Erstellen der App „Hello, world“ (XAML)

In diesem Lernprogramm erfahren Sie, wie Sie XAML und C# zum Erstellen einer einfachen „Hello, World“-App für die Universelle Windows-Plattform (UWP) unter Windows 10 verwenden. Mit nur einem Projekt in Microsoft Visual Studio können Sie eine App erstellen, die auf allen Geräten mit Windows 10 ausgeführt werden kann.

Hier erfahren Sie Folgendes:

* Erstellen Sie ein neues **Visual Studio 2015**-Projekt für **Windows 10** und die **UWP**.
* Schreiben Sie XAML zum Ändern der UI auf der Startseite.
* Führen Sie das Projekt auf dem lokalen Desktop und auf dem Smartphone-Emulator in Visual Studio aus.
* Verwenden Sie einen SpeechSynthesizer, um die App sprechen zu lassen, wenn Sie auf eine Schaltfläche klicken.

Vorbereitung

* [Was ist eine universelle Windows-App](https://docs.microsoft.com/de-de/windows/uwp/get-started/whats-a-uwp)?
* Zum Durcharbeiten dieses Lernprogramms benötigen Sie Windows 10 und Visual Studio 2015. [Vorbereiten](https://docs.microsoft.com/de-de/windows/uwp/get-started/get-set-up).
* Außerdem wird davon ausgegangen, dass Sie das Standardfensterlayout in Visual Studio verwenden. Wenn Sie das Standardlayout ändern, können Sie es im Menü **Fenster** mit dem Befehl **Fensterlayout zurücksetzen** wiederherstellen.

Wenn Sie sich lieber ein Video ansehen möchten...

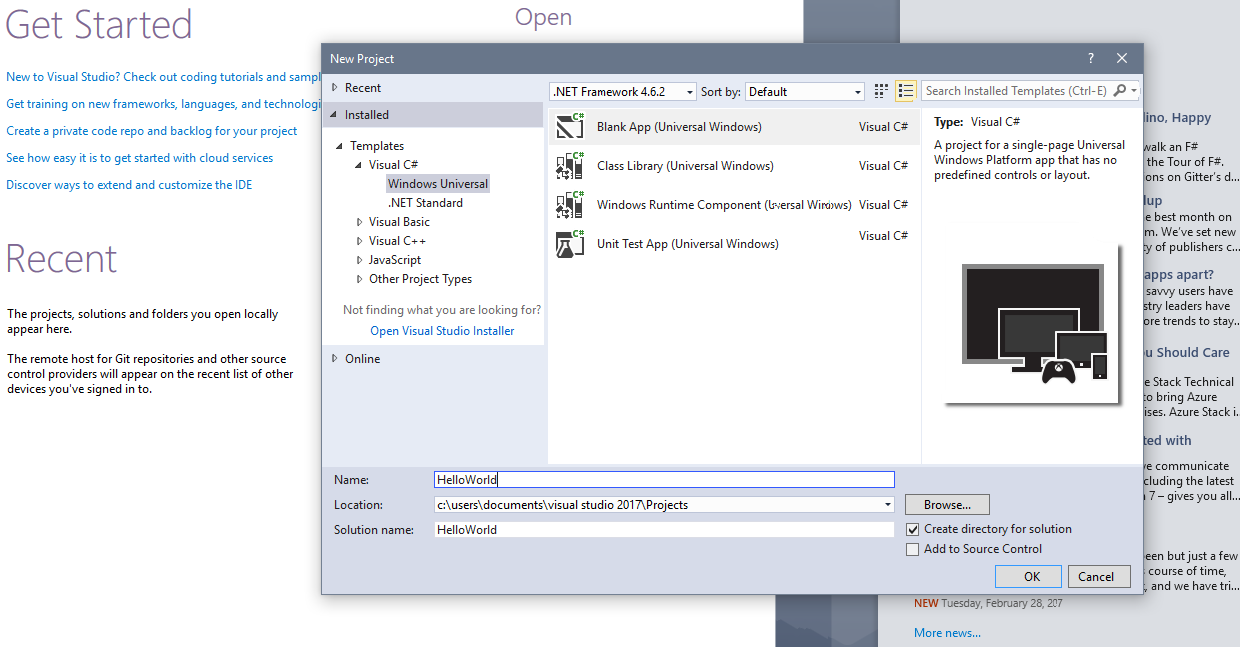
Falls Sie einen visuellen Ansatz statt eine Schritt-für-Schritt-Anleitung wünschen, wird in diesem Video dasselbe Material behandelt, jedoch mit einem schönen Soundtrack.

