Acquia
EXPERIENCE DIGITAL FREEDOM

ヘッドレス＆ハイブリッド DRUPAL 入門

どのCMSソリューションが最も効果的かを判断する方法

目次

イントロ

Web開発の現場において、ヘッドレスコンテンツマネジメントシステム（CMS）やデカップルドアプリケーションほど急速に普及しているトレンドは他にありません。

これは、プレゼンテーション層がバックエンド側のCMSに密接に結合していた従来のアプローチから進化したものであることは明らかです。しかし、選択肢が増えたことで、どのように機能するかをより深く理解し、企業の目指すゴールと最も合致するアプローチを選択する必要性が生じています。

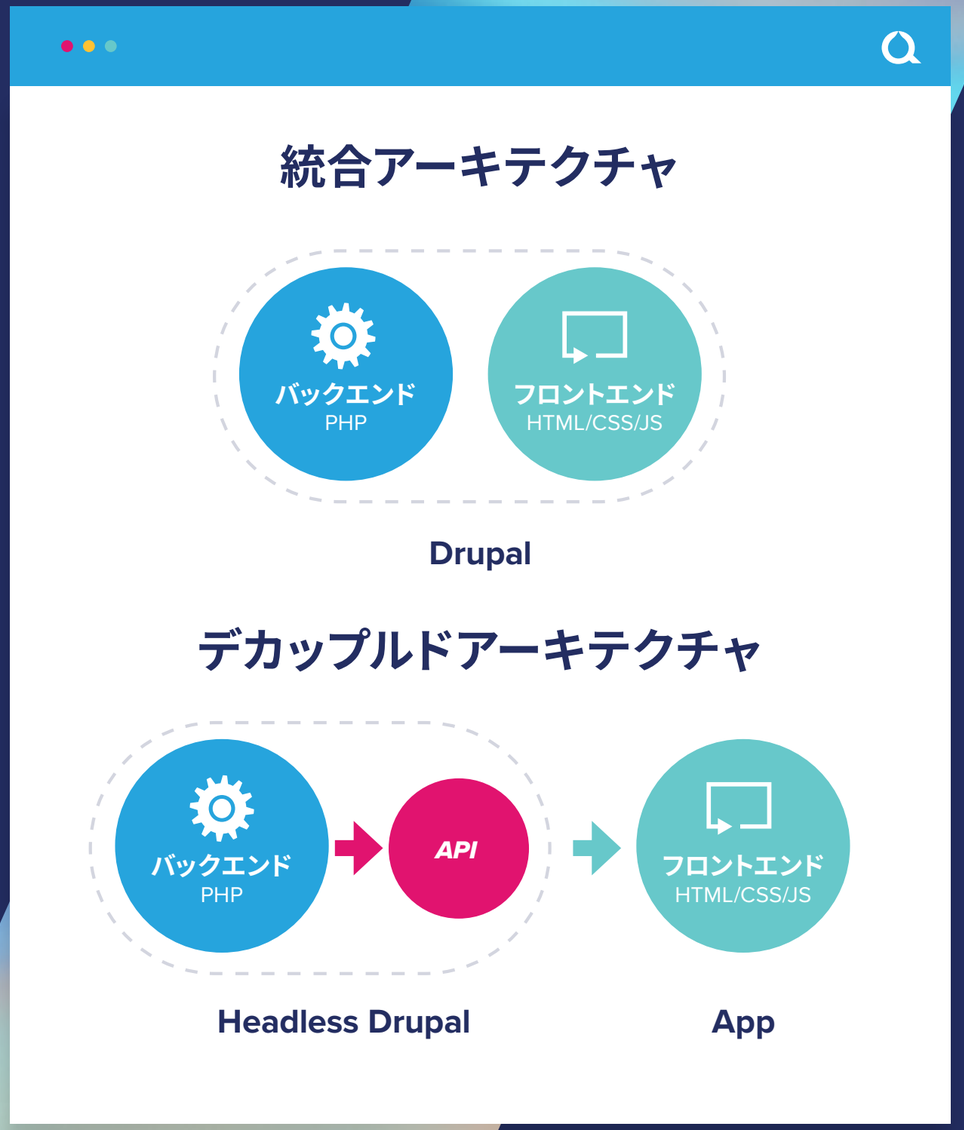
このe-bookでは、デカップリングアーキテクチャの仕組み、デカップリングを検討すべきタイミング、そしてマーケターと開発者の両方がヘッドレスDrupalを活用し、理想的なデジタル体験を提供する方法を解説しています。また、いかにDrupalが他にはない高度な「ハイブリッドヘッドレス」機能を備えているか、についても説明します。

統合アーキテクチャとデカップルドアーキテクチャがどう違うのかを明確にすることから始めましょう。

従来の **統合アーキテクチャ** では、フロントエンドとバックエンドの両方が一つのシステムに収められています。

これにより、CMSはコンテンツを管理するだけでなく、テンプレートやローコードツールを使用して、フロントエンドエクスペリエンスのためのマークアップ（HTML）を表示することができるようになります。このようにツールをパッケージ化することで、複雑な機能を追加することなく、さまざまなWebサイトやアプリケーションに適した **一つのまとまった** システムが実現します。

対して、**デカップルドアーキテクチャ** を使うと、Drupalのバックエンドシステムはコンテンツ作成のためのAPIサービスレイヤを提供します。この場合、開発者はすぐに使えるTwigテンプレートエンジンを使うのではなく、別の技術を利用してフロントエンドエクスペリエンスを表示します。ユーザーが（画面を通して）コンテンツを収集し、接触するデバイスがますます増えていることを考えると、これは非常に理にかなっています。



このモデルでは、DrupalはヘッドレスCMSのリポジトリとみなされ、コンテンツとデータを他のアプリケーションで利用できるように公開します。これには以下のようなアプリケーションが含まれます。

- **ネイティブアプリケーション** は、デスクトップやモバイルなど、特定のデバイスやプラットフォーム向けに開発されます。例えば、モバイルアプリケーションは、特定のスマートフォンやスマートウォッチ向けに開発されることが一般的です。
- **JavaScript アプリケーション** は、サーバーへの非同期型要求を通じてページ更新を動的に行い、ページの完全な再読み込みを防ぎます。つまり、ブラウザがページを再読み込みしなくても、Webアプリケーションがサーバーを呼び出すことができるのです。
- **デジタルサイネージ** は公共施設やレストラン、オフィスなどで目にする、コンテンツを表示する手段として一般的です。通常、外部のAPIからコンテンツや画像を取得して表示するタイプのデジタルディスプレイで、遠隔から簡単に更新・管理することができます。
- **モノのインターネット（IoT）アプリケーション** には、スマートテレビ、Amazon Echo、Apple Watch、またはフィットネストラッカーなどのデバイスが含まれます。この分野は急速に拡大しており、これらのデバイスはコンテンツやデータを外部サービスに依存することが一般的です 。

企業がヘッドレスアプローチを採用する理由

企業がヘッドレスDrupalのアプローチを採用するのには、いくつかの理由があります。Drupalをコンテンツリポジトリとして活用し、複雑になっているデジタルエコシステム内であらゆるデバイスにコンテンツを提供するためにヘッドレス戦略を導入する場合もあれば、フロントエンドチームがDrupalのバックエンド機能をそのまま使用しながら、一般的なJavaScript（JS）フレームワークを使用できるようにするためにデカップリングを採用する場合もあります。例として、多くの企業はヘッドレスDrupalをReact、Svelte、NextJS、VueJSなどのJavaScriptフレームワークと組み合わせて利用しています。

