

作って学ぶ

WordPress

ブロックテーマ

エビスコム 著

補足 PDF

ハイブリッドテーマ



本 PDF は下記書籍の副読本です。



作って学ぶ WordPress ブロックテーマ

- ⤒ <https://book.mynavi.jp/ec/products/detail/id=135047>
- ⤓ <https://ebisu.com/wp-blocktheme/>
- ⤔ <https://amzn.to/3WGSfj7>

- ・本書に記載された内容は、情報の提供のみを目的としております。したがって、本書を用いての運用はすべてお客様自身の責任と判断において行ってください。
- ・本書の制作にあたっては正確な記述につとめましたが、著者や出版社のいずれも、本書の内容に関してなんらかの保証をするものではなく、内容に関するいかなる運用結果についてもいっさいの責任を負いません。あらかじめご了承ください。
- ・本書中に掲載している画面イメージなどは、特定の設定に基づいた環境にて再現される一例です。ハードウェアやソフトウェアの環境によっては、必ずしも本書通りの画面にならないことがあります。あらかじめご了承ください。
- ・本書は 2022 年 12 月段階での情報に基づいて執筆されています。本書に登場するソフトウェアのバージョン、URL、製品のスペックなどの情報は、すべてその原稿執筆時点でのものです。執筆以降に変更されている可能性がありますので、ご了承ください。
- ・本書中に登場する会社名および商品名は、該当する各社の商標または登録商標です。本書では®およびTMマークは省略させていただいております。



もくじ

Step 1 ハイブリッドテーマの作成 5

ハイブリッドテーマの構成ファイルを用意する.....	7
各種機能を有効化する.....	8
投稿エディターを確認する.....	9

Step 2 フロントにコンテンツを出力する 10

ページの構成に必要なコードを用意する.....	10
コンテンツとタイトルを出力する.....	11

Step 3 ブロックの横幅、間隔、左右の余白を コントロールする 12

ブロックの横幅と間隔をコントロールする.....	12
左右の余白をコントロールする.....	13
タイトルの横幅と左右の余白もコントロールする.....	14
タイトルとコンテンツの間隔をコントロールする.....	15

Step 4 クラスを使ってスタイリングする 17

タイトルをスタイリングする	17
コンテンツ内のリンクをスタイリングする	18

Step 5 ヘッダーとフッターをテンプレートpartsで追加する.....	19
partsの間隔をカスタマイズする	20
Step 6 ブロックベースのテンプレートpartsを作成する.....	22
テンプレートpartsを編集する	23
テンプレートpartsの編集結果をテーマに反映する.....	24
テンプレートpartsを使う	25
Step 7 記事一覧ページを作成する	26
Step 8 アーカイブページを作成する.....	27
Step 9 404 ページと検索結果ページを作成する.....	28
Step 10 タイトルを出力しない カスタムテンプレートを作成する.....	30
Step 11 ハイブリッドテーマの構成.....	32



ハイブリッドテーマの作成

ブロックテーマと比較すると、従来のクラシックテーマでは利用できる機能が限られます。そのため、ブロックテーマの機能を取り込む「ハイブリッドテーマ」の作成を検討します。

機能	ブロック	ハイブリッド	クラシック	書籍参照
サイトエディター	○	×	×	P.92
スタイルサイドバー	○	×	×	P.138
テンプレートパーツエディター	○	○	○	P.206
テンプレートやテンプレートパーツの新規作成	○	×	×	P.222
ブロックパターン	○	○	○	P.301
スタイルバリエーション	○	×	×	P.320
Create Block Themeプラグイン	○	×	×	P.70
エクスポート機能	○	○	○	P.107
レイアウト機能	○	○	✖	P.39
テーマの theme.json による	UIコントロールの有効化・無効化	○	○	△
	プリセットの指定	○	○	△
	ベースとなるスタイルの指定	○	○	✖

※WordPress 6.1.1で確認しています

✖…ブロックテーマではないため利用できません

✖…theme.jsonがないため利用できません

△…add_theme_support()で用意されたもののみ利用できます(参照:P.57)

ハイブリッドテーマでは、ブロックテーマのときと同じように theme.json でエディターの UI コントローラやプリセットの選択肢を用意し、ベースとなるスタイルを作成できます。ブロックベースのテンプレートパーツ、レイアウト機能、グローバルスタイルを活用すると、ページ全体を効率よくコーディングしていくことも可能です。

ただし、サイトエディターやスタイルサイドバーは使えないため、theme.json などはブロックテーマで作成したものを利用するのが簡単です。

Step 1 ハイブリッドテーマの作成

ここでは書籍で完成させたブロックテーマ「My Theme」をベースに、使える機能はできるだけ使う形でハイブリッドテーマ「Hybrid My Theme」を作成していきます。ハイブリッドテーマでも、次のようにブロックテーマと同じデザインのサイトを構築します。

※コンテンツデータも書籍で完成させたものを使用します。



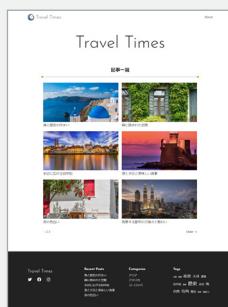
サイト型のトップページ
/



記事ページ
/スラッグ/



アバウトページ
/about/



記事一覧ページ
/blog/



アーカイブページ
/category/スラッグ/
/tag/スラッグ/



404ページ
/見つからないページ



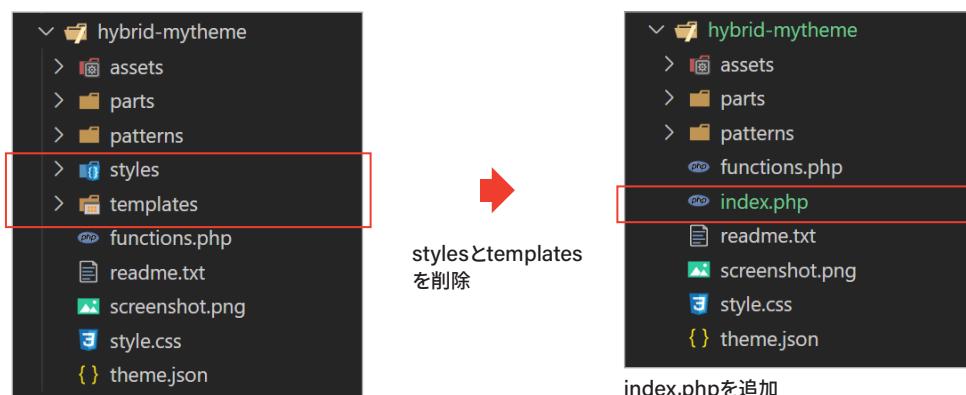
検索結果ページ
/?s=キーワード

+ ハイブリッドテーマの構成ファイルを用意する

まずは、ハイブリッドテーマの構成ファイルを用意します。ブロックテーマ「My Theme」のテーマフォルダ「mytheme」をコピーし、「hybrid-mytheme」フォルダを作成します。