Schritt 1: Erstellen eines neuen Projekts in Visual Studio

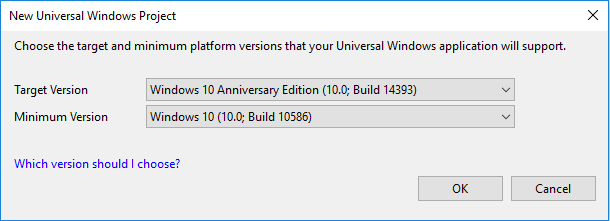
1. Starten Sie Visual Studio 2015.
2. Wählen Sie im Menü **Datei** die Befehle **Neu > Projekt...** aus, um das Dialogfeld *Neues Projekt* anzuzeigen.
3. Öffnen Sie in der Liste der Vorlagen auf der linken Seite **Installiert > Vorlagen > Visual C# > Windows**, und wählen Sie dann **Universell** aus, um eine Liste der UWP-Projektvorlagen anzuzeigen.

(Wenn keine universellen Vorlagen angezeigt wird, verfügen Sie möglicherweise nicht über Visual Studio 2015, oder die Komponenten zum Erstellen von UWP-Apps fehlen. Informationen zum Reparieren der Tools finden Sie unter [Vorbereiten](https://docs.microsoft.com/de-de/windows/uwp/get-started/get-set-up).)

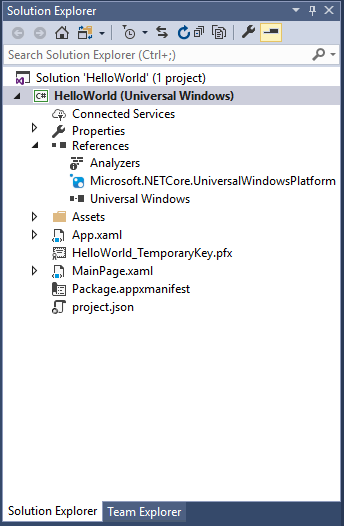
1. Wählen Sie die Vorlage **Leere App (universelle Windows-App)** aus, und geben Sie „HelloWorld“ als **Name** ein. Wählen Sie **OK** aus.



1. Das Dialogfeld für die Zielversion/mindestens erforderliche Version wird angezeigt. Die Standardeinstellungen sind in Ordnung, wählen Sie daher **OK** aus, um das Projekt zu erstellen.



1. Wenn das neue Projekt geöffnet wird, werden die Dateien im Bereich **Projektmappen-Explorer** auf der rechten Seite angezeigt. Möglicherweise müssen Sie die Registerkarte **Projektmappen-Explorer** anstelle der Registerkarte **Eigenschaften** auswählen, um die Dateien anzuzeigen.



**Leere App (universelle Windows-App)** ist zwar nur eine Minimalvorlage, umfasst aber trotzdem eine Reihe von Dateien. Diese Dateien werden für alle UWP-Apps mit C# benötigt. Sie sind Teil jedes Projekts, das Sie mit Visual Studio erstellen.

Inhalt der Dateien

Doppelklicken Sie zum Anzeigen und Bearbeiten einer Datei im Projekt im **Projektmappen-Explorer** auf die gewünschte Datei. Erweitern Sie eine XAML-Datei genau wie einen Ordner, um die zugeordnete Codedatei anzuzeigen. XAML-Dateien werden in einer geteilten Ansicht geöffnet, die sowohl die Entwurfsoberfläche als auch den XAML-Editor enthält.

Hinweis

Was ist XAML? Extensible Application Markup Language (XAML) ist die Sprache, die zum Definieren der Benutzeroberfläche Ihrer App verwendet wird. Sie kann manuell eingegeben oder mit den Visual Studio-Entwicklungstools erstellt wurden. Eine XAML-Datei verfügt über eine CodeBehind-Datei („.xaml.cs“), die die Logik enthält. Zusammen bilden XAML und CodeBehind eine vollständige Klasse. Weitere Informationen finden Sie in der [XAML-Übersicht](https://msdn.microsoft.com/library/windows/apps/Mt185595).