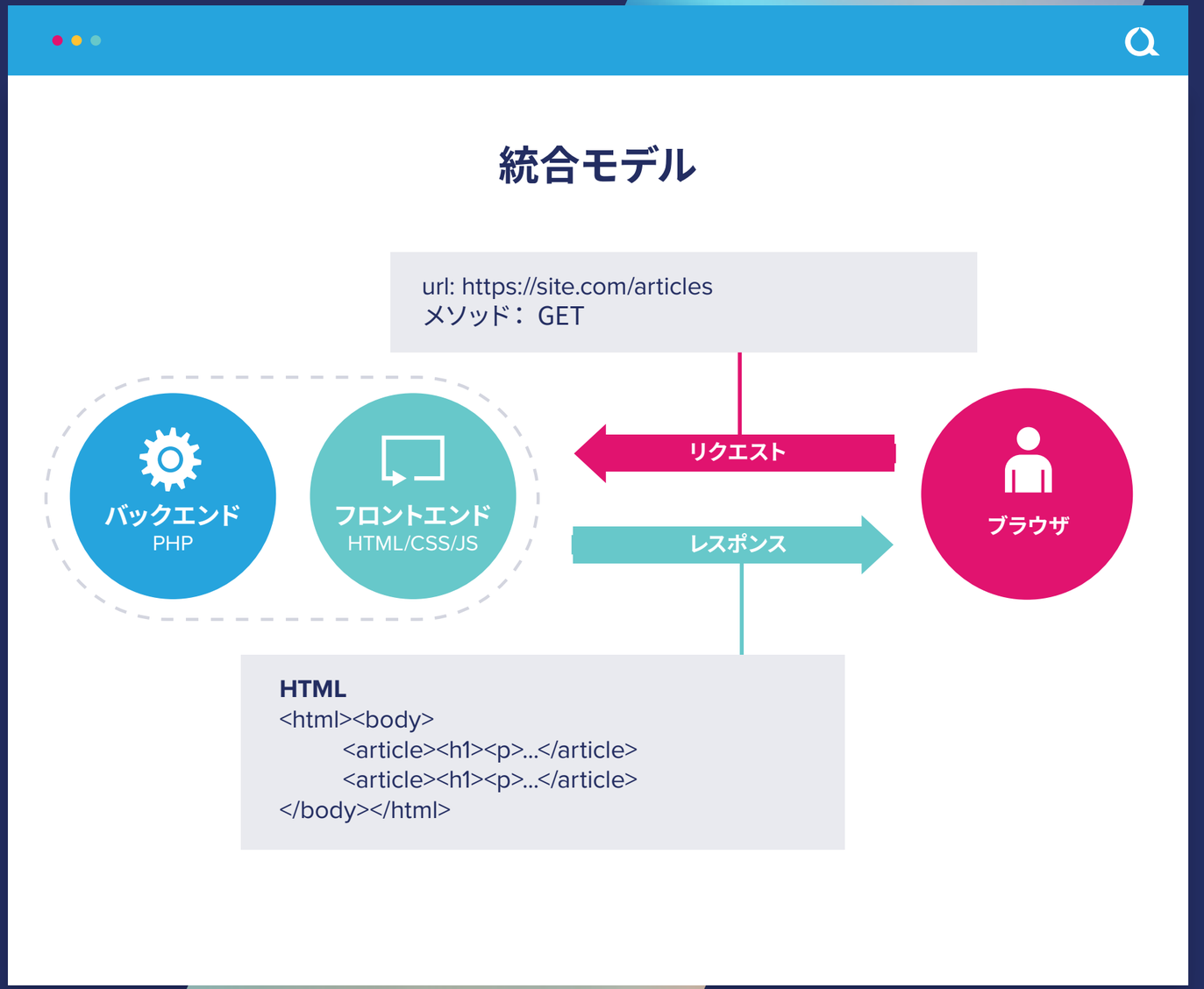
ここで重要なことは、ヘッドレスCMSを採用する理由が何であれ、Web開発者とマーケティング担当者の両方が必要なものを得られるかどうかを見極める必要があるということです。

統合アーキテクチャの仕組み

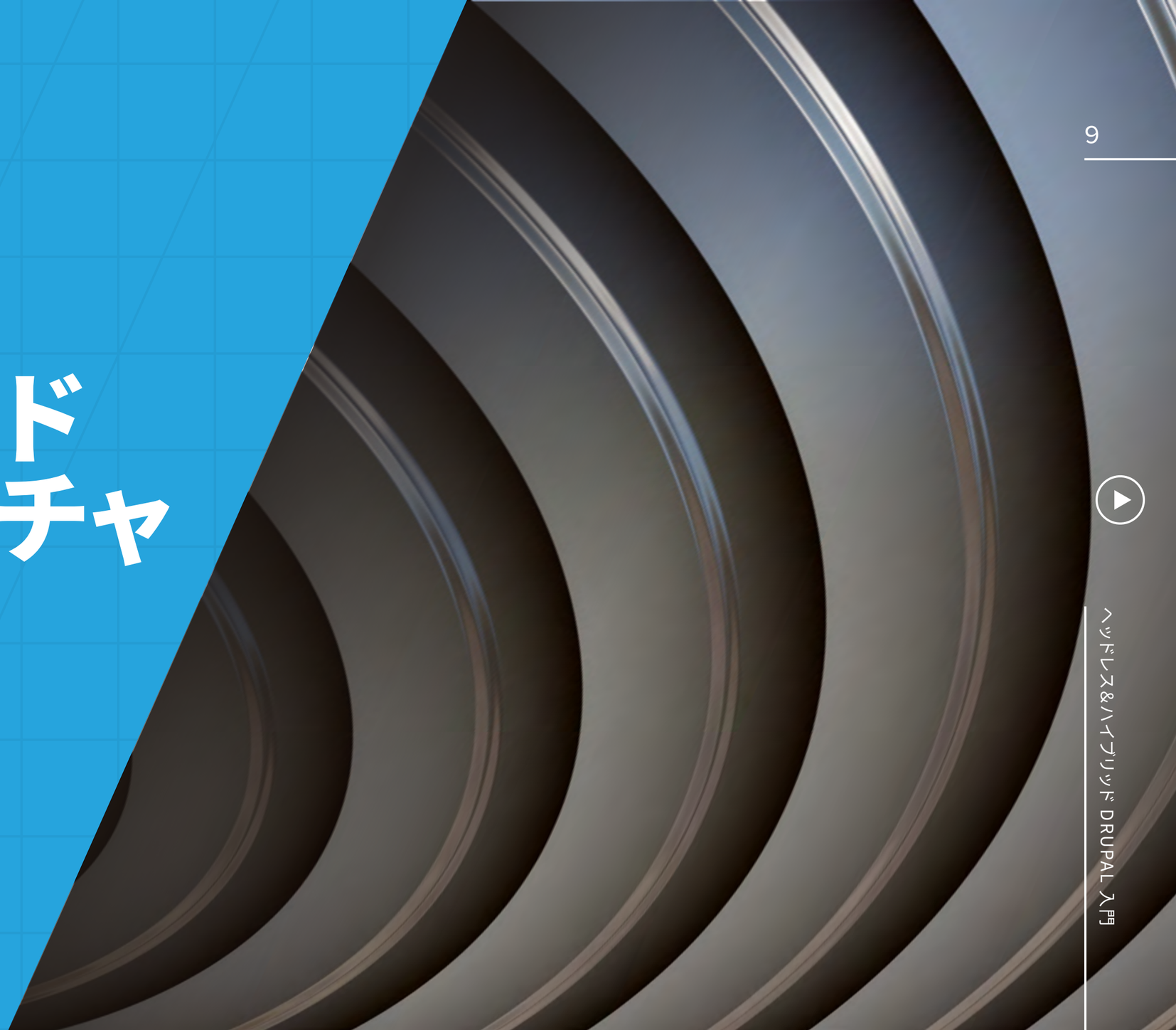
Web上で行われるほとんどのやりとりは、ユーザーがデータを要求し、事前に定義されたHTMLテンプレート内で適切なコンテンツをシステムが収集、フォーマット、表示することによって応えるという、**リクエスト/レスポンス型**　の考え方に基づきます。