このフォルダ内のファイル構成は次のようにになります。ハイブリッドテーマで使用できないスタイルバリエーション `styles/` とブロックベースのテンプレート `templates/` は削除します。代わりに、PHP ベースのインデックステンプレート `index.php` を追加します。

「hybrid-mytheme」フォルダを作成



`style.css` ではテーマ名を「Hybrid My Theme」にします。`screenshot.png` はテーマに合わせた画像に置き換えます。

`assets/`、`parts/`、`patterns/` に保存したフォントと画像、ブロックベースのテンプレートパート、ブロックパターンはハイブリッドテーマでも使用します。さらに、`functions.php`、`style.css`、`theme.json` で指定した設定もそのまま使用します。

The top part shows a snippet of the 'style.css' file with the following code:

```
/*
Theme Name: Hybrid My Theme
*/
/* 見出し： 丸付き飾り罫 */
:is(h1, h2, h3, h4, h5, h6).is-style-decoration-line {
```

The bottom part shows a screenshot of a browser window displaying the theme. The header says 'Hybrid My Theme' and 'New Site hybrid-mytheme'. The screenshot also shows the URL 'hybrid-mytheme/screenshot.png'.

+ 各種機能を有効化する

functions.php に以下の設定を追加し、ブロックテーマでは標準で有効になる機能をハイブリッドテーマでも有効化します。

```
<?php

function mytheme_support() {

    // コアブロックの追加分の CSS を読み込む
    add_theme_support( 'wp-block-styles' );

    // テーマの CSS (style.css) をエディターに読み込む
    add_editor_style( 'style.css' );

    // add_editor_style() を有効化
    add_theme_support( 'editor-styles' ); •—————> add_editor_style()で指定したCSSをエディターに読み込む機能を有効化します。

    // ページのタイトルの出力を有効化
    add_theme_support( 'title-tag' ); •—————> 書籍P.330のようにページのタイトル<title>を出力します。

    // HTML5 対応を有効化
    add_theme_support( 'html5', array(
        'style',
        'script'
    ) ); •—————> <style>や<script>にtype属性を付加せず、HTML5に準拠した出力にします。

    // アイキャッチ画像を有効化
    add_theme_support( 'post-thumbnails' ); •—————> アイキャッチ画像を有効化します。

    // 埋め込みブロックのレスポンシブを有効化
    add_theme_support( 'responsive-embeds' ); •—————> 埋め込みブロックを縦横比を維持したレスポンシブにします。
}

add_action( 'after_setup_theme', 'mytheme_support' );
...
```

hybrid-mytheme/functions.php

ここでは配置の全幅・幅広を有効化する add_theme_support('align-wide') を指定していません。この指定がなくても、theme.json で指定した contentSize と wideSize によって有効化されます。

```
{
    "settings": {
        ...
        "layout": {
            "contentSize": "756px",
            "wideSize": "980px"
        },
        ...
    }
}
```

hybrid-mytheme/theme.json

+ 投稿エディターを確認する

ここまで設定で、投稿エディターがロックテーマのときと同じ状態で機能するようになります。作成したハイブリッドテーマ「Hybrid My Theme」を【外観>テーマ】で有効化し、記事や固定ページのコンテンツを【投稿>投稿一覧】、【固定ページ>固定ページ一覧】から投稿エディターで開きます。



theme.json で設定したスタイルにしたがってコンテンツが表示され、UI コントロールやプリセットが機能していることを確認します。

The diagram illustrates several features of the hybrid editor interface:

- ブロックの横幅: コンテンツ、幅広、全幅 (P.153)**: Shows a screenshot of the editor interface with a callout pointing to a specific block width setting.
- ブロックの間隔 (P.153)**: Shows a screenshot of the editor interface with a callout pointing to a gap setting.
- ブロックパターン (P.301)**: Shows a screenshot of the editor interface with a callout pointing to a block pattern section.
- テキスト、見出し、リンク、ボタン、画像のベースとなるスタイル (P.139～)**: Shows a screenshot of the editor interface with a callout pointing to a style panel.
- レイアウト機能 (P.94)**: Shows a screenshot of the editor interface with a callout pointing to a layout panel.
- ブロックスタイル (P.189)**: Shows a screenshot of the editor interface with a callout pointing to a block style panel.
- フォントサイズ、色、スペースなどのプリセット (P.114～)**: Shows a screenshot of the editor interface with a callout pointing to a font size and color panel.

2

Hybrid Theme

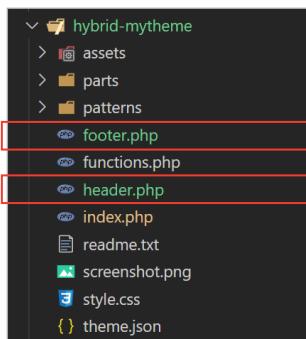


フロントにコンテンツを出力する

ここからはフロントのページを作っています。まずは、ページの構成に必要なコードを用意し、コンテンツを出力します。エディターでの表示と揃えるため、コンテンツには `theme.json` で設定したスタイルが適用されるようにします。

+ ページの構成に必要なコードを用意する

ブロックテーマと異なり、ハイブリッドテーマではページの構成に必要なコードを用意しなければなりません。PHP ベースのテンプレートパート `header.php`、`footer.php` を追加し、次のように HTML を記述して `index.php` に読み込みます。WordPress が各種設定の出力に使用する `<?php wp_head(); ?>` なども記述します。



```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width">
  <?php wp_head(); ?>
</head>
<body <?php body_class(); ?>>
  <?php wp_body_open(); ?>
```

header.php

hybrid-mytheme/header.php

```
<?php get_header(); ?>
```

index.php

hybrid-mytheme/index.php

```
<?php wp_footer(); ?>
</body>
</html>
```

footer.php

hybrid-mytheme/footer.php

`wp_head()`などは以下の場所に記述することが求められています。ブロックのスタイルやグローバルスタイルもこれらを使って出力されます。

`wp_head()`</head>の直前
`wp_body_open()`.....<body>の直後
`wp_footer()`</body>の直前

`body_class()`はページを区別するクラス名を出力します。

+ コンテンツとタイトルを出力する

コンテンツを `<?php the_content(); ?>` で、タイトルを `<?php the_title(); ?>` で出力します。タイトルは `<h1>` で、全体はメインコンテンツとして `<main>` でマークアップします。

