これで最後のレッスンでは、BMI計算機の入力ページの基礎となるもののスケルトンを作成しました。

そして、再利用可能なカードを作成し、別のステートレスウィジェットとして抽出すると、「final」という単語を省略した場合にプロパティを作成すると、再利用可能なカードクラスまたはクラスステートレスウィジェットである継承元は不変としてマークされます。

しかし、そのインスタンスフィールドの1つは最終的なものではなく、明らかに最終的なキーワードを持たないカラーフィールドを指します。

今、プログラミングには多くの言葉が投げかけられており、非常によく似たものを意味しますが、人々は異なる用語を選択するだけです。

プログラミングを初めて学習する場合、実際には、それぞれの用語の意味を理解できるようにするために、ほとんど辞書や用語集を用意しておくのが理にかなっています。また、前の警告で、インスタンスフィールドと呼ばれるものについて述べました。

そして、人々がインスタンス変数、インスタンスフィールド、またはプロパティについて話すとき、彼らは実際に同じことについて話していることに注意することが重要です。そして、それが参照するのは、クラス内で変更できるプロパティのみを作成するか、クラスを構築するときに別の値を設定することができます。

この警告で見たもう1つの用語は不変性です。

不変性とはどういう意味ですか？

可変性とは、変更可能であることを意味します。したがって、不変とは変更できないことを意味し、Flutterの開発に関しては変更できないことは、ほとんどすべてのステートレスウィジェットです。ウィジェットは一種のレゴブロックのようなもので、各ブロック自体は変更できません。

あなたは本当にレゴのピースを半分に壊したり、見ようとすることはできません。

あなたはおそらく半分でそれを見ることができましたが、それは実際にはゲームの精神ではありませんか？

そのため、各レゴブロックは不変ですが、変更することはできません。

それでは、不変または不変のブロックを使用して構築されたアプリにどのように変更を加えるのでしょうか？

さて、変更が必要なこれらの不変ウィジェットの1つを取得して破棄し、代わりに、必要な変更を含む新しいウィジェットを作成できます。

つまり、画面上で特定のウィジェットを更新すると、不変で変更できない古いウィジェットが発生します。そのため、実際には破壊され、更新された場所に新しいものが作成されます。

そのため、たとえばこの場合、ウィジェットの背景は白でした。

そして、そのウィジェットが青色の背景を持つように更新する必要がある場合、そのウィジェットは破棄され、青色の背景を持つ新しいウィジェットがその場所に作成されます。

これが不変性の仕組みです。

不変性について言えば、ステートレスウィジェットは不変です。変更することはできません。ステートレスなので、ステートは変わりません。

そのため、すべてのプロパティも一度しか設定できず、その後は石で設定されます。更新する場合は、新しいカードを作成して、新しい再利用可能なカードの新しい色を渡す必要があります。

そのため、プロパティの前にそのキーワードfinalを追加して、これを一度だけ設定してから変更できないようにする必要があります。

再利用可能なカードの色の最終的な値です。

ここで不変について説明しますが、実際には両方とも不変のプロパティの2つの異なるタイプがあります。また、Flutterプロジェクト全体でこれらの2つの異なるキーワードがよく使用されます。

そして、彼らが実際に行うことの間で混乱することは非常に簡単です。

それで、このレッスンを最後のキーワードとconstキーワードの違いは何であるかについて素早く話したいと思いました。

そして、何かをfinalまたはconstとしてマークすると、それが実際に意味するものになります。

ラウンド1、戦います。それを行うには、DartPadに戻り、constとfinalを作成しましょう。

したがって、int myNumber = 2のようなものを記述することで、新しい変数を作成できることがわかります。

この場合、最初のキーワードは、変数に含めることができるデータ型をマークします。

2番目は変数の名前です。

そして、等号の後に変数に含まれる値があります。

これは何度も見ました。しかし、この変数をもう変数にしたくない場合、変更できないようにしたい場合は、定数を使用するか最終を使用するかを選択できます。

いずれにせよ、これらのキーワードを最初に追加します。

それでは、constを作成し、これをmyConstに名前変更しましょう。そして、最後のintを作成し、このmyFinalに名前を付けて、3に等しくします。

これで、後日、myConstに別の値を保持することを決定したので、myConstが4に等しくなるようにしたかったということを示すことができる以外は、右側は実際には重要ではありません。定数は不変であるため、エラーが発生します。そして、ここで、定数変数に値を再び割り当てることはできません。そして、それはすでに開始値を持っているからであり、それは決して変わらないでしょう。