これが従来からのインターネットの仕組みです。HTMLのリクエストをブラウザが行い、サーバーがデータを返し、ブラウザがそのページをユーザーに表示する。非常にシンプルで拡張性があり、Webの仕組みの基礎となるものです。

Drupalは統合モデルとして、バックエンドのコンテンツマネジメントシステムとHTMLレンダリングエンジン（Twig）を複合的なフレームワークで提供しています。従来のCMSシステムも似たような機能を持っていますが、大きな違いは **APIファースト** であることと、独立した構成のシステムをベースにしていることです。このため、ニーズに応じて、統合アーキテクチャでもデカップルドアーキテクチャでも使用することができます。



デカップルドアーキテクチャの仕組み

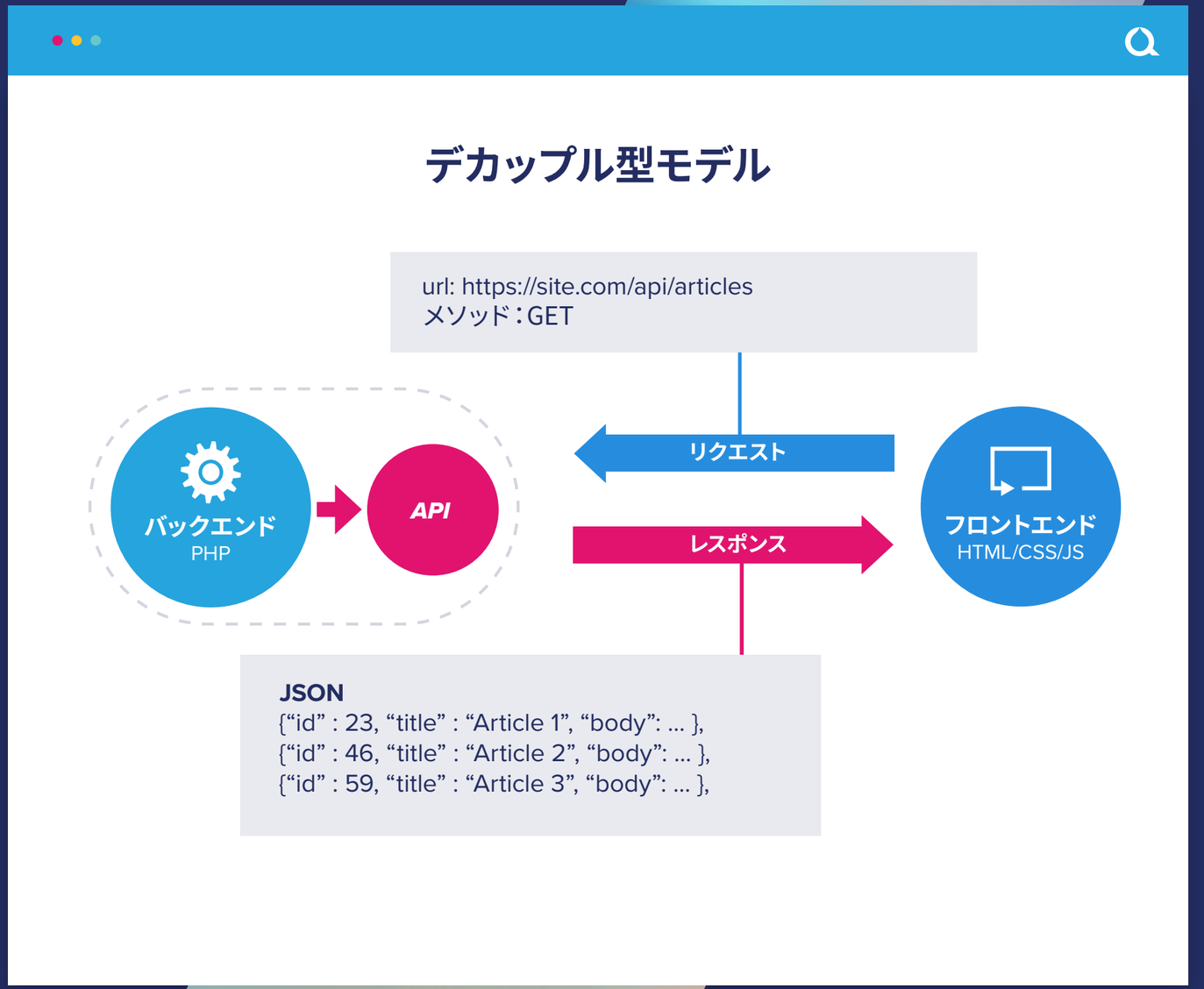


デカップルドアーキテクチャは、基本的には同じアプローチを取りますが、少し違いがあります。単一システムですべてを行うのではなく、複数のシステムで責任を分散させるのです。

デカップルドアーキテクチャでは、Drupalのバックエンドはコンテンツリポジトリとして機能します。このシナリオでは、ヘッドレス型のDrupalリポジトリとデカップル型のアプリケーションは標準的なHTTPメソッドでデータを交換します。アプリケーションがリクエストを行い、APIにパラメータを渡します。それに対して通常JSON形式のレスポンスが返されます。

DrupalはAPIファーストであり、RESTful WebサービスモジュールはDrupalコア（Drupalの最新リリースを定義する標準ベースパッケージ）に含まれています。このモジュールは、Drupalによって管理されるデータの柔軟なカスタマイズと、拡張が可能なRESTful APIを提供します。HTTPレスポンスはJSON、XML、またはその他の形式で出力することができます。

上の図では、REST APIとHTTPクライアントがデカップルドアーキテクチャの媒介役となり、バックエンドとフロントエンドの両方の開発者が好きなフレームワークで作業できるようになっています。Drupalはコア部分で基本的なREST APIと完全に機能するJSON:API形式を提供し、エコシステムではGraphQLや他の形式も利用可能です。



ヘッドレスと
デカップルの違い

このガイドでは、ヘッドレスとデカップルという言葉を明確に使用していますが、この区別はかなりニュアンスが異なり、定義するのが難しい場合もあります。

おおまかに言うと、「ヘッドレス」とは、CMSやAPI経由でデータを提供するサービスを指し、「デカップル」とは、アーキテクチャや フロントエンドアプリケーションそのものを指すことが一般的です。

簡単に言うと、**ヘッドレスCMS** はエンドユーザーに何の表示も提供せず、コンテンツや　データを利用可能にするためのAPIだけを提供します。これにより、どんなフロントエンドエクスペリエンスでもコンテンツを利用することができるようになります。DrupalはAPIファースト（APIのみではなく）であるため、ヘッドレスCMSとしてパワフルで人気のある選択肢です。これはAPIサービスレイヤーを中核としてデザインされていることを意味します。

一般的に　**デカップルドアーキテクチャ**とは、フロントエンドのエクスペリエンスを提供する何かしらのアプリケーションと、コンテンツやデータを提供する一つ以上の API サービスのことを指します。つまり、JSアプリケーション、モバイルアプリ、スマートテレビ、デジタルサイネージなど、ユーザーインターフェイスもデカップルドアーキテクチャに含まれるのが典型的な例です。

ヘッドレスDRUPALのアーキテクチャにより、WEB開発チームは以下のことが可能になります

- **複数のデバイスに対応:**
  柔軟なAPIとWebサービスにより、Drupalはあらゆる場所でコンテンツを配信するための頭脳になる

- **他のフロントエンド技術を活用:**
  Drupalは、Drupal CMSで作成されたコンテンツをReact、VueJS、AngularなどのJavaScriptフレームワークで表示するためのサービスレイヤーとして機能することができる.

