郵便番号や電話番号のようにある程度の文字数が決められたフォームは、フォーム幅が短いことで入力する内容を推測することができます。

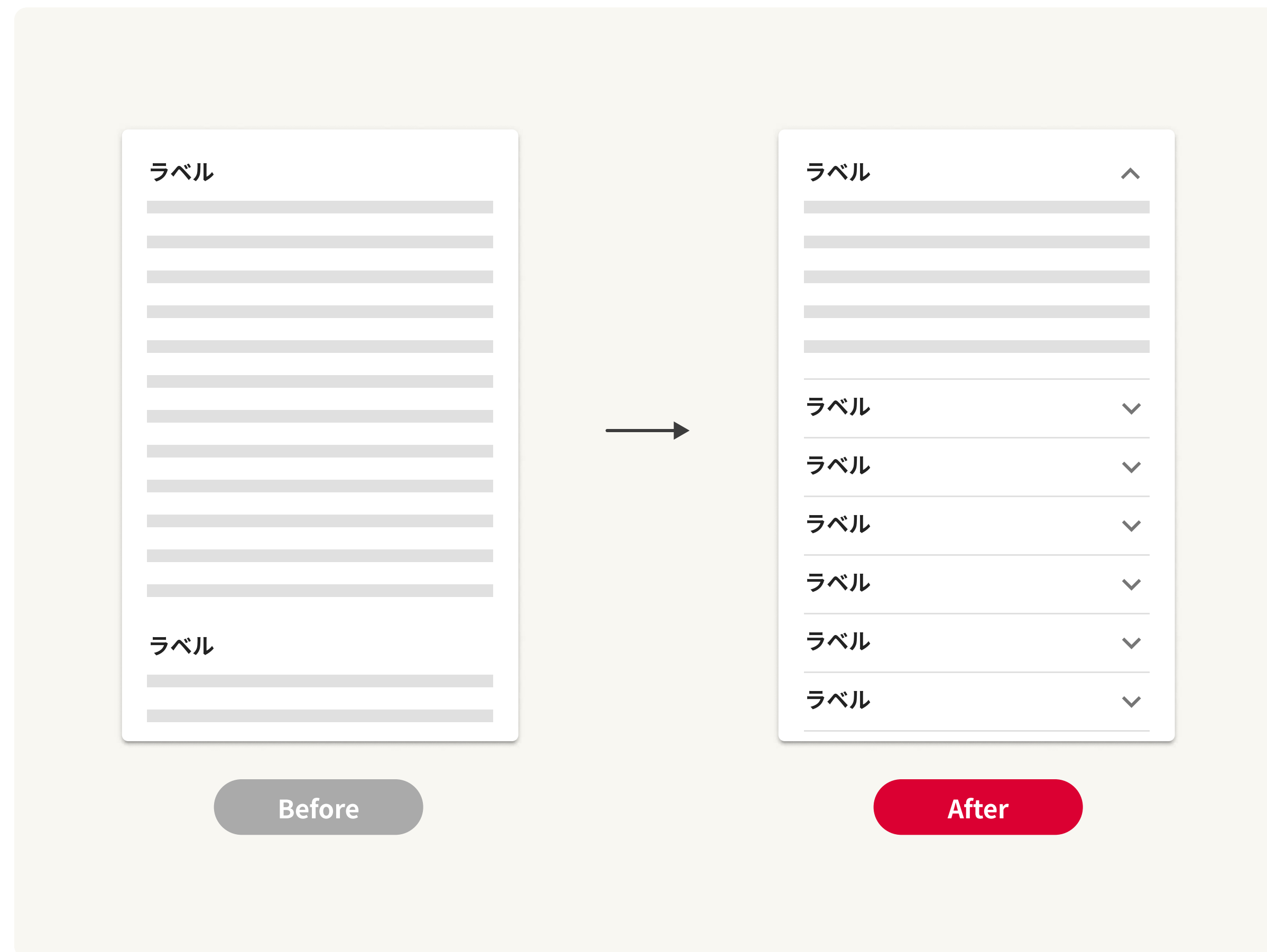
07

使用の柔軟性と効率性

Flexibility and efficiency of use



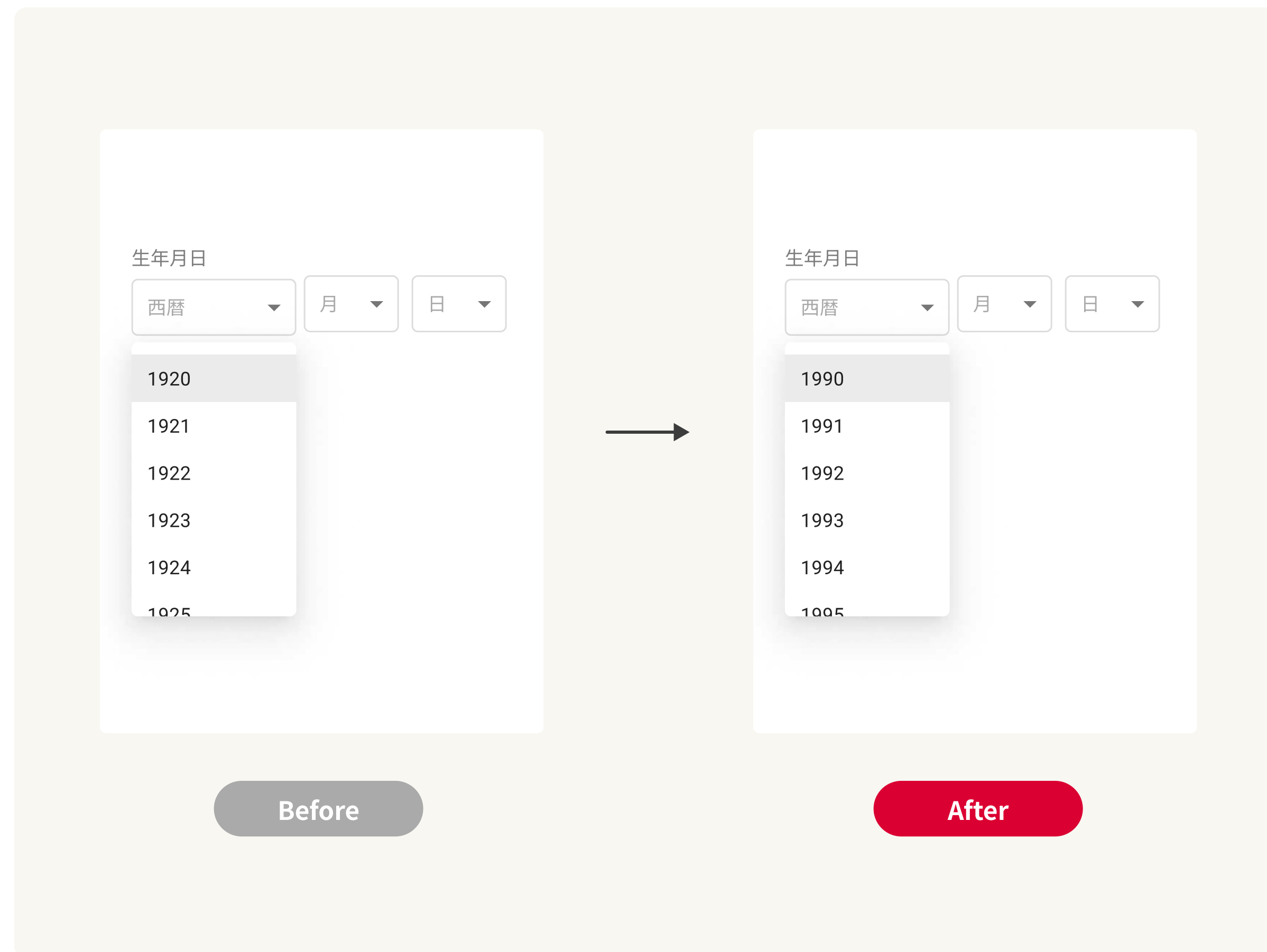
スクロール負荷を少なくする



コンテンツ幅が長いものが複数あると、ユーザーが下までスクロールするのに時間がかかります。

アコーディオンをつけてコンテンツ幅を削減することでスクロールの負荷を少なくすることができます。

価値のあるデフォルト値にする



設定項目やセレクトメニューの選択肢などは、ユーザーにとって価値のあるデフォルト値を設定して操作負荷を軽減しましょう。

利用するユーザーの大多数が該当する値であれば、その値をデフォルト値として設定することでスムーズに入力できます。

例えば、若者向けのサービスで生年月日の入力フォームがある場合に、西暦のセレクト要素は、1920年より1990年の値をデフォルト値に設定することで多くのユーザーの操作負荷が軽減できます。

自動入力できるものは入力させない

The image shows a comparison of a form before and after an automatic address input feature. The 'Before' state shows a form with fields for birth date (1996/05/10), age (25), postal code (1230045), prefecture (東京都), and city/ward/district (東京都渋谷区). The 'After' state shows the same form, but with a button labeled '住所自動入力' (Automatic address input) next to the postal code field, indicating that the system has automatically populated the address information based on the postal code.