*„App.xaml“ und „App.xaml.cs“*

* In „App.xaml“ deklarieren Sie Ressourcen, die in der gesamten App zur Anwendung kommen.
* „App.xaml.cs“ ist die CodeBehind-Datei für „App.xaml“. Sie enthält wie alle CodeBehind-Seiten einen Konstruktor, der die InitializeComponent-Methode aufruft. Die InitializeComponent-Methode wird nicht von Ihnen geschrieben. Sie wird von Visual Studio generiert und dient in erster Linie dazu, die in der XAML-Datei deklarierten Elemente zu initialisieren.
* „App.xaml.cs“ ist der Einstiegspunkt für Ihre App.
* „App.xaml.cs“ enthält außerdem Methoden zum Behandeln der Aktivierung und Unterbrechung der App.

*MainPage.xaml*

* In „MainPage.xaml“ definieren Sie die Benutzeroberfläche für Ihre App. Sie können Elemente direkt per XAML-Markup hinzufügen oder die Designtools von Visual Studio verwenden.
* „MainPage.xaml.cs“ ist die CodeBehind-Seite für „MainPage.xaml“. Hier fügen Sie Ihre App-Logik und Ereignishandler hinzu.
* Zusammen definieren diese beiden Dateien im HelloWorld-Namespace eine neue Klasse mit dem Namen MainPage, die von [**Page**](https://msdn.microsoft.com/library/windows/apps/BR227503) erbt.

*Package.appxmanifest*

* Eine Manifestdatei, die Ihre App beschreibt (Name, Beschreibung, Kachel, Startseite usw.)
* Umfasst eine Liste der Dateien, die Ihre App enthält.

*Ein Satz mit Logobildern*

* „Assets/Square150x150Logo.scale-200.png“ stellt Ihre App im Startmenü dar.
* „Assets/StoreLogo.png“ stellt Ihre App im Windows Store dar.
* „Assets/SplashScreen.scale-200.png“ ist der Begrüßungsbildschirm, der beim Start der App angezeigt wird.