- **あらゆるメディアをコントロール:**
  ヘッドレスDrupalは、現在使われている多くのメディアに動画やデータを送信するための中枢として機能させることができる.

- **複数のシステムとの統合:**
  バックエンドにDrupalを導入することで、既存のシステムのサポートが可能になる

バックエンドWEB サービスの選択肢

今日、Drupalは様々なWebサービスモジュールを提供しており、誰でも簡単にDrupalからデータを利用できるようになっています。これらには、現在コアで利用可能なモジュールに加え、コミュニティモジュールも含まれています。

- **RESTful Webサービスのコア:** Drupalは標準でREST APIを装備しています。これには、コンテンツエンティティの作成、読込、更新、削除（CRUD）、および設定の読込を行うための操作が含まれます。また、核となるRESTモジュールには、**Serialization、RESTful Web Services、HAL、Basic Authの四種類があります。**Core RESTは、広範な機能を提供しながらも、限られた設定しか必要としません。

- **JSON:APIのコア:** **JSON:API** は共通のフォーマットとして開発者に採用され、複雑なデータでもしっかりとサポートすることから人気が高まっています。これはJSON形式を使用したREST APIの仕様であり、コアサービスレイヤーを上回る機能を提供します。

- **GraphQL:** 元々Facebookによって開発されたもので、クライアントに合わせたクエリを行うために開発されたクエリ言語です。GraphQLはヘッドレスDrupalを利用するクライアントが、バックエンドから簡単にデータを引き出すことを可能にし、一度のリクエストでカスタムセットのデータを取り出せるようにします。なお、**DrupalはGraphQL** モジュールをサポートしています。

- **ローコードでのクエリビルダー:** Drupalには **Views**というコアモジュールがあり、それを使ってコンテンツを取得し、ユーザーに表示するコンポーネントを構築することができます。このダイナミックなクエリービルダーを使用すると、UIで完全に管理されたRESTエンドポイントが提供されます。これは、コードを書かずにカスタムエンドポイントを作成、管理するための優れた方法です。

- **カスタムコード:** Drupalのサービスレイヤーは最初からプラグイン可能で、かつコンポーザブルに設計されています。これは、たとえ非常にユニークなサービスの統合を提供する場合でも、必要に応じてDrupal上で構築できることを意味します。

  このカスタマイズは、ゼロから始めるのではなく必要な機能をCMSに追加するだけなので、他のどんなカスタムメイドのアプローチよりもずっと速く、簡単で、安定したものになるでしょう。

フロントエンド SDKの選択肢

Webサービスの安定したコレクションに加え、Drupalはソフトウェア開発キット（SDK）のエコシステムを提供し、別の技術によるアプリケーション開発の加速を後押ししています。このSDKは、DrupalがPHP以外の言語でも使用できるようにするものです。

従来Drupalのコンテンツを消費するには、REST APIの実装についてある程度の理解が必要で、カスタムアプリケーションの開発も必要でした。Drupalは完全なコンポーザブルCMSとして、一般的なフレームワークとアプリケーション開発プラットフォームのサポートを提供し続けています。これは、多くの選択肢がある中で、導入したその日に動くアプリケーションを手に入れることができる、ということなのです。

Drupal State は、DrupalやJSON:APIのスペックに関する知識が限られているJavaScript開発者でも、最大限に機能を活用できるような、一連のツールを揃えています。

フレームワークにとらわれず、Drupal Stateは通常のJavaScriptやその他の一般的なJavaScriptフレームワークと一緒に使用することができます。また、モジュール化、拡張性、書き換え可能であることを目指し、DrupalのJavaScriptエコシステムをサポートするものです。

- **Next.js for Drupal**（Next-Drupal）は、Drupalのフロントエンドを Next.jsに置き換え、同時にDrupal の主要な編集機能も維持する、両者の良いとこ取りをしたようなシステムです。Drupalの編集作業においてNext.jsのサイトを即座にプレビューすることができ、複雑なモジュールや煩わしいTwigのテンプレートに悩まされることなく、カスタムコンテンツアーキテクチャを作成。さらにはマルチサイトやマルチビューを支える機能も搭載されています。

- **Gatsby** はReactとGraphQLで構築された静的サイトジェネレータで、Reactアプリケーションの構築に伴う拡張性とパフォーマンスに関する問題を軽減します。バックエンドでは、JSON:APIモジュールとともにDrupalのコンテンツ管理、作成、編集ツールを活用し、Gatsbyのフロントエンドにコンテンツを提供することで、高価なエンタープライズCMS劣らないレベルのCMSが、オープンソースのため無料で手に入れることができるのです。

- **Druxt.js** は、NuxtJSとDrupalという2つのフレームワークの間にあるオープンソースのブリッジで、Drupal独自のエンティティ／フィールドの表示システム、ブロックリージョン、ビューなどを活用できるようになっています。DruxtJSは、DrupalのJSON:APIクライアントとVuexキャッシングをサポートし、DrupalのVue.jsテーマレイヤーのように動作します。Druxtコンポーネントは、Vue.jsのスロット、$attrs、propsと一緒にWrapperコンポーネントを使用してテーマ化することができます。

APIの枠を超える

現在の一般的なヘッドレスCMSプラットフォームは、APIのみのアプローチをとっています。デカップルドアーキテクチャを利用すれば、単純にCMSにデータ入力するためのAPIとバックエンドを持つことができ、フロントエンドのエクスペリエンスはありません。

Webサイト、アプリ、その他のオムニチャネル体験など、完全なエクスペリエンスを作り上げるために必要なコードは、自分たちで書かなければなりません。このため、デカップルドアーキテクチャはしばしば「ハイコード」と呼ばれます。フロントエンドのエクスペリエンスのためのコードを書き、それをメンテナンスするのは、開発者に依存することになります。統合アーキテクチャは、開発者でなくてもエクスペリエンスを組み立てることができるUIツールが提供されるため、よく「ローコード」と呼ばれます。

アプリケーションによっては、ハイコード・アプローチは許容範囲であり、むしろ望ましいとさえ言えます。モバイル、Web、IoTアプリケーションを構築するときなど、エクスペリエンスとコードを結びつけるべき特定のタイミングがあるからです。開発者がエクスペリエンスを管理することで、フロントエンドの構築と保守を最も効率的に行えるようになることが多いのです。

しかし、ビジネスユーザーを後押しするローコードツールなど、これまでとは異なるニーズがある場合、これは大きな制約になります。このモデルではあらゆる種類のカスタムレイアウトを実装するために開発者が必要となり、一般的に市場投入までの時間を遅くしてしまいます。さらに問題なのは、APIのみのヘッドレスCMSプラットフォームでは、サイトのアーキテクチャと管理に関して制限があるため、より高額な料金を請求される傾向があることです。

例えば、コンテンツタイプやフィールドの数、ユーザー数、あるいはカスタムロールや権限などの制約を受ける可能性があります。そのプラットフォームの利用が進むほど、料金が高くなる可能性があるのです。

一方DrupalはAPIファーストのアプローチを採用しており、APIは常に利用可能ですが、必須ではありません。

コンテンツの種類、データの複雑さ、インテグレーション回数、ユーザー数、さらにはロールやユーザーに関する高度なオプションに至るまで、一切の制限を設けていません。

DrupalはヘッドレスCMSプラットフォームのすべての機能を、制限なく提供します。また、各プロジェクトに適したアーキテクチャを選択することも可能です。

統合アーキテクチャは必要に応じ、サイトを構築するために必要なツール、テンプレート、その他のレンダリングオプションを無制限に提供するという、さらに一歩進んだサービスを提供します。これは非常に大きなメリットであり、**Drupalを選択すれば、最高のものを手に入れることができる** のです。必要に応じて同じプラットフォーム上で統合アーキテクチャとデカップルドアーキテクチャを使い分けることや、場合によっては同時に使用することも可能です。これがハイブリッドCMSのアプローチであり、自由度と柔軟性が最も高い方法なのです。