生年月日 年齢
1996/05/10 25

郵便番号
1230045

都道府県
東京都

市町村区
東京都渋谷区

Before

生年月日 年齢
1996/05/10 25

郵便番号
1230045 住所自動入力

都道府県
東京都

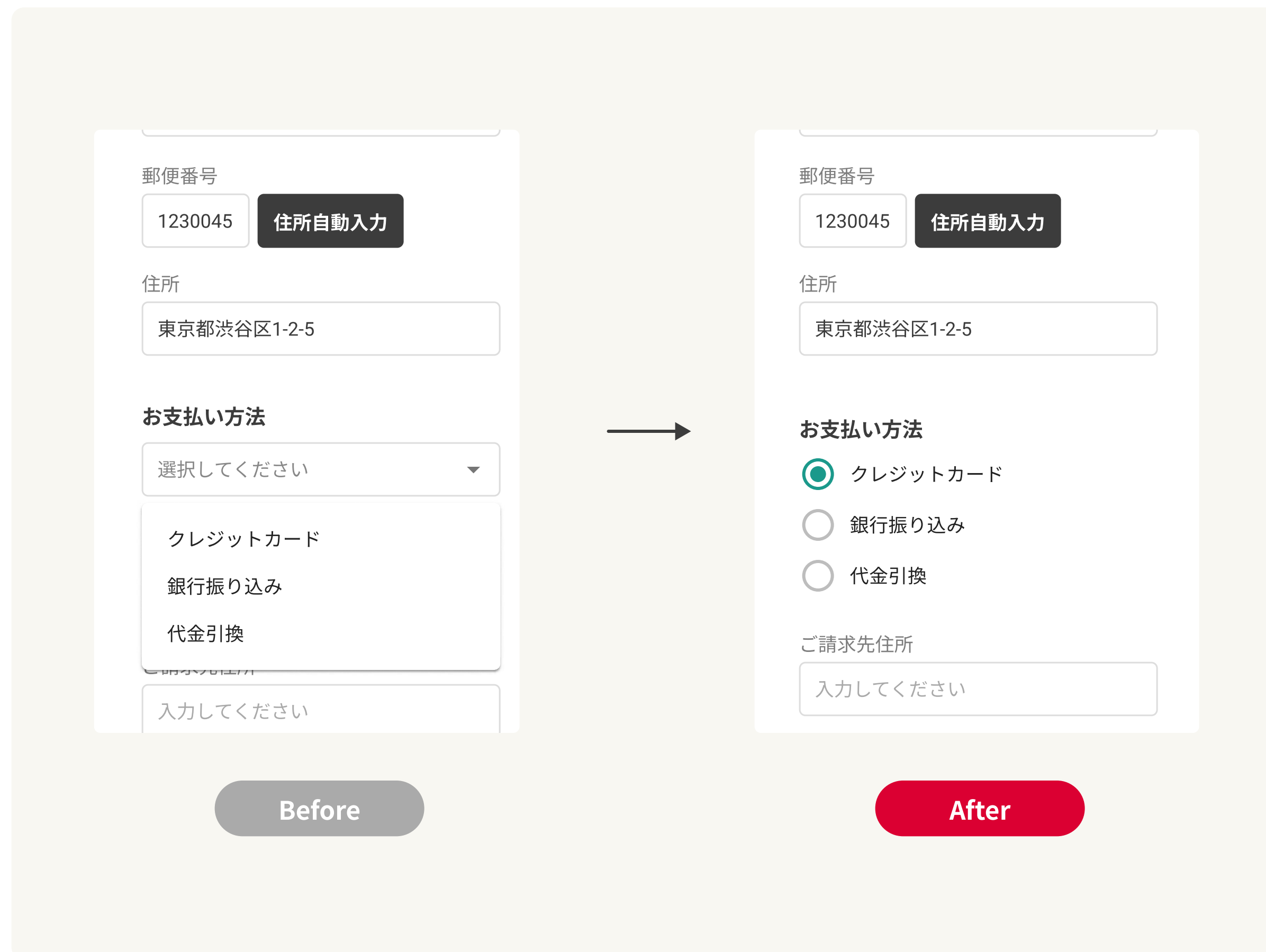
市町村区
東京都渋谷区

After

入力した情報のつじつまが合わなくなる可能性がある操作をユーザーに求めてはいけません。

ユーザーに不要入力させる必要がない要素は自動入力されるようにしましょう。

選択肢が3個以下の時には、セレクト要素ではなくラジオボタンを使う



セレクト要素は画面領域を必要とせず、コンパクトに選択肢を表示したいときに便利なUIパーツです。

一方でラジオボタンと比較すると、選択肢を展開する手間がかかる、一目で全ての選択肢を俯瞰できない、というデメリットがあります。

そのようなデメリットを考慮し、選択肢が2,3個の場合にはラジオボタンを、選択肢が多い場合はセレクトボックスを使用しましょう。

スイッチとチェックボックスを使い分ける



スイッチとチェックボックスは、どちらも何かを設定するときに使われますが、設定する項目が適用されるタイミングがそれぞれ違います。

スイッチはそれ自体がトグルボタンのため、設定を決めるときに「保存」ボタンを押す必要がありません。私たちが普段現実でスイッチを押す時のように即座に条件が適応される必要がある場合に使います。

チェックボックスは、最終的にユーザーが「保存」ボタンのようにアクションを実行するボタンを押した後に初めて適用されます。

頻繁に複数の変更をまとめて行う場合はチェックボックスを使用し、数カ所の変更で済む場合はスイッチを使用しましょう。

08

美的でミニマルなデザイン

Aesthetic and minimalist design



可読性の高いテキストデザインにする

Before

行間0% (0文字) 1行あたり60文字

吾輩は猫である。名前はまだ無い。どこで生れたかとうと見当けんとうがつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番狸悪くあくな種族であったそうだ。この書生というのは時々我々を捕つかまえて煮にて食うという話である。しかしその当時は何という考もなかったから別段恐いとも思わなかった。ただ彼の掌てのひらに載せられてスーと持ち上げられた時何だかフワフワした感じがあつたばかりである。掌の上で少し落ちついて書生の顔を見たのがいわゆる人間というものを見始めはじめてであらう。この時妙なものだと思つた感じが今でも残っている。第一毛をもって裝飾されべきはずの顔がつるつるしてまるで薬缶やかんだ。



After

行間100% (1文字) 1行あたり30文字

吾輩は猫である。名前はまだ無い。どこで生れたかとうと見当けんとうがつかぬ。何でも薄暗いじめじめした所でニャーニャー泣いていた事だけは記憶している。吾輩はここで始めて人間というものを見た。しかもあとで聞くとそれは書生という人間中で一番狸悪くあくな種族であったそうだ。この書生というのは時々我々を捕つかまえて煮にて食うという話である。しかしその当時は何という考もなかったから別段恐いとも思わなかった。ただ彼の掌てのひら…