フロントで記事ページや固定ページを開くと、次のように出力されます。

```
<?php get_header(); ?>  
  
<main>  
  <h1><?php the_title(); ?></h1>  
  <?php the_content(); ?>  
</main>
```

```
<?php get_footer(); ?>
```

hybrid-mytheme/index.php

海と歴史の佇まい

人は海とともに豊かになりました。エーゲ海に位置するこの島でも、そんな豊かな佇まいを感じることができます。夏真っ盛りという時期でもカラッとした空気で、爽やかです。窓がよく開き、日射に入ることで快適な空間になります。

白と青に彩られた日々

海中の島は白と青のコントラストで印象的でとてもきれいです。絶対的に花が咲くのは美しいお洒落な組み合せ特徴的で、非常に洗練されています。バスに乗りる前に、ちょっと休んで立ち寄った後では、美味しいリーフバーを楽しむことができました。



記事ページ

/waterfront/

About

旅のログ

Travel Timesでは皆様から届いた記事を中心し、旅に関する情報を毎週ご紹介しています。どこの国で何をしたらいいのかわからないときや、旅の途中で迷ったとき、積極的に問い合わせなどには、Travel Timesで旅のプランを見てみてください。



アバウトページ

/about/

コンテンツの横幅はコントロールされていませんが、前ページの `<?php wp_head(); ?>` により、この段階でもグローバルスタイルは出力されています。そのため、テキストや見出しが theme.json で指定したスタイルになり、Fluid タイポグラフィ（可変フォントサイズ）も機能しています。

海と歴史の佇まい

人は海とともに豊かになりました。エーゲ海に位置するこの島でも、そんな豊かな佇まいを感じることができます。夏真っ盛りという時期でもカラッとした空気で、爽やかです。窓がよく開き、日射に入ることで快適な空間になります。

白と青に彩られた日々

海中の島は白と青のコントラストで印象的でとてもきれいです。絶対的に花が咲くのは美しいお洒落な組み合せ特徴的で、非常に洗練されています。バスに乗りる前に、ちょっと休んで立ち寄った後では、美味しいリーフバーを楽しむことができました。



theme.jsonで指定したスタイル

<body>のテキスト (P144で設定)

フォントファミリーSystem Font

フォントサイズ中 (medium)

行の高さ1.8

<h1>～<h6>の見出し (P149で設定)

フォントファミリーJosefin Sans

行の高さ1.3

<h1>の見出し (P149で設定)

フォントサイズ特大 (x-large)

太さ標準

<h2>の見出し (P149で設定)

フォントサイズ大 (large)

3

Hybrid Theme



ブロックの横幅、間隔、左右の余白 をコントロールする

ブロックの横幅、間隔、左右の余白は theme.json で右のように指定しています。この設定でレイアウトするためには、P.39 のレイアウト機能で出力されるグローバルスタイルを利用します。

theme.jsonで指定したスタイル

コンテンツ幅 756px
幅広の横幅 980px
間隔 1.8em
左右の余白 プリセットの 3
(P.153、157、160で設定)

+ ブロックの横幅と間隔をコントロールする

まず、横幅と間隔をコントロールするためには、ブロックをコンテナに入れ、コンテナのレイアウトタイプを Constrained にします。たとえば、コンテンツを `<div>` でマークアップしてコンテナを構成し、クラスを `is-style-constrained` と指定します。これで P.96 と P.155 のスタイルが適用され、ブロックの横幅と間隔が整います。この `<div>` は、P.95 の Constrained レイアウトタイプの「投稿コンテンツ」ブロックに相当するコンテナと考えることができます。



```
<?php get_header(); ?>

<main>
    <h1><?php the_title(); ?></h1>

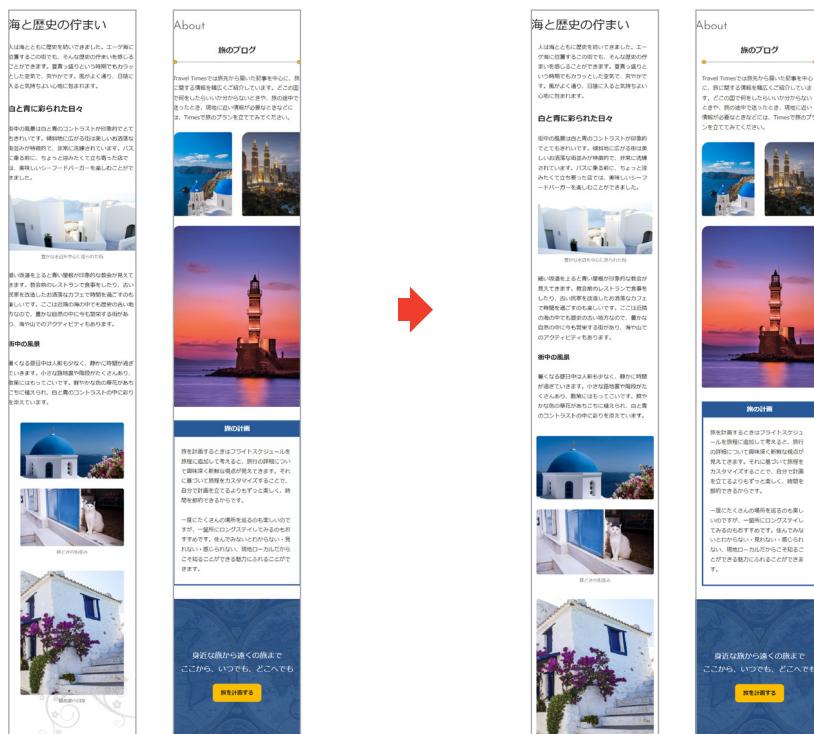
    <div class="is-style-constrained">
        <?php the_content(); ?>
    </div>
</main>
```

`<?php get_footer(); ?>`

hybrid-mytheme/index.php

+ 左右の余白をコントロールする

モバイルの画面では左右に余白が入りません。これは、theme.json で `useRootPaddingAwareAlignments` を有効化しているためです。余白を入れるためにコンテナに `has-global-padding` クラスを追加し、P162 のスタイルを適用します。余白を入れても全幅のブロックは画面の横幅いっぱいに表示されます。



```
<?php get_header(); ?>

<main>
    <h1><?php the_title(); ?></h1>

    <div class="is-layout-constrained has-global-padding">
        <?php the_content(); ?>
    </div>
</main>

<?php get_footer(); ?>
```

Step 3 ブロックの横幅、間隔、左右の余白をコントロールする

+ タイトルの横幅と左右の余白もコントロールする

タイトルの横幅と左右の余白も同じようにコントロールします。`<div>`でマークアップし、`is-style-constrained` と `has-global-padding` クラスを指定します。横幅はコンテンツサイズ（756px）になります。