正しいアプローチがわかるたった一つの質問

デカップルドアーキテクチャには多くの誇大広告があることも事実なので、プロジェクトに着手する前に、よく分析することが重要です。Drupalをコンテンツリポジトリとして使用し、複数のアプリケーションにサービスを提供する場合は、デカップルドアーキテクチャが適している可能性があります。基本的には、ある簡単な質問をすることでこれを判断することができます。それは、 **エクスペリエンスの組み立ては誰が行うのか?**ということです。

開発者が担当する場合＝ハイコードでの構築

ITチーム（開発者）が主導しコーディング中心で体験設計をする場合は、デカップルドアーキテクチャを用いたハイコードアプローチが最適かもしれません。このアプローチでは、コンテンツ作成を効率化するために、ビジネスユーザーには必要最低限のツールしか提供せず、開発者にはコンテンツ表示用のテンプレートやコードを構築する最大の権限が与えられます。また、これらのエクスペリエンスとコードは一括して管理され、デプロイされます。

デカップルドアーキテクチャには、いくつかの利点があります。

課題の分離
Drupalをコンテンツリポジトリとしてのみ利用することで、課題を分離できるようになります。デカップルドアーキテクチャでは、コンテンツの取り扱いはバックエンドに限定されます。つまりバックエンドとフロントエンドは分離され、フロントエンドはコンテンツの表示とエンドユーザーへの配信にのみ対応します。

パイプライン型開発
このように開発を分離することで、バックエンドとフロントエンドの両チームがそれぞれ別の開発スピードで作業することが可能になります。フロントエンドの開発者はマークアップとレンダリングを自由にコントロールし、バックエンドの開発者は強固なAPIの開発に力を注ぐことができます。

効率的なコンテンツ管理
エクスペリエンスの管理責任を開発チームに移すことで、ビジネスユーザーはコンテンツの構築だけに集中できるようになります。コンテンツの作成と管理に必要な業務が減ることで、CMSはよりシンプルでスピーディにコンテンツ作成のワークフローを提供できるようになります。そのコンテンツが高いレベルで標準化されている場合、いっそう効率的な作業が可能となり、その効果は絶大です。

しかし、フロントエンド全体が別のアプリケーションでコントロールされている場合、エンジニアは多くの人が評価するDrupalの機能を活かすことができないのです。

このように完全に切り離す方法では、Drupalの特徴であるインプレイス編集やディスプレイ管理などを無効にしてしまいます。また、障害の原因となる要素を増やし、結果として技術的な問題を引き起こす可能性があります。

その他のリスクとしては、以下のようなものがあります。

- クロスサイトスクリプティング対策や入力データのサニタイズがされない（特にフレームワークを使用していない場合）
- ビジネスユーザー向けのレイアウトや表示管理ができない
- プレビュー可能なコンテンツのワークフローがない
- フロントエンドに影響を与える既成のモジュールや統合をしない
- システム通知、エラー、メッセージがない
- BigPipeのプログレッシブ読み込みや高度なキャッシュ戦略が無い
- OOTBアクセシブルマークアップや ユーザー体験のメリットがない
- OOTBによる多言語管理、言語切替ができない

マーケターが管理する場合 ＝ローコードでの構築

一方、マーケティングチーム（非開発者）が主導するコンテンツ主導型の体験設計であれば、統合アーキテクチャを用いたローコードアプローチが最も効果的でしょう。この場合、ビジネスユーザーが表現をコントロールできるようにセルフサービスツールが与えられる一方で、開発者は機能、デカップリングされたコンポーネントの構築、そして並行した作業を行えるようになります。つまり、エクスペリエンスとコードは別々に管理されるのです。

Drupalによる統合アーキテクチャには、多くの**優れた点**があります。

ローコードツール

一番大きなメリットは、開発者でなくてもローコードツールを使ってコンテンツの組み立てや作成を行えることです。つまり、ドラッグ&ドロップのUIを使用してコンテンツを「構築」し、固有のニーズに合わせて調整することができるのです。開発者にテンプレートの構築を依頼する代わりに、ビジネスユーザー自身が、求めるエクスペリエンスをできるだけ早く作り上げることができるのです。

エクスペリエンスプレビューとワークフロー

ローコードツールでは、コンテンツ制作者が必要なページやエクスペリエンスを目で見ながら構築することができます。このようにフィードバックが即座に得られるため、制作スピードが向上し、何日どころかたった数時間で公開の準備をすることが可能になります。また、ワークフローツールは、コンテンツの承認プロセスにおけるガバナンスを強化するだけでなく、承認者にも同様のプレビューオプションを提供できることを意味します。

革新的なデジタル体験

ローコードツールは、美しい静的コンテンツを組み立てるだけでなく、ダイナミックで高度なコンポーネントをクリエイターが組み立てることもできるのです。開発者が提供するこれらのコンポーネントは、多くの場合デカップルドアプリケーションと同じ部品を、再利用可能な小さな部品に分割したものです。これにより、開発者でなくても、デカップルドアプリケーションと似たような機能を持つ、魅力的でモダンなエクスペリエンスを作成することができます。

これらのメリットがあるからこそ、Drupalは多くのマーケターに愛されているCMSなのです。しかしその一方で、責任も伴います。ローコードアプローチを採用する場合、コンテンツ作成プロセス自体のガバナンスと管理に、より多くの思考と労力を割く必要があります。システムの拡張や修正が簡単にできるため、大きくなりすぎる可能性があるのです。

より複雑な設定項目を管理し、CMSが肥大化し非効率的なシステムにならないように注意することが課題と言えるでしょう。CMSの成長と標準化のバランスを取るには、統制と技術的な指導が必要な場合が多いのです。