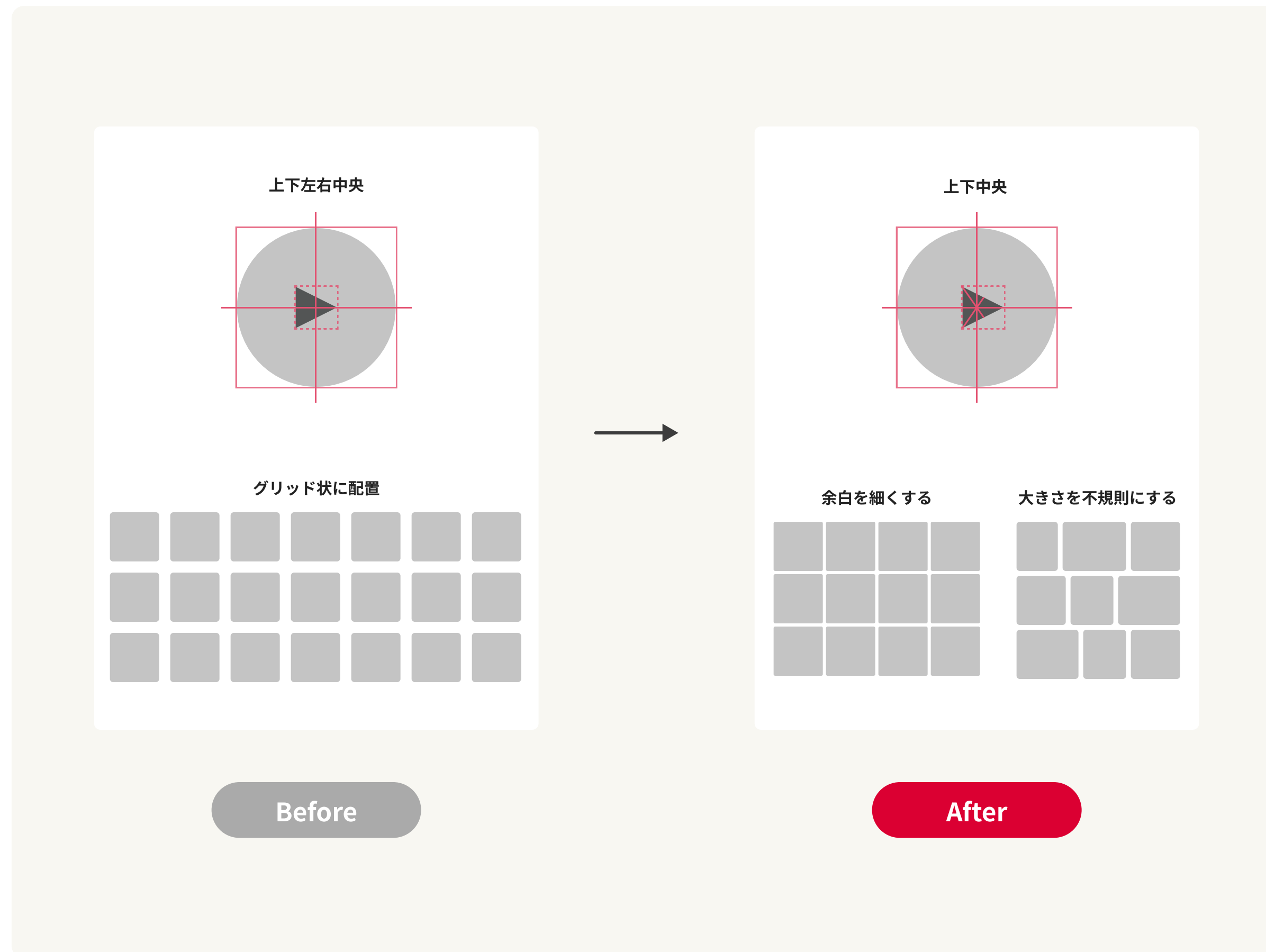
行間・行長は文章の読みやすさに関わる重要な要素です。

上下のテキスト間の距離が詰まりすぎたり広がりすぎていると読みづらく、1行あたりの文字数が多すぎたり少なすぎたりすると次の行を見失う可能性や、文章が途切れ途切れになり、可読性の低いテキストになります。

行間は、文字サイズの50%～100%前後（0.5文字～1文字分）の行間が適切とされています。

行長は、文字サイズによって変わる場合もありますが、1行に25文字～40文字程度の長さがストレスなく読むことのできる長さとしてされています。

錯視を考慮して視覚調整を行う



「錯視」とは、錯覚の一種で、目から入った情報を、脳が勝手に推測したり補完することにより、実際の姿とは異なって見える現象を指します。

錯視の代表的な例として図のように、

- ・ 三角形を数値上では円の中心に配置しているにもかかわらず、左側に少し寄っているように見える。（視覚的な中心ではない）
- ・ グリッド状に配置された模様（ハンマーグリッド）の白い交差点の部分が、灰色あるいは、影になったように見える。