The screenshot shows the transformation of a page layout. On the left, a narrow design is shown with two columns: '海と歴史の併まい' and '白と青に彩られた日々'. On the right, a wide design is shown with three columns: '海と歴史の併まい', '白と青に彩られた日々', and 'About'. A red arrow points from the narrow design to the wide design. Below the screenshots is the corresponding PHP code:

```
...<main><div class="is-layout-constrained has-global-padding"><h1><?php the_title(); ?></h1></div><div class="is-layout-constrained has-global-padding"><?php the_content(); ?></div></main>
```

hybrid-mytheme/index.php

タイトルの横幅を幅広にするため、`<h1>` に `alignwide` クラスを追加します。

The screenshot shows the 'About' section with the title 'About' and '他のブログ' below it. The title is now wider than in the previous screenshot. Below the screenshots is the corresponding PHP code:

```
...<main><div class="is-layout-constrained has-global-padding"><h1 class="alignwide"><?php the_title(); ?></h1></div>
```

hybrid-mytheme/index.php

+ タイトルとコンテンツの間隔をコントロールする

タイトルとコンテンツの間には余白が入っていましたが、それそれをコンテナの中に入れたことで削除されています。レイアウト機能ではコンテナの外側に不要な影響を与えないようにマージンがコントロールされるためです。



レイアウト機能のスタイルリング方法にしたがってタイトルとコンテンツの間に余白を入れます。そのためには、タイトルとコンテンツの親要素 `<main>` をコンテナにします。ただし、横幅をコントロールする必要はないため、クラスを `is-layout-flow` と指定し、Flow レイアウトタイプにします。

これで P.155 のフクロウセレクタのスタイルが適用され、コンテンツの上マージンで余白が挿入されます。間隔の大きさは `theme.json` で指定した `1.8em` になります。



```
...
<main class="is-layout-flow">
  <div class="is-layout-constrained has-global-padding">
    <h1 class="alignwide"><?php the_title(); ?></h1>
  </div>

  <div class="is-layout-constrained has-global-padding">
    <?php the_content(); ?>
  </div>
</main>
```

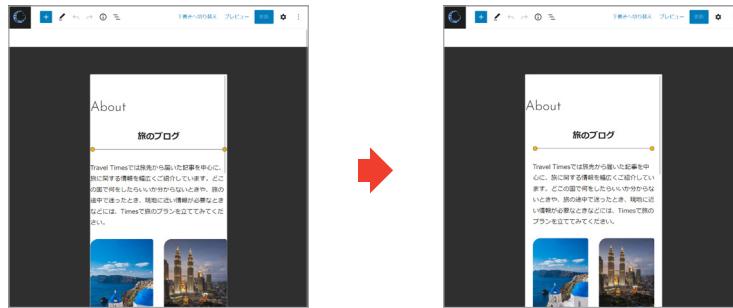
hybrid-mytheme/index.php

Step 3 ブロックの横幅、間隔、左右の余白をコントロールする

◆ 投稿エディターでの左右の余白

ブロックテーマ以外では、フロントと投稿エディターの両方で「投稿コンテンツ」ブロックに相当する Constrained レイアウトタイプのコンテナが存在しない状態になります。

そのため、theme.json の `useRootPaddingAwareAlignments` を有効化すると、投稿エディターにも左右の余白が入らなくなります。余白を入れる場合、コンテンツ全体を内包している `<div class="is-root-container">` に P.162 の③と④のスタイルを適用します。なお、横幅と間隔のスタイルについては、Constrained のコンテナがなくても機能するように処理されます。



```
...
/* 投稿エディターに左右の余白を入れる */
.is-root-container {
    padding-right: var(--wp--style--root--padding-right);
    padding-left: var(--wp--style--root--padding-left);
}
.is-root-container :where(.has-global-padding) {
    padding-right: 0;
    padding-left: 0;
}

.is-root-container > .alignfull {
    margin-right: calc(var(--wp--style--root--padding-right) * -1);
    margin-left: calc(var(--wp--style--root--padding-left) * -1);
}
.is-root-container :where(.has-global-padding) > .alignfull {
    margin-right: 0;
    margin-left: 0;
}
```

hybrid-mytheme/style.css

4

Hybrid Theme

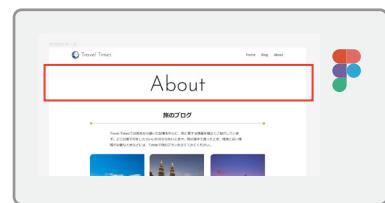


クラスを使ってスタイルを整える

スタイルにはレイアウト機能のクラスだけでなく、P.27 のようにさまざまな形で用意したクラスも利用します。

+ タイトルをスタイルを整える

固定ページのデザインに合わせて、`<h1>` のタイトルをスタイルを整えます。ここではテキストの配置を中央寄せにする `has-text-align-center` クラスと、フォントサイズをプリセットの XXL (xx-large) にする `has-xx-large-font-size` クラスを追加しています。



海と歴史の佇まい

人は海とともに歴史を絆いてきました。エーゲ海に位置するこの街でも、そんな歴史の佇まいを感じることができます。夏まつ盛りという時期でもカラッとした空気で、爽やかです。風がよく通り、日陰に入ると気持ちよい心地に包まれます。

白と青に彩られた日々

街中の風景は白と青のコントラストが印象的でとてもきれいです。特に海に広がる街は美しい
お洒落な街並みが特徴的で、非常に洗練されています。バスに乗る前に、ちょっと済みなくて

About

旅のブログ

Travel Timesでは旅先から届いた記事を中心に、旅に関する情報を幅広くご紹介しています。
どこの国で何をしたらいいか分からないときや、旅の途中で迷ったとき、現地に近い情報が必要なときなどには、Timesで旅のプランを立ててみてください。

海と歴史の佇まい

人は海とともに歴史を絆いてきました。エーゲ海に位置するこの街でも、そんな歴史の佇まいを感じることができます。夏まつ盛りという時期でもカラッとした空気で、爽やかです。風がよく通り、日陰に入ると気持ちよい心地に包まれます。

白と青に彩られた日々

街中の風景は白と青のコントラストが印象的でとてもきれいです。特に海に広がる街は美しい
お洒落な街並みが特徴的で、非常に洗練されています。バスに乗る前に、ちょっと済みなくて

About

旅のブログ

Travel Timesでは旅先から届いた記事を中心に、旅に関する情報を幅広くご紹介しています。
どこの国で何をしたらいいか分からないときや、旅の途中で迷ったとき、現地に近い情報が必要なときなどには、Timesで旅のプランを立ててみてください。

```
<main class="is-layout-flow">
  <div class="is-layout-constrained has-global-padding">
    <h1 class="alignwide has-text-align-center has-xx-large-font-size">
      <?php the_title(); ?>
    </h1>
  </div>
  ...