どちらのアプローチも有効

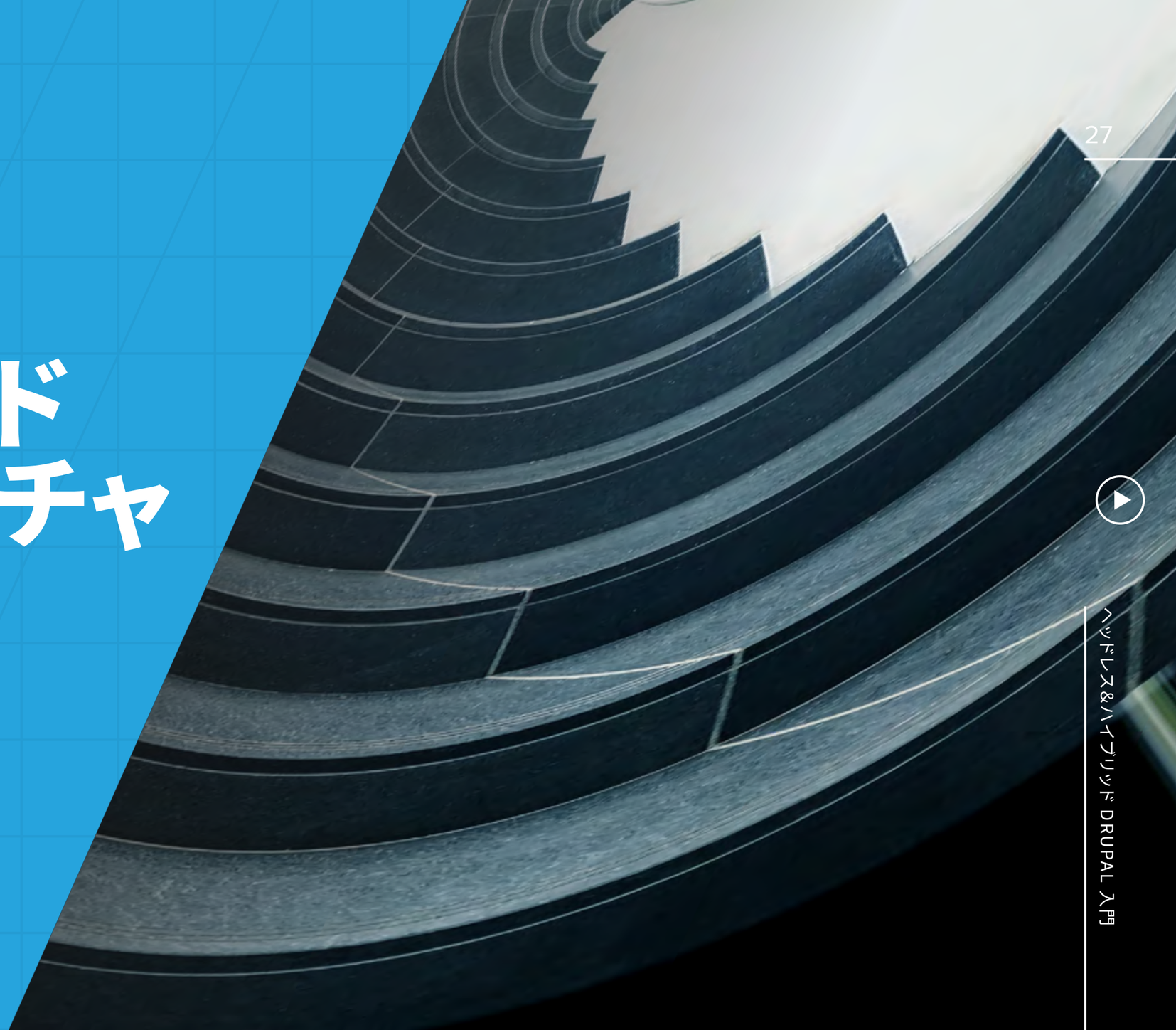
どちらのアプローチにもメリットとデメリットがあり、「正しい」方法というのは存在しません。そのため、体験設計を誰が主導するのか、変更はどのように行われるのか、体験設計をコードのデプロイメントから分離する必要があるのか、そしてそのプロジェクトのニーズは何かなどを明確にすることが重要です。

いずれにせよ、DrupalはAPIのみのツールとは異なり、あらゆるアーキテクチャに対応できるため、企業にとって最適なCMSと言えるでしょう。実際、Drupalはマルチサイト管理をネイティブでサポートし、柔軟なプラットフォームであるため、一つのコードベースから異なるニーズを持つ複数のアプリケーションに対応することが可能です。

各アプリケーションは、共通のコードベースから必要に応じて特定のモジュールや機能を有効にするだけでよいのです。

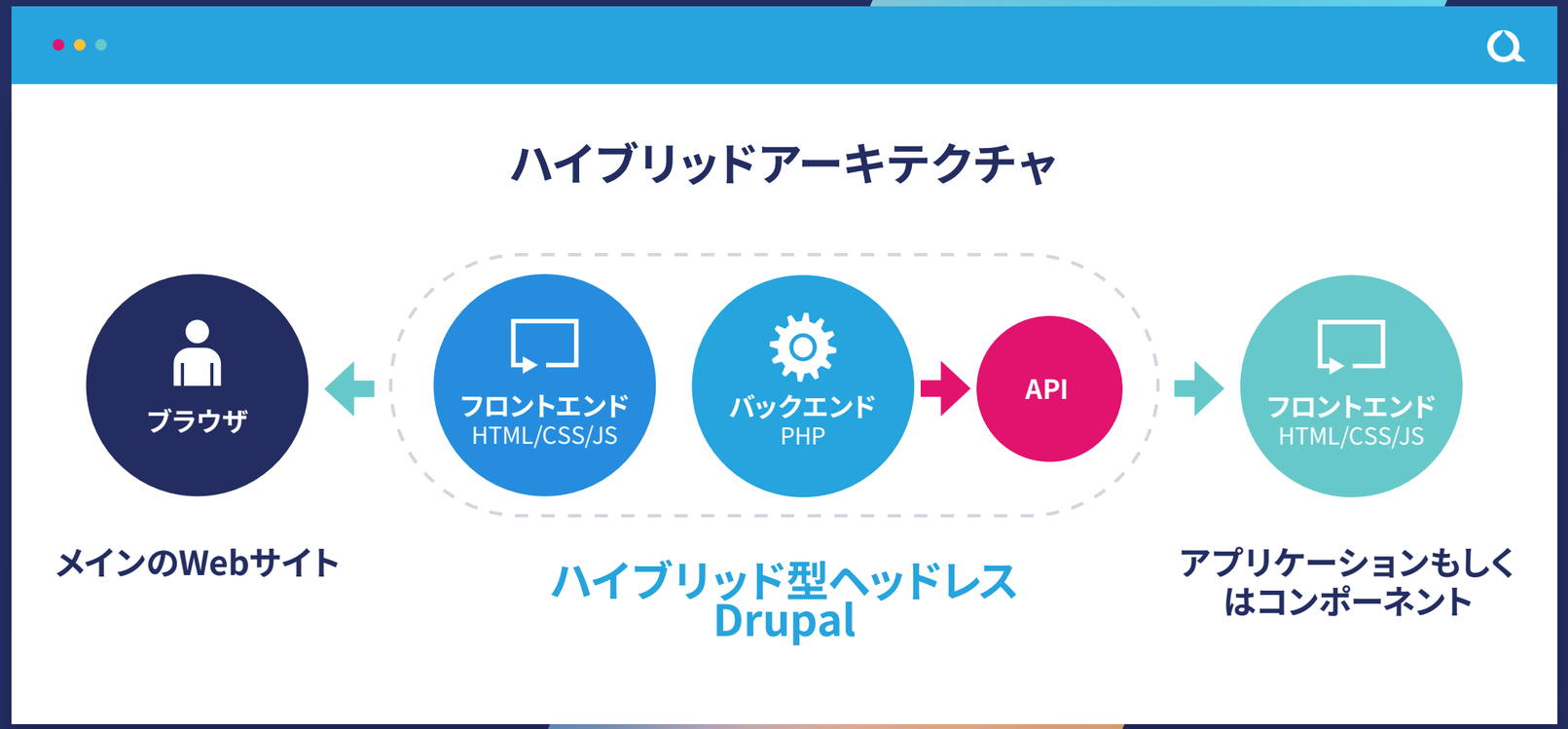
複数のやり方で組み替えられる基本ツールを持つことは、組織にとって大きなメリットになります。アーキテクチャに縛られることなく、同じツールをさまざまな方法で再利用し、さまざまな問題を解決することができるのです。Drupalがしばしば「未来志向」と言われるのはこのためで、業界の変化に合わせて進化できるように設計されているからです。

ハイブリッド アーキテクチャの柔軟性



これまで、私たちは主に統合アーキテクチャとデカップルドアーキテクチャに焦点を当てて説明してきました。実際、CMSの統合アーキテクチャを引き続き採用することは、多くのサイトやアプリにとって十分妥当な選択肢です。しかし、もし自社のニーズがDrupalの標準的な機能を超えると思われる場合、完全な **ヘッドレスかハイブリッド型** のどちらかを選択すべきでしょう。統合アーキテクチャかデカップルドの二つしかないと考える人も多いのですが、実は三つ目の選択肢があります。それが、近年Drupalユーザーの間で人気が高まっている、ハイブリッドアーキテクチャというものです。