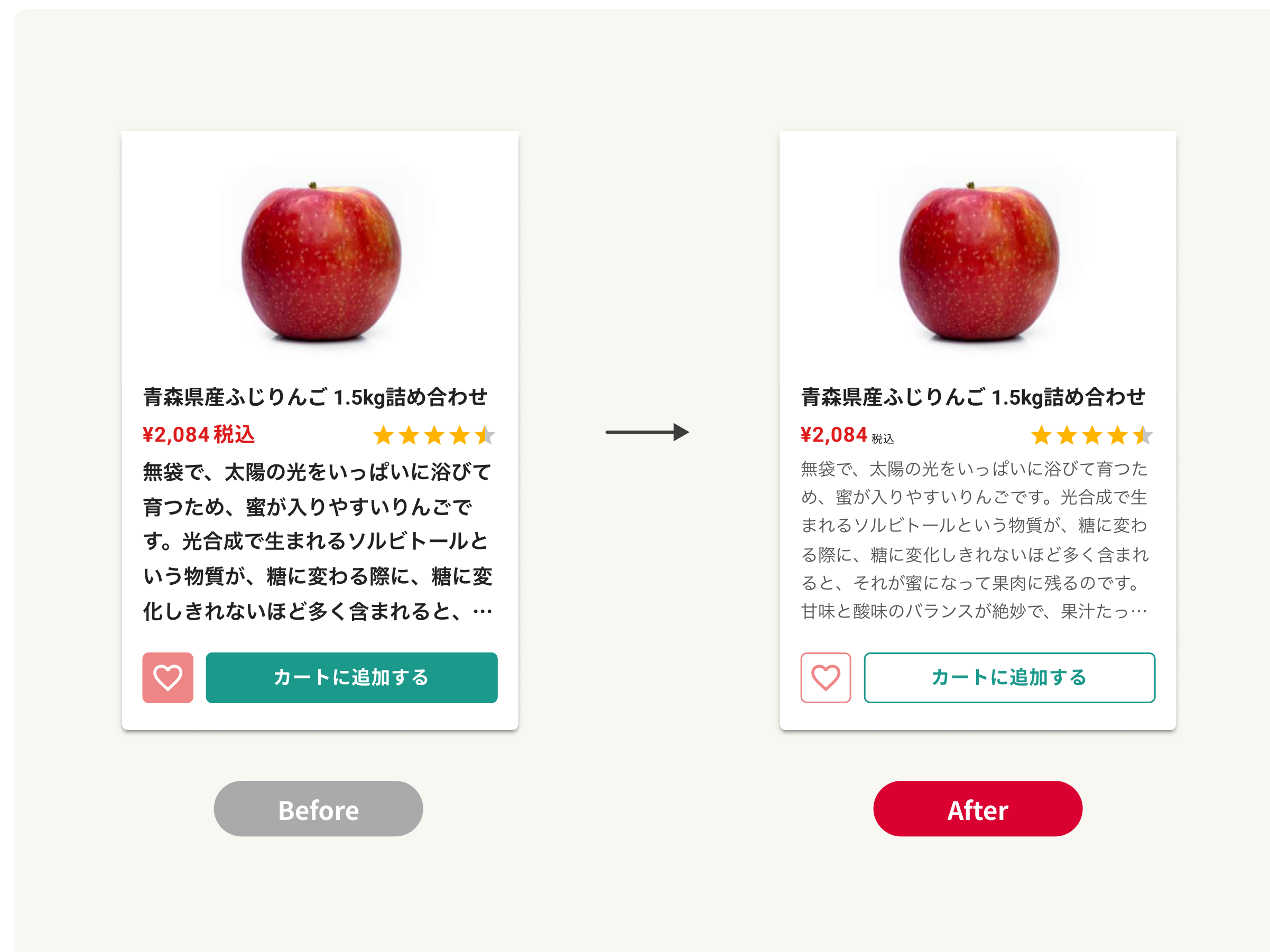
などがあります。

単純な数値上での配置をするだけでなく、

- ・ 三角形の重心が中央にくるように配置する
- ・ 余白を細くする
- ・ オブジェクトの大きさを不規則にする

などを行い、錯視が起きないように調整しましょう。

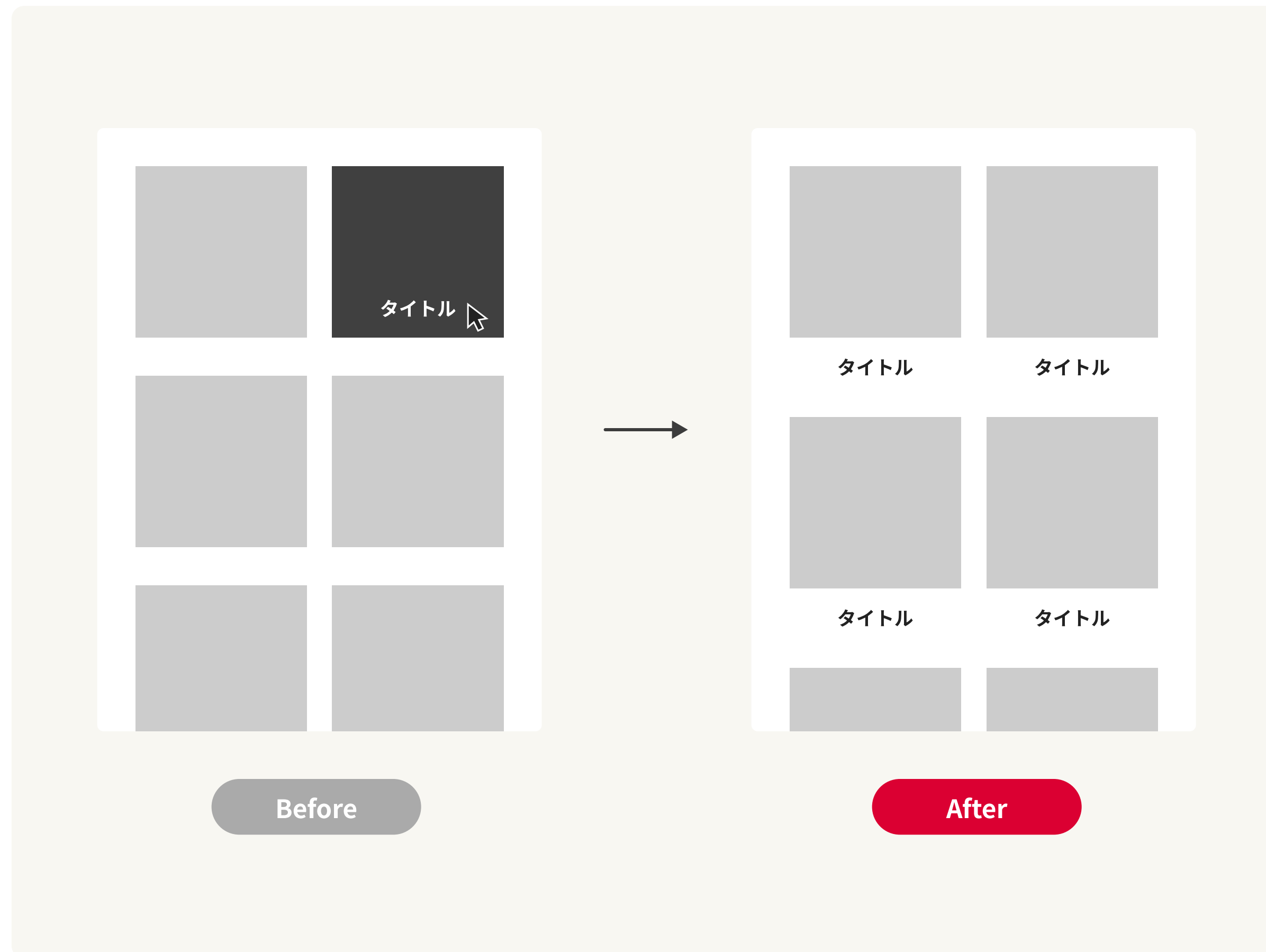
シンプルにする



UIにおける情報の優先度と視覚誘導のためには、特定の要素を太字にしたり色をつけたりと、強調表現によって他の要素と差をつける必要がありますが、あれもこれもと足し算のように属性を増やしていくと優先度がインフレ・渋滞して逆効果となってしまいます。