```

hybrid-mytheme/index.php

+ コンテンツ内のリンクをスタイルリングする

コンテンツ内のリンクの色は P.205 のように theme.json で Secondary (青) にしています。ただし、投稿コンテンツブロック内のリンクのスタイルとして指定していますので、コンテンツのコンテナに `wp-block-post-content` クラスを追加して適用します。



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center has-xx-large-font-size">
            <?php the_title(); ?>
        </h1>
    </div>

    <div class="is-layout-constrained has-global-padding wp-block-post-content">
        <?php the_content(); ?>
    </div>
</main>

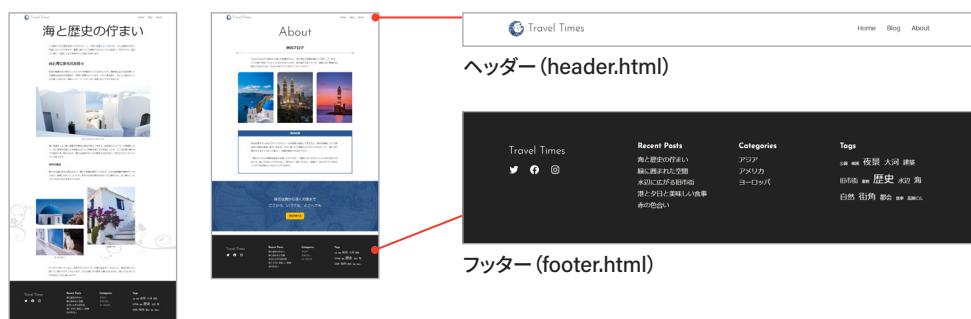
<?php get_footer(); ?>
```

hybrid-mytheme/index.php



ヘッダーとフッターを テンプレートパートで追加する

parts フォルダに保存したブロックベースのテンプレートパートは `<?php block_template_part(); ?>` でファイル名（スラッグ）を指定して利用することができます。ここでは P.207 と P.212 で作成したヘッダー（header.html）とフッター（footer.html）を追加し、`<header>` と `<footer>` でマークアップします。さらに、P.217 のように全体を `<div class="wp-site-blocks">` でマークアップし、ヘッダー、メインコンテンツ、フッターの間隔をコントロールします。



```
...  
<body <?php body_class(); ?>>  
    <?php wp_body_open(); ?>  
  
    <div class="wp-site-blocks">  
        <header>  
            <?php block_template_part( 'header' ); ?>  
        </header>
```

ヘッダーの設定はheader.phpに追加。

hybrid-mytheme/header.php

```
<footer>  
    <?php block_template_part( 'footer' ); ?>  
</footer>  
</div>  
  
<?php wp_footer(); ?>  
</body>  
</html>
```

フッターの設定はfooter.phpに追加。

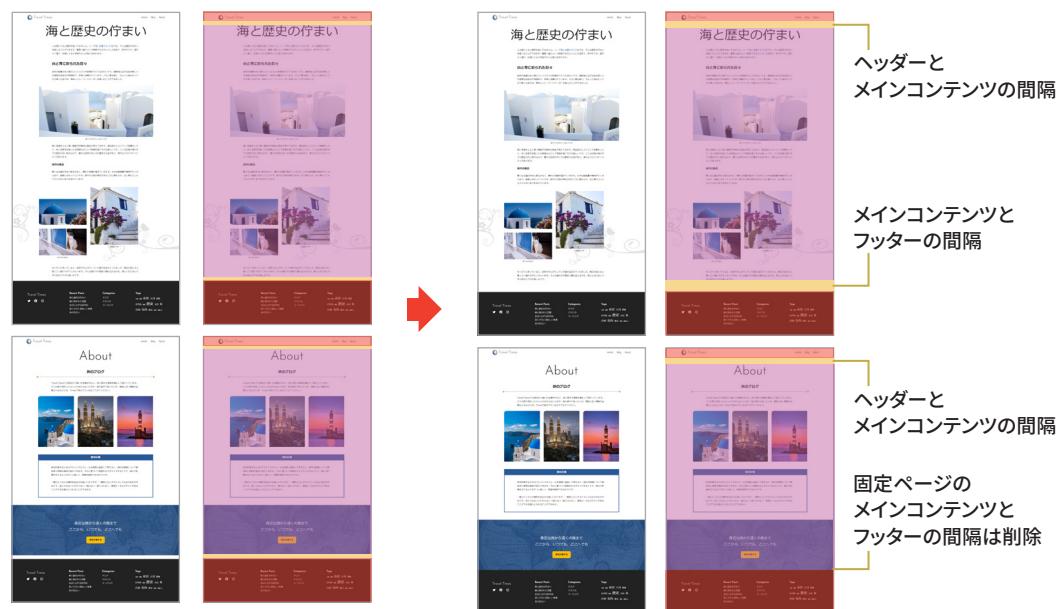
hybrid-mytheme/footer.php

※全体を`<div class="wp-site-blocks">`でマークアップすることで、
上下のルートパディングもP.163のように挿入できます。

+ パーツの間隔をカスタマイズする

<div class="wp-site-blocks"> で挿入される間隔を大きくします。ただし、固定ページのメインコンテンツとフッターの間隔は削除するため、次のようにテーマの style.css で CSS を指定します。

ここではレイアウト機能のスタイル方法にしたがい、P155 と同じクロウセレクタの形式で上マージンを上書きしています。



```
...
/* 間隔をカスタマイズする */
.wp-site-blocks > header + main {
    margin-block-start: var(--wp--preset--spacing--70);
}

.wp-site-blocks > main + footer {
    margin-block-start: var(--wp--preset--spacing--80);
}

.page .wp-site-blocks > main + footer { •—————
    margin-block-start: 0;
}
```

ヘッダー<header>とメインコンテンツ<main>の間隔をプリセットの5(スラッシュ70)に指定。

メインコンテンツ<main>とフッター<footer>の間隔をプリセットの6(スラッシュ80)に指定。

固定ページのメインコンテンツ<main>とフッター<footer>の間隔は削除。

hybrid-mytheme/style.css

Step 5 ヘッダーとフッターをテンプレートパートで追加する

<main>内のタイトルとコンテンツの間隔も大きくします。この間隔はコンテナの<main>でコントロールしていますので、同じようにフクロウセレクタの形式で上マージンを上書きします。なお、タイトルには `mainTitle`、コンテンツには `mainBody` とクラスを追加して設定しています。



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center has-xx-large-font-size">
            <?php the_title(); ?>
        </h1>
    </div>

    <div class="mainBody is-layout-constrained has-global-padding wp-block-post-content">
        <?php the_content(); ?>
    </div>
</main>

<?php get_footer(); ?>
```

hybrid-mytheme/index.php

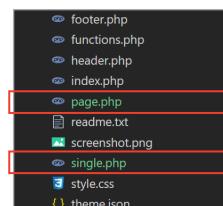
```
/* 間隔をカスタマイズする */
...
.page .wp-site-blocks > main + footer {
    margin-block-start: 0;
}

main > .mainTitle + .mainBody {
    margin-block-start: var(--wp--preset--spacing--70);
```