完全なるデカップルのアプリケーションにおけるリスクは、Drupalが提供する多くの機能と特徴を統合モデルで再現し、管理しなければならないことです。これは費用の増加、開発サイクルの長期化、開発者でないユーザーへのサポートの低下につながります。これが、多くの組織がDrupalをハイブリッドCMSとして活用することを検討している理由です。



Drupalをハイブリッドアーキテクチャで使用する場合、統合されたシステムとヘッドレスCMSの長所の両方を活かすことができます。これはDrupalがAPIファーストであるためです。つまりAPIサービスレイヤは、すべての表示ツールと一緒にDrupalコアで利用することが可能なのです。このハイブリッドアプローチは、高度な機能の実装を求められる開発者チームと、ユーザー体験を大切にするマーケティングチーム、どちらにニーズも求められる現代の企業にとって、理想的な選択肢であることが多いのです。

ひとたびハイブリッドアーキテクチャに着手すると、状況はより複雑になります。なぜなら、2つの選択肢ではなく、さまざまな選択肢と可能性が広がっていることに気がつくからです。このような自由度の高さが、開発者がDrupalを最も好きな理由の一つなのです。

ハイブリッドアーキテクチャの実装には、通常二つのアプローチがあります。

1. 内蔵型デカップリングコンポーネント

Drupalをローコードの組み立てツールとして使うことで、開発者でなくてもコンテンツを作成することができ、様々なコンポーネントを組み合わせて体験設計し、そしてブラウザに送信する形で表示させることができます。また、Drupalは他のCMSのアセットと一緒に組み立てられるよう、切り離されたコンポーネントライブラリを追加する機能も備えています。

これは既存のWebコンポーネントや、他のJSフレームワークでサポートされているコンポーネントなどです。これらのコンポーネントは高度なインタラクションオプションを提供したり、あるいはミニアプリケーションとして、JSアプリケーションのようなWebベースのサービスと連携することができます。実際、これらのコンポーネントはDrupalのAPIレイヤーを使ってヘッドレスCMSのようにシステムと対話したり、eコマース決済のような外部システムに接続したりすることもできます。

2. サテライト型デカップリングアプリケーション

その他の主なアプローチとしては、メインサイトではDrupalを統一したアーキテクチャで動作させ、API経由で提供されるコンテンツを「サテライト」アプリケーションで利用するという方法が挙げられます。これは、これまで述べてきたようなデカップリングアプリケーションのどれでもかまいません。例えば、JSフレームワーク、モバイルアプリ、スマートTV、その他のIoTアプリケーションなどです。

この柔軟性が、開発者や業界アナリストがこぞってハイブリッドアーキテクチャを支持する理由です。このアーキテクチャは、統合アプリケーションのツールとサポートにより、ヘッドレスサービスが保証する順応性を実現するのです。

プログレッシブデカップリングの利点

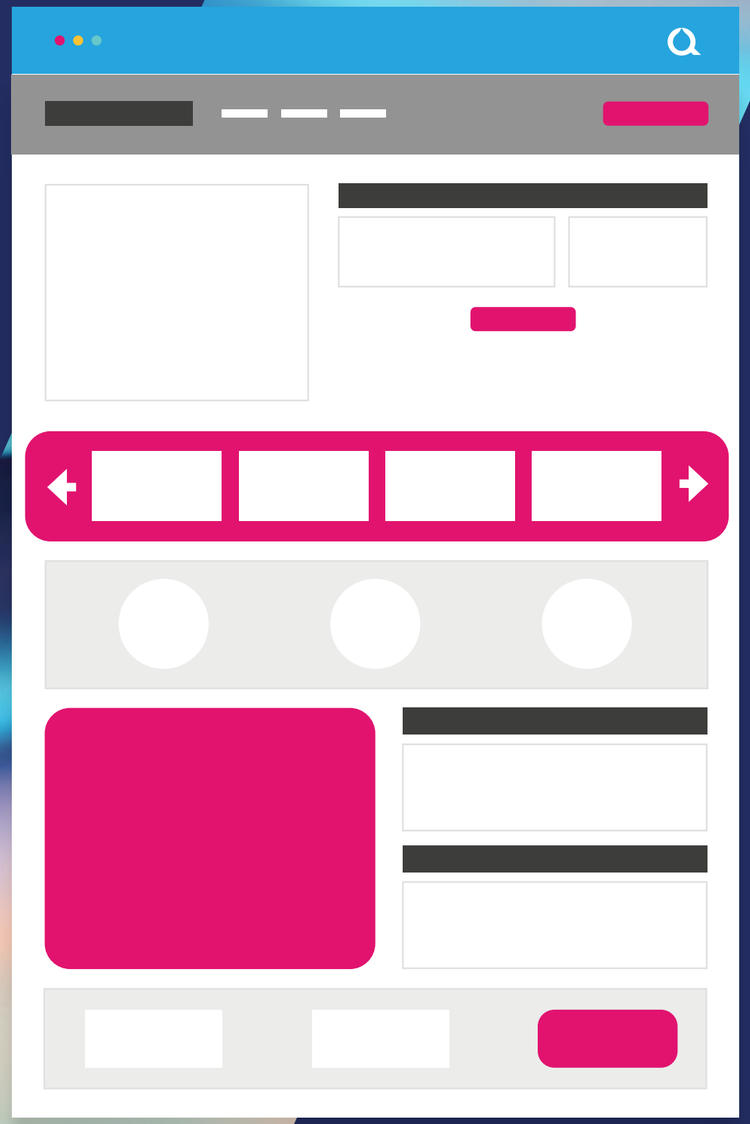
完全にDrupalを切り離すことのリスクを軽減するソリューションとして人気が高まっているのが、**プログレッシブデカップリング**という戦略です。デカップルドアーキテクチャとは異なり、ハイブリッドの導入では、デカップルドコンポーネントとしてJavaScriptフレームワークをDrupalサイトのフロントエンドに組み込みます。JavaScriptフレームワークは引き続きREST APIを使用しますが、一部のコンテンツはDrupalで制御・表示し、他のコンテンツはクライアント側で処理を行うことができます。

このハイブリッドなアプローチにより、編集チームと技術チームがどちらも Drupalの優れた機能を利用し続けながら、JavaScript によるフロントエンドのエクスペリエンスを実現することができます。これらのデカップリングされたコンポーネントは、必要に応じて段階的に作成、実装されます。そのため、統合アーキテクチャかデカップルドアーキテクチャなのかにこだわることなく、サイトの一部を切り離すことができるのです。

また、ビジネスユーザーは開発者の力を借りることなく、ローコードツールを使ってコンポーネントを組み立てたり、並べ替えたりすることができるのです。

プログレッシブデカップリング戦略により、企業は特定のページをさまざまな方法で管理することができるようになります。

この場合、ピンクで示したデカップリングコンポーネントは、必要に応じてページ単位で管理することができます。アプリケーションは、特定のページのどの部分を切り離すかを、操作性のニーズに応じて選択することができます。



事例

ジョージア州技術局

ジョージア州技術局（GTA）は、ジョージア州にあるデジタルサービス機関です。アクイアの支援により、ジョージア州が使用していた旧式のCMSを移行、再構築、再設計し、単一のシステムにする戦略を策定しました。

彼らは、Acquia Cloud Platform上ですべてのサイトの一貫性と柔軟性を高めるマルチサイト・アーキテクチャを構築。その結果、コンテンツを効率的に移行しながら、レスポンシブでモバイルファーストの検索アプリケーションを開発することができました。