使用する色やフォントの種類は極力絞り、シンプルにすることを心がけましょう。

むやみに情報を隠さない



ビジュアルを優先させた結果が、「マウスホバーしないと情報や機能が表示されない」ようなデザインになってしまうことで、偶然性・意外性・判断や操作のスピードなどをユーザーに求めてはいけません。

たまたまその操作を行なったから気づいた、では全てのユーザーが平等に情報や機能を扱うことができません。

明示的で可能な限り誘導的なデザインにすることで、どのユーザーにとっても常に同じレベルで情報や機能を提示することができます。

ボタンの優先度を考慮する



複数のボタンが存在する画面で、同じ見た目のボタンを並べると、何のボタンであるかを一目で判断できず、テキストやアイコンなどを注視して判断する必要があるため、正しく認識するまでに時間がかかります。

そのような状態を防ぐために、優先度を考慮して見た目に差をつけ、それぞれがどのようなタイプのアクションであるかを素早く識別できるようにしましょう。

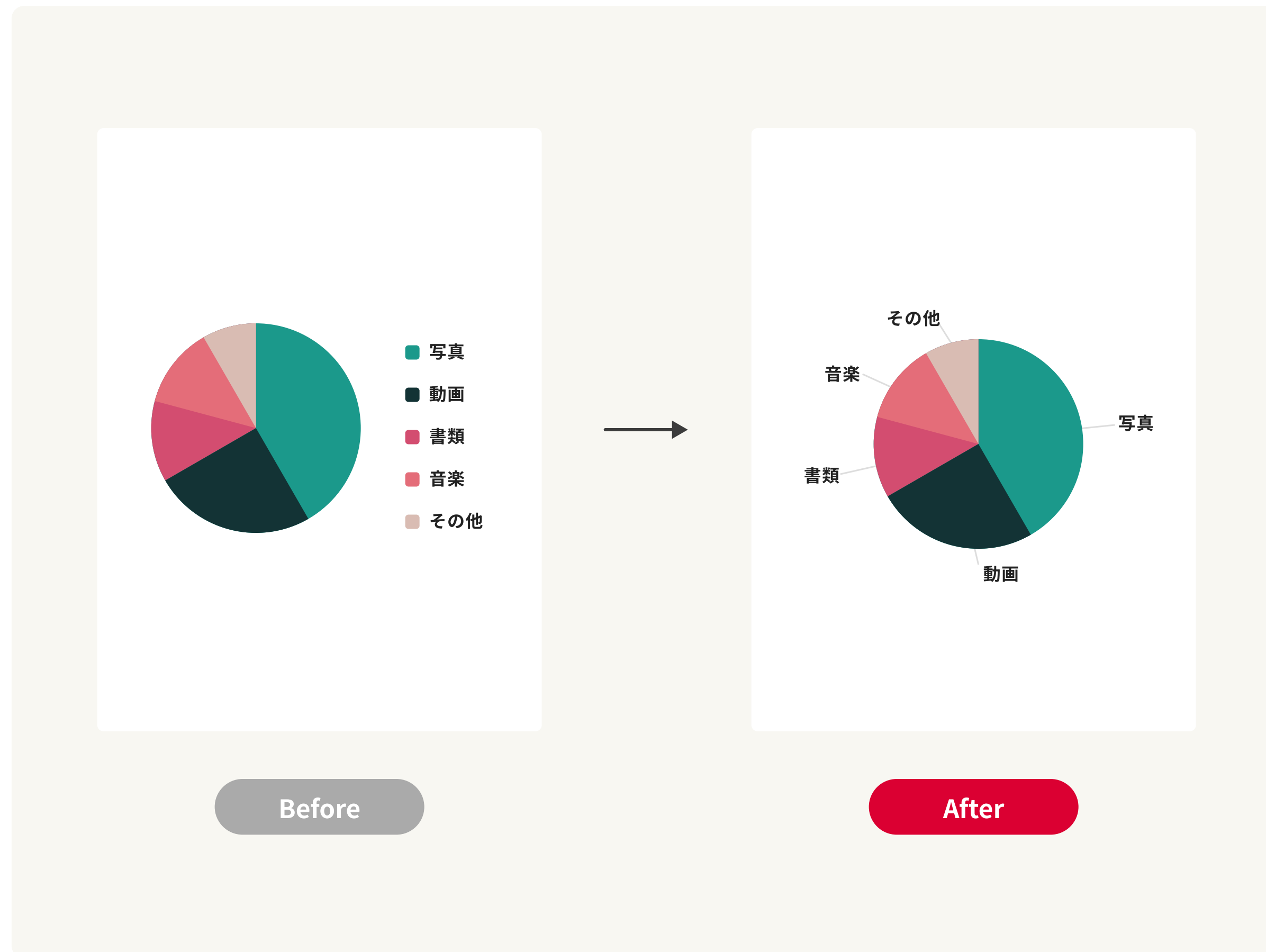
フォームでの入力項目はグループごとにまとめる

The image shows a comparison of two form designs. On the left, labeled 'Before', the form fields are arranged vertically: 名前 (田中太郎), フリガナ (タナカタロウ), 生年月日 (1997/2/22), 郵便番号 (1230045) with a 住所自動入力 button, 住所 (東京都渋谷区1-2-5), and 電話番号. On the right, labeled 'After', the form is grouped into three sections: 'お客様情報' containing 名前 (田中太郎), フリガナ (タナカタロウ), and 生年月日 (1997/2/22); 'お届け先' containing 郵便番号 (1230045) with a 住所自動入力 button and 住所; and a separate 電話番号 field. An arrow points from the 'Before' form to the 'After' form.