タイトルとコンテンツの間隔をプリセットの5(スラッグ70)に指定。

hybrid-mytheme/style.css

以上で、主要ページに共通する設定は完了です。index.php をコピーして、記事ページ用のテンプレート single.php と、固定ページ用のテンプレート page.php を作成します。固定ページはここまで設定で完成ですので、次のステップでは single.php を編集して記事ページを仕上げます。



6

Hybrid Theme



ブロックベースの テンプレート-partsを作成する

記事ページには記事のヘッダーとフッターを追加して、投稿日やアイキャッチ画像、前後の記事へのリンクなどを表示します。



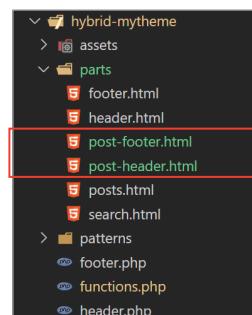
これらはスタイルも含めて、ブロックで構成するのが簡単です。ブロックベースのテンプレート-partsとして作成するため、`add_theme_support()` でテンプレート-partsエディターを有効化します。ただし、このエディターにはテンプレート-partsを新規作成する機能が用意されていないため、parts フォルダにあらかじめファイルを用意しておきます。ここでは `post-header.html` と `post-footer.html` を追加します。

```
<?php

function mytheme_support() {
    ...
    // ブロックベースのテンプレート-partsエディターを有効化
    add_theme_support( 'block-template-parts' );
}

add_action( 'after_setup_theme', 'mytheme_support' );
...
```

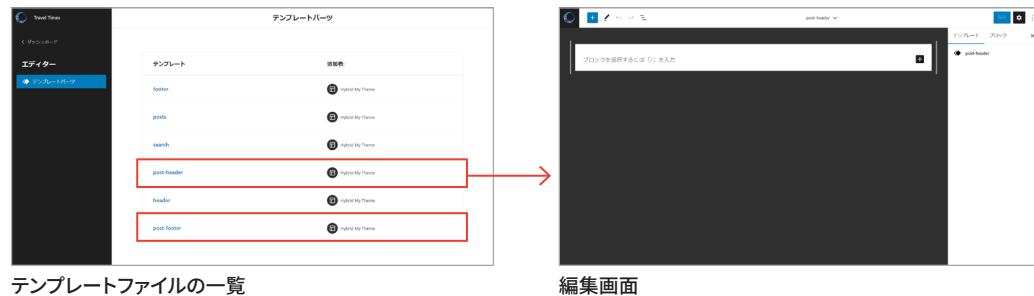
hybrid-mytheme/functions.php



空のファイルを追加。

+ テンプレート-partsを編集する

[外観>テンプレート-parts] でテンプレート-partsエディターを開くと、parts フォルダに保存したテンプレート-partsがリストアップされます。空のファイルを用意した「post-header」や「post-footer」をクリックすると、白紙の編集画面が開きます。



`post-header` は P.223、`post-footer` は P.229 のようにブロックを構成し、スタイルリングして保存します。ここではブロックテーマで作成したものをコピーして追加しています。

post-header

post-footer

+ テンプレートパートの編集結果をテーマに反映する

テンプレートパートエディターでの編集結果は、P.105 のようにデータベースに保存されます。そのため、ブロックテーマのときと同じようにテーマに反映する作業が必要です。ただし、ハイブリッドテーマでは Create Block Theme プラグインを使用できないため、次のように手動で反映します。



post-headerの編集画面で
「すべてのブロックをコピー」を選択。

コピーしたコードをペーストして保存します。

```
<!-- wp:group {"style":{"spacing":{"margin":{"bottom":"var:preset|spacing|70"}}}, "layout":{"type":"constrained"} -->
...
<!-- /wp:group -->
```

hybrid-mytheme/parties/post-header.html



post-footerの編集画面で
「すべてのブロックをコピー」を選択。

```
<!-- wp:group {"layout":{"type":"constrained"} -->
<div class="wp-block-group">
...
</div>
<!-- /wp:group -->
```

hybrid-mytheme/parties/post-footer.html

データベース上のデータをクリアするため、テンプレートパートの一覧で `post-header` と `post-footer` の「カスタマイズをクリア」を選択します。以上で、テンプレートパートは完成です。



テンプレートパートエディターのメニューには「エクスポート」が用意されていますので、P.107 の方法でも反映することができます。

+ テンプレートパートを使う

作成したテンプレートパートを記事ページのテンプレート single.php に追加します。ここでは既存のタイトルを `post-header` テンプレートパートに置き換え、コンテンツの下に `post-footer` テンプレートを追加します。

以上で、記事ページは完成です。



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center has-xx-large-font-size">
            <?php the_title(); ?>
        </h1>
    </div>

    <div class="mainBody is-layout-constrained has-global-padding wp-block-post-content">
        <?php the_content(); ?>
    </div>
</main>

<?php get_footer(); ?>
```



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <?php block_template_part( 'post-header' ); ?>

    <div class="mainBody is-layout-constrained has-global-padding wp-block-post-content">
        <?php the_content(); ?>
    </div>

    <?php block_template_part( 'post-footer' ); ?>
</main>

<?php get_footer(); ?>
```

hybrid-mytheme/single.php

7

Hybrid Theme



記事一覧ページを作成する

index.php を使用して記事一覧ページを作成します。タイトルはサイト名に、コンテンツは記事一覧のテンプレートパート `posts` に置き換えます。「記事一覧」という見出しが `<h2>` でマークアップし、クラスでスタイルリングしています。なお、投稿コンテンツブロックのスタイルを適用する必要はないため、`wp-block-post-content` クラスは削除します。記事一覧ページ (`/blog/`) にアクセスし、右のように表示されたら完成です。



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center has-xx-large-font-size">
            <?php the_title(); ?>
        </h1>
    </div>

    <div class="mainBody is-layout-constrained has-global-padding wp-block-post-content">
        <?php the_content(); ?>
    </div>
</main>

<?php get_footer(); ?>
```



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center has-xx-large-font-size">
            <?php bloginfo( 'name' ); ?>
        </h1>
    </div>

    <div class="mainBody is-layout-constrained has-global-padding">
        <h2 class="alignwide has-text-align-center is-style-decoration-line">記事一覧 </h2>
        <?php block_template_part( 'posts' ); ?>
    </div>
</main>

<?php get_footer(); ?>
```

hybrid-mytheme/index.php

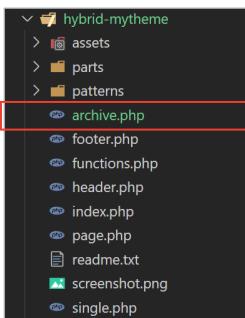
8

Hybrid Theme

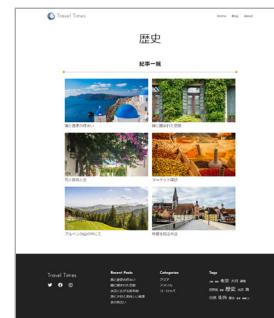


アーカイブページを作成する

アーカイブページを作成します。index.php をコピーして archive.php を用意し、`<h1>` のタイトルを `<?php single_term_title(); ?>` に置き換える、カテゴリー名やタグ名を出力します。さらに、`<h1>` の `has-xx-large-font-size` クラスを削除し、P.149 で指定したベースとなるスタイルのフォントサイズ（x-large）にします。カテゴリーページやタグページの表示を確認したら完成です。



archive.phpを作成

カテゴリーページ
`/category/europe/`タグページ
`/tag/history/`

```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center has-xx-large-font-size">
            <?php bloginfo( 'name' ); ?>
        </h1>
    </div>
    ...