常にその値を保持します。

この特定のmyConstは2つだけであり、作成後は変更できません。

現在、同じことがファイナルにも当てはまります。

したがって、myFinalを6つに変更したい場合、エラーも発生します。

Finalはfinal変数であり、一度しか設定できません。どちらの場合も、不変です。

しかし、一方で、変数が通常の変数である場合。

それで、再びmyNumberを引き出しましょう。myNumberを新しい値、たとえば4に設定したかったのです。

これが変数のしくみだからです。

それらを作成し、後日変更することができます。

それらはさまざまです。

これが変数と呼ばれる理由です。しかし、定数と最終変数については、値を変更できません。

そのため、通常、不変性部分は非常に簡単に把握できます。

それを作成するときに最初に値を割り当て、その後はその値にする必要があります。二度と変更することはできません。

しかし、難しいのは、いつconstを使用し、いつfinalを使用するかを知ることです。

一見したところ、彼らはほとんど同じことをしているようだからですよね？

さて、この2つには微妙な違いがいくつかあります。それがこのレッスンで検討する内容です。したがって、Dart言語ツアーを読むと、finalとconstについて説明します。また、const変数はコンパイル時の定数であるのに対して、最終変数は1回しか設定できないと言われます。

これはどういう意味ですか？

実行ボタンをクリックすると、ここでDart言語で記述したコードが、マシンが理解できる形式にコンパイルされるため、1と0に近いものになります。そして、その瞬間に、定数内に保持されるべき値を解決できなければなりません。

しかし、最終的には、そうである必要はありません。

いつでも解決できます。

これを実演してみましょう。たとえば、定数は2 + 5 \* 8です。

これはconstとして完全に有効です。実行をクリックした瞬間に計算でき、その右側の値がconstに設定されるためです。実行時またはアプリが実際に電話で実行されているときに、その時点でしか使用できないものを使用したい場合は、画面上にあるボタンを言って、取得してみてくださいその次元では、constでそれを行うことができず、finalを使用する必要があります。そのため、constは実行時に何にもアクセスできません。

アプリが実行されているとき。コードのコンパイル後に作成されたものがある場合は、constに設定しないでください。

たとえば、現在の時刻を計算する場合、dateTimeから新しいオブジェクトを作成する必要はなく、dateTimeオブジェクトを使用して、今呼び出されたものを利用できます。これにより、現在の日付と時刻がわかります。ただし、定数に割り当てることはできません。これは、コードの実行後に解決する必要があるためです。

そして、dateTimeにアクセスできます。しかし、代わりにこれを自分のファイナルに移動した場合、整数ではないという事実を除いて、これは完全に有効なコードであることがわかります。

ここでmyFinalを印刷すると、コードがコンパイルされた後に日付と時刻が計算され、その時刻が表示されていることがわかります。

コードに戻ると、色をファイナルに変更できることがわかりますが、色は新しい再利用可能なカードを作成するときに由来し、いつでも作成できるため、constにはなりませんコードがコンパイルされたときだけでなく、アプリの実行中であればいつでもかまいません。そして、色の値を取得するのはその瞬間であるため、ここで実際にconstを使用することはできません。

最終的なものでなければなりません。

現在、同様に、再利用可能なカードは不変であるため、単純な変数にすることはできません。

したがって、新しい再利用可能なカードを作成するたびに、不変のステートレスウィジェットが作成されます。

そして、再利用可能なカードの色やサイズを変える必要があるときはいつでも、その再利用可能なカード、その特定のカードは実際に破壊されます。そして、新しいものが代わりになります。そのため、そのプロパティは変更できません。

変更できません。

これが最終として宣言する必要がある理由です。現在、通常、コードを記述しているとき、デザイン内の数値とハードコードされたものは通常、定数で表されます。

したがって、単に高さが固定された単なるコンテナであるデザインの下部を作成する場合、列の一番下でそれを行うことができるとしましょう。

したがって、おそらくここで列が終了する直前に、0xFFEB1555の色を持つ新しいコンテナを追加できます。これは、ドリブル側のデザインから得た素敵なピンク色です。