フォームの入力項目はグループを認識しやすいように適切なグループ・サイズに切り分けましょう。

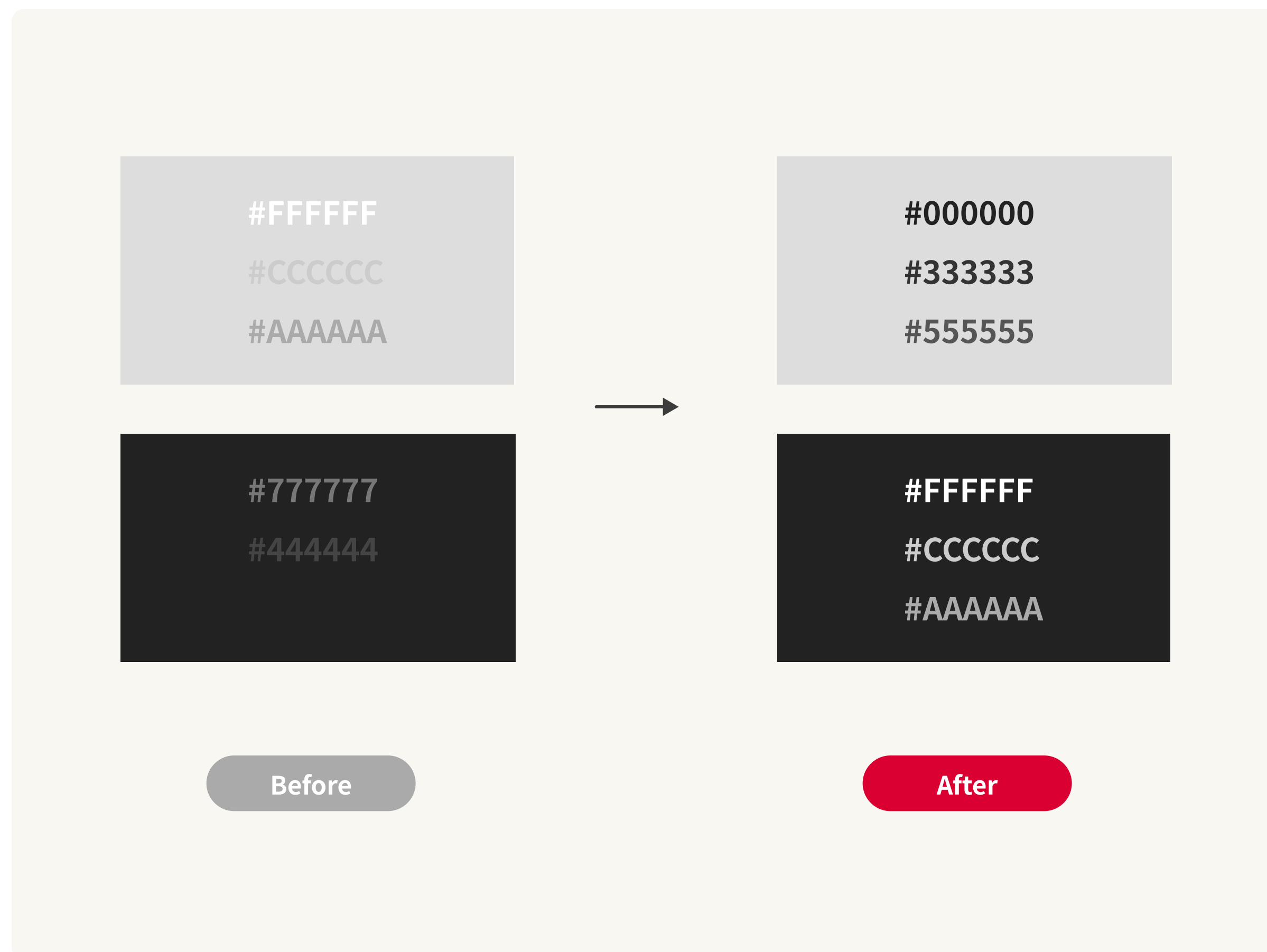
関連する情報をあらかじめグループごとにまとめることで認知負荷が軽減され、ユーザーが各項目の意味や分類をより早く理解できるようになります。

色だけに頼らない



色だけに頼ったデザインは避けましょう。
色がなくても情報が伝わるようにする必要があります。
日本人男性の20人に1人が色覚特性を持っていると言われています。
ユーザーの中には色の識別が困難であるユーザーがいるという前提で
色の違いのみによる情報の区別を避けましょう。
枠線や下線による装飾、形状や位置関係によるマッピング、文字による
補足、などを用いてユーザーの識別を補助しましょう。

コントラスト比をクリアしたテキストカラーを使う



情報の優先度でテキストカラーを調整することは有効な手段ですが、テキストが読めなくては意味がありません。

一つの基準として、WCAG（Web Contents Accessibility Guideline）において「テキスト及び画像化された文字の視覚的な表現には、少なくとも4.5:1のコントラスト比がある」と定められています。

すべてのテキストが4.5:1のコントラスト基準をクリアすることは難しいですが、読んでもらいたいテキストはコントラスト比が4.5:1以上になるようにしましょう。

情報のかたまりの数を7つ以下にする



ジョージ・ミラーによると、短期記憶で処理できる情報量は7±2個程度であると言われています。

7±2より少なくすると、脳への負荷が減り、認知が容易になり、結果的に使いやすく感じる可能性が高まります。

逆に7±2より多くなれば、認知しづらく、使いにくくストレスに感じ可能性が高まります。

ナビゲーションのような項目が複数存在するものは、単なる羅列ではなく、各カテゴリにタイトルラベルをつけるなどして項目を7±2以下にすることが望ましいです。

09

エラー時のユーザーによる認識・診断・回復の補助

Help users recognize, diagnose, and recover from errors



エラー内容をリアルタイムに明示する

The image illustrates a design improvement for error handling in a form. It is divided into two parts: 'Before' and 'After', connected by a right-pointing arrow.