```



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center">
            <?php single_term_title(); ?>
        </h1>
    </div>
    ...

```

hybrid-mytheme/archive.php

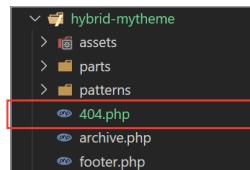
9

Hybrid Theme

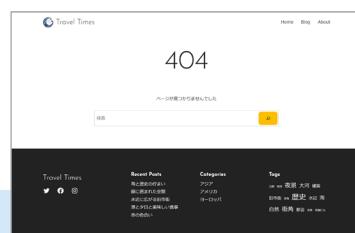


404ページと検索結果ページを作成する

404 ページと検索結果ページを作成します。まずは index.php をコピーし、404.php を作成します。
<h1> のタイトルを「404」に変更し、<h2> の見出しを「ページが見つかりませんでした」というメッセージに置き換えます。さらに記事一覧を検索フォームのテンプレートパート search に置き換えたら、404 ページは完成です。



404.phpを作成



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center has-xx-large-font-size">
            <?php bloginfo( 'name' ); ?>
        </h1>
    </div>

    <div class="mainBody is-layout-constrained has-global-padding">
        <h2 class="alignwide has-text-align-center is-style-decoration-line">記事一覧 </h2>
        <?php block_template_part( 'posts' ); ?>
    </div>
</main>
...
```



```
<?php get_header(); ?>

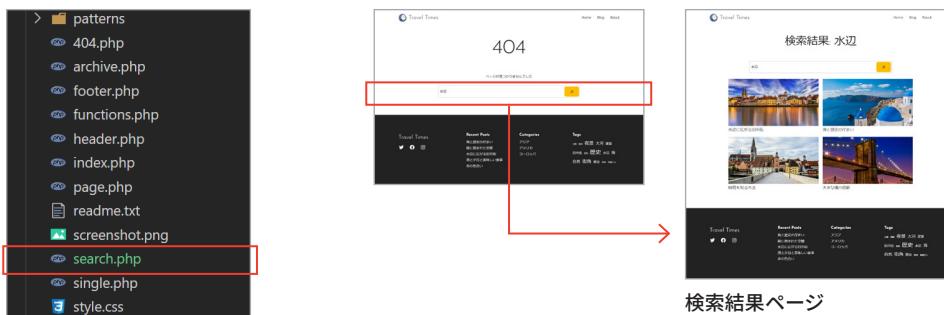
<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center has-xx-large-font-size">
            404
        </h1>
    </div>

    <div class="mainBody is-layout-constrained has-global-padding">
        <p class="has-text-align-center">ページが見つかりませんでした </p>
        <?php block_template_part( 'search' ); ?>
    </div>
</main>
...
```

hybrid-mytheme/404.php

Step 9 404 ページと検索結果ページを作成する

続けて、検索結果ページを作成します。`<h1>` のフォントサイズは `x-large` にしたいので、`archive.php` をコピーして `search.php` を作成します。このページでは `<h1>` のタイトルに検索キーワードを表示します。さらに、`<h2>` の見出しを検索フォームのテンプレートパート `search` に置き換えたら完成です。



search.phpを作成

```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center">
            <?php single_term_title(); ?>
        </h1>
    </div>

    <div class="mainBody is-layout-constrained has-global-padding">
        <h2 class="alignwide has-text-align-center is-style-decoration-line">記事一覧 </h2>
        <?php block_template_part( 'posts' ); ?>
    </div>
</main>
...
```



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center">
            検索結果： <?php echo esc_html( get_search_query() ); ?>
        </h1>
    </div>

    <div class="mainBody is-layout-constrained has-global-padding">
        <?php block_template_part( 'search' ); ?>
        <?php block_template_part( 'posts' ); ?>
    </div>
</main>
...
```

hybrid-mytheme/search.php

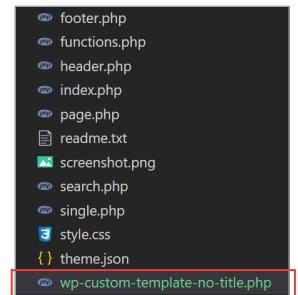
10

Hybrid Theme



タイトルを出力しない カスタムテンプレートを作成する

サイト型のトップページで使用するカスタムテンプレートを作成します。固定ページのテンプレートからタイトルを除いたものを作るため、page.php をコピーして wp-custom-template-no-title.php を作成します。ファイル名はブロックテーマで作成したものに揃えています。タイトルを削除し、カスタムテンプレート名を「no-title」、使用を許可する投稿タイプを「page（固定ページ）」と指定します。



```
<?php get_header(); ?>

<main class="is-layout-flow">
    <div class="mainTitle is-layout-constrained has-global-padding">
        <h1 class="alignwide has-text-align-center has-xx-large-font-size">
            <?php the_title(); ?>
        </h1>
    </div>

    <div class="mainBody is-layout-constrained has-global-padding wp-block-post-content">
        <?php the_content(); ?>
    </div>
</main>

<?php get_footer(); ?>
```



```
<?php
/*
Template Name: no-title
Template Post Type: page
*/
?>
<?php get_header(); ?>