たった12ヶ月間で、Georgia.govは55個のサイトをAcquia Platform上で立ち上げました。

関係者は、DrupalとAcquia Platformへの移行により、5年間で470万ドルの節約になると見積もっています。この移行により、Georgia.govは少なくとも20台のサーバー管理から解放され、すべてのWebプロパティをより効率的に管理するための手法が確立されたのです。

サイト開設以来、ハイブリッドCMSのアプローチを活用し、政府機関を「ブラウザの向こう側」に導くためにGeorgia.govはアクイアと提携を続けています。

3ヶ月のプロジェクトの間で、ジョージア州のフードスタンプ（無料の食料クーポン）に関する情報や、州外の免許証の譲渡、選挙の期日前投票の取得、釣り免許の登録といった簡単な問い合わせなど、Amazon Echoデバイスを持っていれば誰でも利用できるAlexaの機能を構築しました。

この質問内容は、APIを介してジョージア州のWebサイトから提供されます。この情報は一箇所で更新することができ、またその情報を使用するすべての場所で同様に即座にアップデートされます。

まとめとポイント

プロジェクトごとに最適なアプローチが選択できるDRUPAL

ブランドは、ユーザーがどんな手段でコンテンツに接するかを問わず、効率的で最適化された、価値のあるデジタル体験を提供することが求められます。しかしそのためには、マーケティング担当者と開発者の双方にとって有効なCMSの活用が重要なのです。

バックエンドをプレゼンテーション層から切り離したヘッドレスDrupalアーキテクチャを使用することで、Web開発チームが必要とするコントロールと柔軟性を確実に得られます。これにより複数のチャネルやデバイスのオーディエンスに対して、クリエイティブなサービスを届けられるようになるのです。

しかし、多くの企業が感じているように、このアプローチによって技術的な問題が増加したり、マーケティング担当者が思い通りにコンテンツやキャンペーンを実施できなくなったりする場合、ヘッドレスCMSの導入は理想的とは言えないかもしれません。

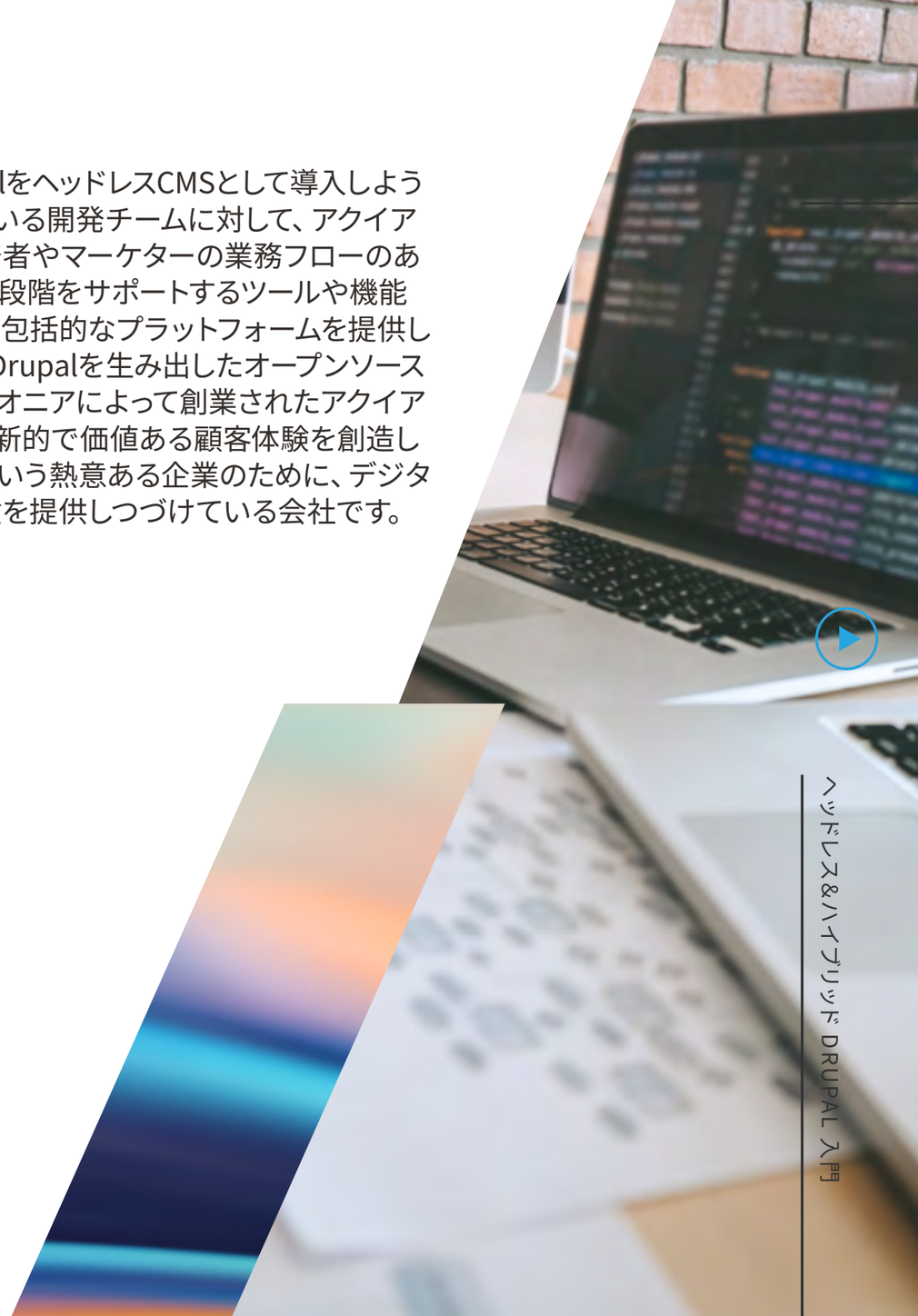
しかし、最近ではこれらの方法とは別のアプローチも増えてきています。

ハイブリッド型のヘッドレスDrupalは、統合されたシステムの利点とヘッドレスCMSの利点を兼ね備えています。このバランスをうまくとることで、コンテンツとデータの構築、管理、最適化が容易になり、さまざまなデバイスで優れたデジタル体験を提供できるようになります。最初はバランスが悪くても、ハイブリッドCMSを使えば、最適化されるまで調整することができるのです。

ハイブリッド型のヘッドレスDrupalを採用することで、Web開発チームは以下のことが可能になります。

- **協調性を持ちつつも個々に仕事をする**
- **コンテンツ管理の合理化**
- **制作スケジュールの短縮**
- **増え続けるインターフェースを通じて大切な情報を提供**
- **今後のテクノロジーへの迅速な対応**
- **デジタルエクスペリエンスの最適化**

DrupalをヘッドレスCMSとして導入しようとしている開発チームに対して、アクイアは開発者やマーケターの業務フローのあらゆる段階をサポートするツールや機能を持つ包括的なプラットフォームを提供します。Drupalを生み出したオープンソースのパイオニアによって創業されたアクイアは、革新的で価値ある顧客体験を創造したいという熱意ある企業のために、デジタル体験を提供しつづけている会社です。



ヘッドレスやハイブリッド型DRUPAL CMSがいかにして柔軟なチームワークを実現させるのかご興味がありますか？

デモをリクエストする

ACQUIA

ACQUIA.COM

アクイアについて

アクイアは、世界中で最も革新的なブランドがデジタルカスタマーエクスペリエンスを創造できるよう支援しています。オープンソースのDrupalを中核とするAcquia Digital Experience Platform（DXP）は、世界中のマーケター、開発者、IT運用チームなどが、顧客満足度やコンバージョン率を高め、競合他社に差をつけるデジタル製品やサービスの迅速な構築・展開を可能にします。