Before: A form with five input fields: '姓' (Surname), '名' (Name), '姓 (カナ)' (Surname in Kana), '名 (カナ)' (Name in Kana), and '郵便番号' (Postal Code). An error message box at the top left contains a red warning icon and the text: '以下の入力に誤りがあります' (There is an error in the following input), '-姓 (カナ)' (Surname in Kana), and '-名 (カナ)' (Name in Kana). The '姓 (カナ)' field contains '田中' (Tanaka) and is highlighted with a red border.

After: The same form, but the error message is moved to the side of the '姓 (カナ)' field. The error message now reads: 'カタカナで入力してください' (Please enter in Katakana). The '姓 (カナ)' field contains '田中' (Tanaka) and is highlighted with a red border. The other fields remain the same.

フォーム上部にエラーメッセージがある場合、ユーザーはそれぞれのフォームに対応するメッセージの内容を記憶し、エラーフォームまで移動し、入力する必要があるため高い認知負荷がかかります。そのため修正の効率に影響が出るでしょう。

上記を避けるために、エラーメッセージはリアルタイムにそのフォームの側に表示しましょう。そうすることで、少ない認知負荷でエラー内容を修正できます。作業や記憶の負担が少なければユーザーはフォームをより早く、正確に修正することができます。

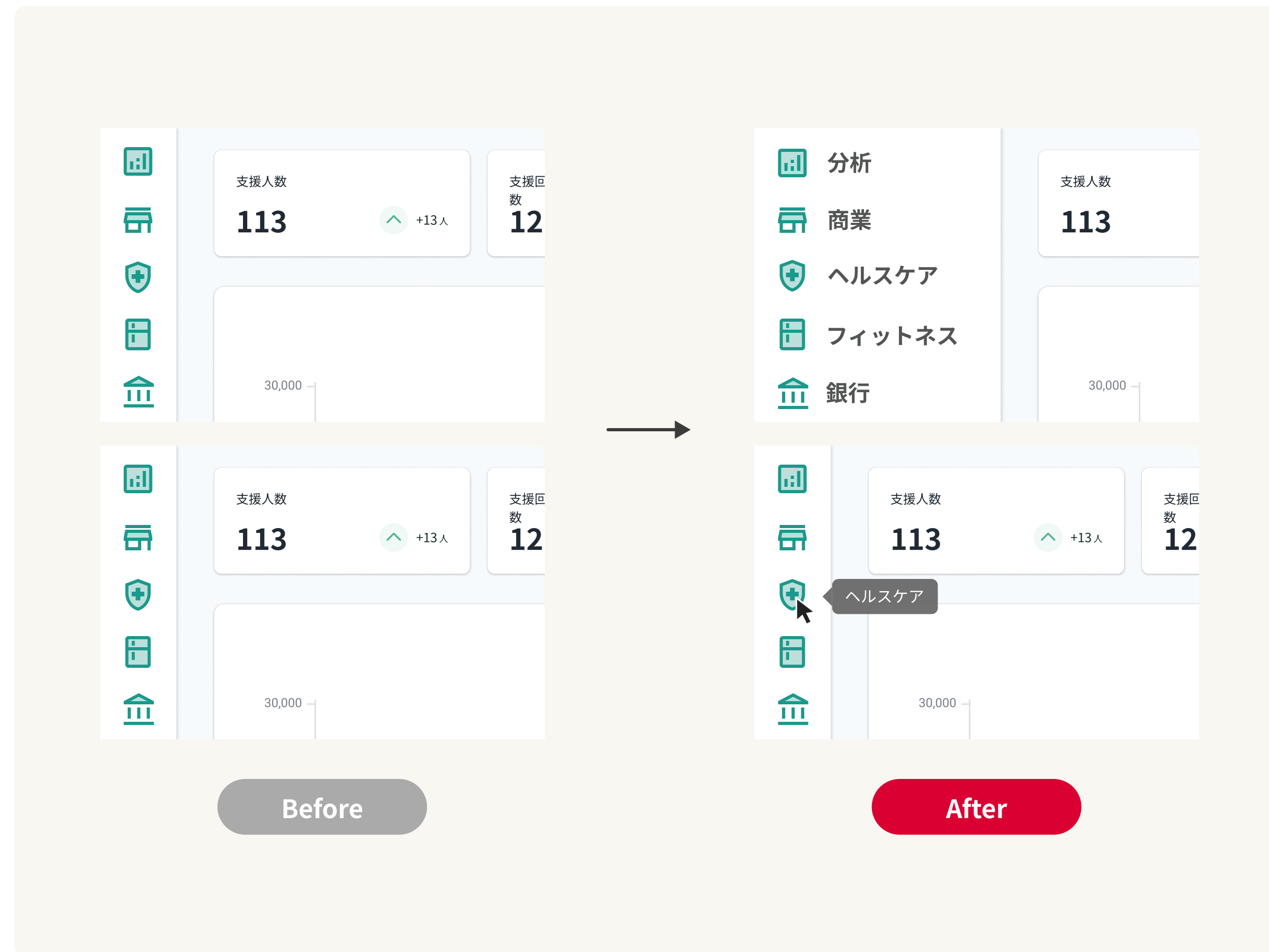
10

ヘルプとドキュメント

Help and documentation



アイコンだけに頼らない



アイコンは、機能を直感的に伝えるのに有効ですが、意味が伝わらなければユーザーは誤解し、混乱してしまいます。

アイコンはラベルと併用し、あくまでラベルの意味を補助するものとして使うことが望ましいです。

アイコン単体で使う場合も、必ず意味の伝わる一般的なものに限定し、ツールチップなどでユーザーの理解を手助けしましょう。

案件のお問い合わせはこちらから

TEL [03-6419-7700](tel:03-6419-7700)

Email sales@i3design.co.jp

お問い合わせフォーム <https://www.i3design.jp/contact>

一緒に働く仲間を募集しています



募集職種一覧はこちら

<https://www.i3design.jp/recruit>

i3DESIGN