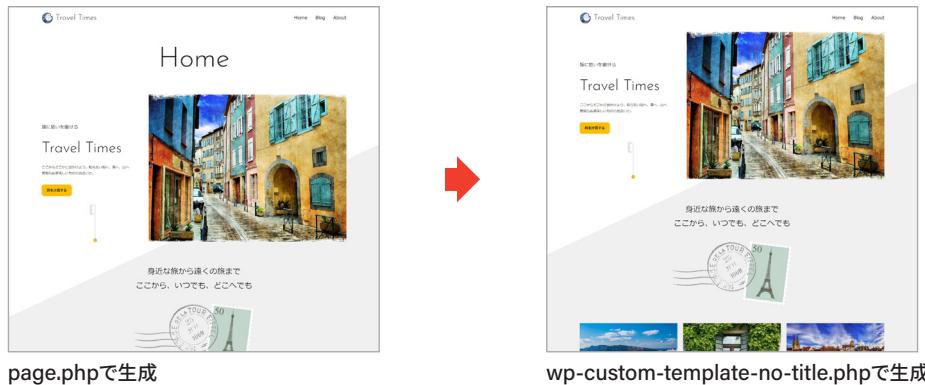
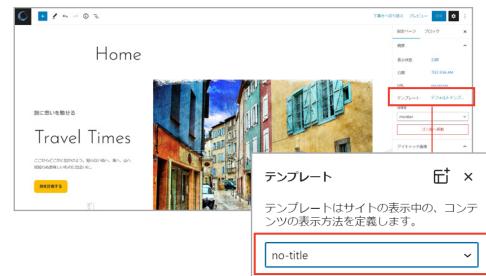
<main class="is-layout-flow">
    <div class="mainBody is-layout-constrained has-global-padding wp-block-post-content">
        <?php the_content(); ?>
    </div>
</main>

<?php get_footer(); ?>
```

hybrid-mytheme/wp-custom-template-no-title.php

Step 10 タイトルを出力しないカスタムテンプレートを作成する

固定ページ「Home」の編集画面を開き、テンプレートを「no-title」にして更新します。これで、page.phpではなく、カスタムテンプレート wp-custom-template-no-title.php で生成されるようになります。



なお、このテンプレートではヘッダー <header> とメインコンテンツ <main> の間隔を小さくするため、次のように上マージンを上書きしています。

```
/* 間隔をカスタマイズする */
...
main > .mainTitle + .mainBody {
    margin-block-start: var(--wp--preset--spacing--70);
}

.page-template-wp-custom-template-no-title-php .wp-site-blocks > header + main {
    margin-top: var(--wp--preset--spacing--50);
}
```

hybrid-mytheme/style.css

11

Hybrid Theme



ハイブリッドテーマの構成

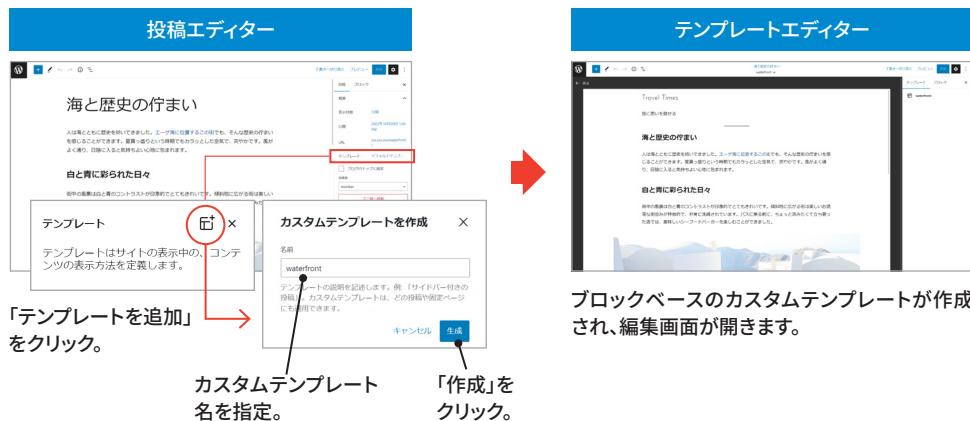
以上で、ハイブリッドテーマは完成です。テーマのファイル構成は次のようにになっています。これから先は、従来のクラシックテーマの機能を使って拡張していくことができます。



Step 11 ハイブリッドテーマの構成

■ ブロックベースのテンプレートエディターを無効化する

ハイブリッドテーマではブロックベースのテンプレートエディターの機能がデフォルトで有効化されています。特定の記事ページや固定ページの生成に使用するブロックベースのカスタムテンプレートを作成・編集することができ、投稿エディターの「テンプレート」から利用します。



ただし、作成したカスタムテンプレートはデータベースに保存され、テーマに反映することも、削除することもできません。誤ってカスタムテンプレートを作成し、ページが想定外の表示になるリスクもあります。そのため、この機能が不要な場合は次の設定を追加して無効化します。

```
<?php

function mytheme_support() {
    ...
    // ブロックベースのテンプレートパーティエディターを有効化
    add_theme_support( 'block-template-parts' );

    // ブロックベースのテンプレートエディターを無効化
    remove_theme_support( 'block-templates' );
}

add_action( 'after_setup_theme', 'mytheme_support' );
...
```

hybrid-mytheme/functions.php

関連書籍



作って学ぶ Next.js/React Webサイト構築

ステップバイステップでマスターする、Next.jsによるWeb制作入門実践書。

- ℳ <https://book.mynavi.jp/ec/products/detail/id=130848>
- ⚡ <https://ebisu.com/next-react-website/>
- ⟳ <https://amzn.to/3RvfR8D>



作って学ぶ HTML&CSSモダンコーディング

モバイルファースト＆レスポンシブなサイト作成を、ステップバイステップでマスターする。デザインを実現するCSSのバリエーションも解説。

- ℳ <https://book.mynavi.jp/ec/products/detail/id=124054>
- ⚡ <https://ebisu.com/html-css-modern-coding/>
- ⟳ <https://amzn.to/2XsZHoU>



HTML5 & CSS3デザイン 現場の新標準ガイド

HTMLとCSSの最新仕様を整理し、制作の現場で必要不可欠な情報をまとめたガイドブック。

- ℳ <https://book.mynavi.jp/ec/products/detail/id=117364>
- ⚡ <https://ebisu.com/html5-css3-practical-design-guide-2/>
- ⟳ <https://amzn.to/378x17B>

■著者紹介

エビスコム

<https://ebisu.com/>

さまざまなメディアにおける企画制作を世界各地のネットワークを駆使して展開。コンピュータ、インターネット関係では書籍、デジタル映像、CG、ソフトウェアの企画制作、WWWシステムの構築などを行う。

主な編著書：『作って学ぶ Next.js/React Web サイト構築』マイナビ出版刊
『作って学ぶ HTML & CSS モダンコーディング』同上
『HTML5 & CSS3 デザイン 現場の新標準ガイド【第2版】』同上
『Web サイト高速化のための 静的サイトジェネレーター活用入門』同上
『CSS グリッドレイアウト デザインブック』同上
『フレキシブルボックスで作る HTML5&CSS3 レッスンブック』ソシム刊
『CSS グリッドで作る HTML5&CSS3 レッスンブック』同上
『HTML&CSS コーディング・プラクティスブック1～8』エビスコム電子書籍出版部刊
ほか多数

作って学ぶ WordPress ブロックテーマ

ハイブリッドテーマ

2023年1月31日 ver.1.0 発行