そして、マージンを追加します。

この場合、実際にはこのコンテナの上部にマージンが必要です。そして、コンテナの他のすべての部分、側面と底面、実際には余白がなく、すべての側面にくっつくようにします。

そのため、EdgeInsets.allと言う代わりに、EdgeInsets.onlyと言います。

設定するのは、10に設定する上部です。そして、コンテナの幅を設定します。

そして、私の場合は、どのスクリーンがオンであっても、このコンテナをスクリーンの幅全体に広げたいと思います。

そのため、小さな画面でもiPhone 4でも大きなiPadでも構いません。

伸びてほしい。

したがって、これを行う最も簡単な方法は、double.infinityを記述することです。

これにより、画面の全幅に等しい値が得られます。

したがって、コンテナに80ピクセルの固定高さを指定するだけです。

そして、保存をクリックすると、ここでそれを確認する必要があります。これは、見た目がとてもきれいです。ここで、高さが80ピクセルに固定されている場所では、後の段階で高すぎると判断する可能性があります。

小さくしたいかもしれないし、背を高くしたいかもしれませんが、実際のコードを掘り下げてこれを変更することは非常に困難です。

非常に頻繁に、ファイルの最上部で定義されているこの場合に定数が使用されていることがわかります。

それをbottomContainerHeightと呼び、80.0に設定します。

そして今、コンテナ内で、高さをその定数bottomContainerHeightに変更できます。

そして今、コードを掘り下げて、それがどこにあるかを見つけ、それが存在するすべての場所を修正しようとする必要はありません。

これで、単に上部を見て定数を確認し、必要に応じて変更できます。

したがって、実行をクリックする時点で解決できるため、これは簡単に定数になります。

つまり、80 \* 2であっても、80ピクセルです。これは、実際にアプリを実行するときに解決できるすべてのものです。

ただし、定数を作成できないのは、アプリを現在実行している必要があるためです。

したがって、たとえば、色に基づいた定数を作成することを決定することはできません。

たとえば、色の文字列値を知りたい場合、colorStringという定数を作成し、通常はその色から文字列値を作成する色プロパティ.toStringの値と等しくなるように設定するとどうなりますか？

しかし、const変数は定数値で初期化する必要があるため、これは完全に違法です。それは後の段階で、または色が実際に値を受け取るときに解決する必要があるものではありません。

これは違法です。

今ではあなたの番です。

そして課題は、生の16進コードを使用して作成されたこれらすべての色があることです。そして、それらは多くの場所のウィジェットの中にあり、コードを乱雑にします。

それで、一番上でそれらを一緒に収集する定数を作成できますか？

また、再利用可能なカードの色を変更したい場合は、6箇所ではなく1箇所で行うだけです。

そのため、ビデオを一時停止して、このチャレンジを完了してみてください。

大丈夫。

したがって、これはbottomContainerHeightを作成するのと同じくらい簡単です。

ファイルの最上部で作成し、呼び出したいものを何でも呼び出しますが、activeCardColourと呼び、その色に等しくなるように設定します。ここにあります。

それをコピーして、ここに貼り付けます。そして今、その色を使用するたびに、代わりにactiveCardColorを使用できます。

そしてこれも、実行ボタンを押したときに解決できるものです。

これでアクティブなカードの色が得られたので、さらに進んで下のコンテナの色を上に移動することもできます。

それをカットして、別の定数コードbottomContainerColourを追加してみましょう。これは、ここでのピンク色に等しくなります。そして、ここでbottomContainerColourに置き換えることができます。

そのため、コードはずっときれいに見え、ハードコードされているものはすべて、画面の上部にある定数を指し、これは非常に簡単に見つけて簡単に変更できます。そして、ここで行ったすべての変更は、それらが使用されているすべての場所に反映されることを知っています。

したがって、これは定数の本当に一般的な使用法です。これからのレッスンでは、多くの場所で定数とfinalを使用します。

だから、あなたはたくさんの練習とたくさんの露出を持つでしょう。

したがって、すぐに完全に意味をなさない場合は、私たちがそれに戻ってくることを心配しないでください。今後のレッスンでそれを再訪します。ただし、次のレッスンでは、再利用可能なカードに子を追加し、コード内に含まれるカスタムコンテンツを使用してさらにカスタマイズします。

それで、そのすべてについて、次のレッスンでお会いしましょう。