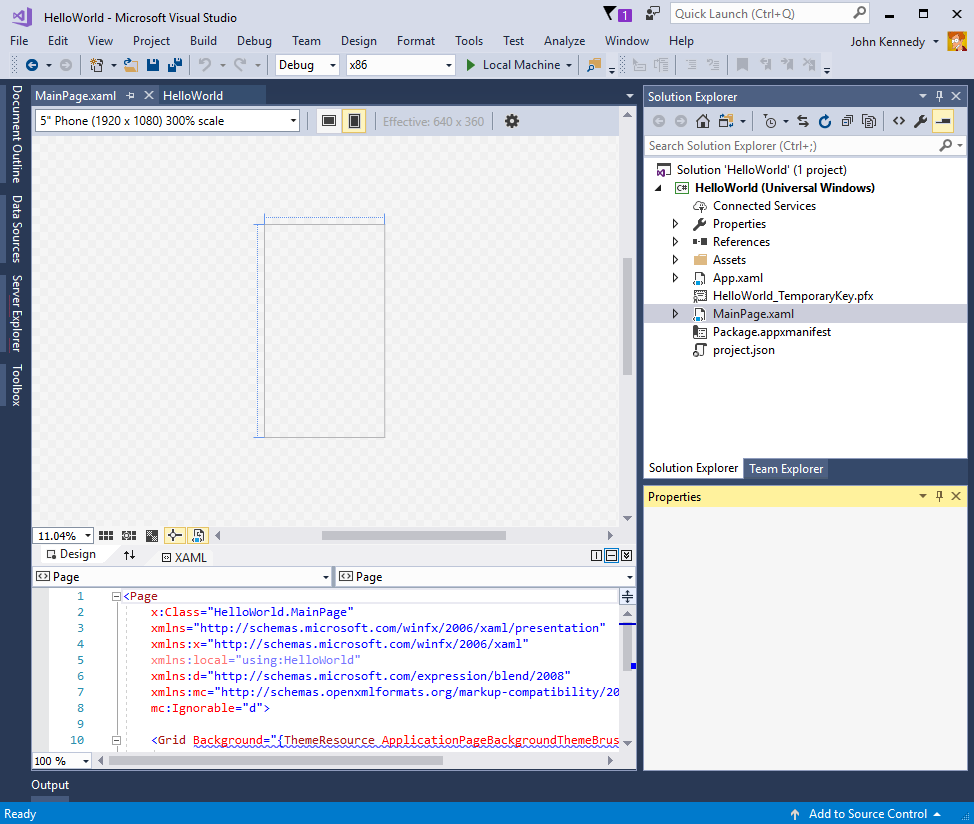
Schritt 2: Hinzufügen von Schaltflächen

Mithilfe der Entwurfsansicht

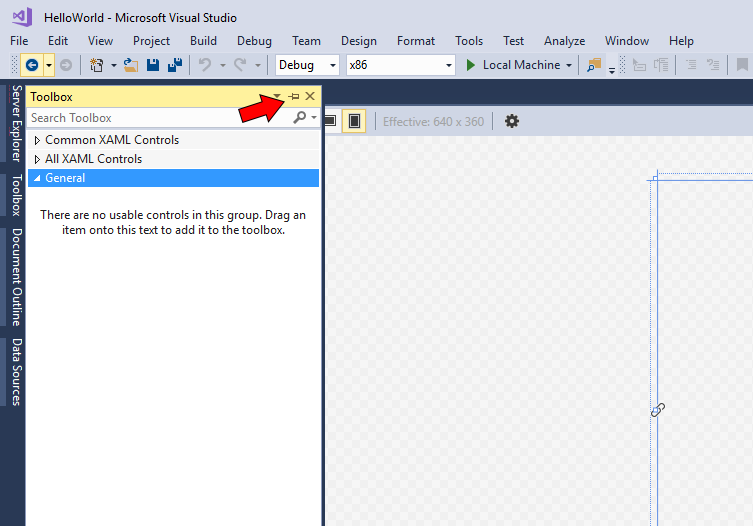
Fügen wir nun der Seite eine Schaltfläche hinzu. In diesem Lernprogramm verwenden Sie lediglich einige der zuvor aufgeführten Dateien: „App.xaml“, „MainPage.xaml“ und „MainPage.xaml.cs“.

1. Doppelklicken Sie auf die Datei **MainPage.xaml**, um sie in der Entwurfsansicht zu öffnen.

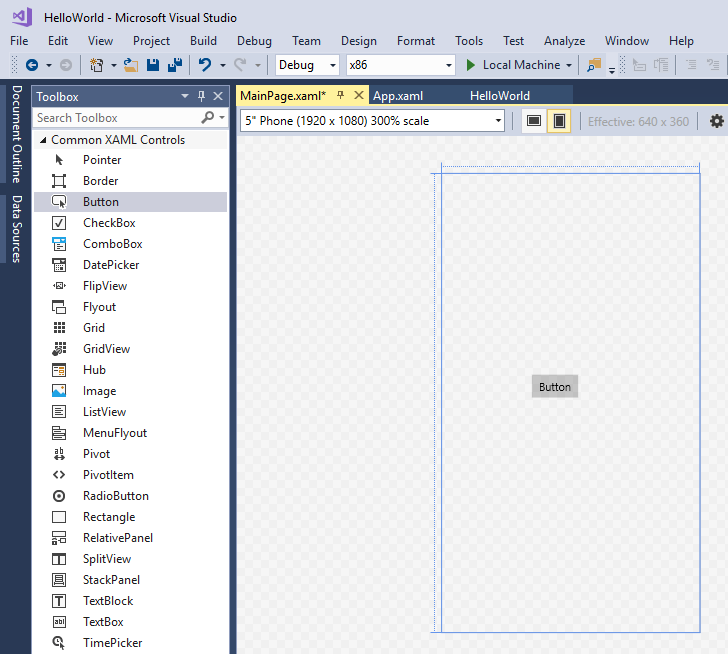
Sie werden feststellen, dass eine grafische Ansicht im oberen Teil des Bildschirms und die XAML-Codeansicht darunter vorhanden ist. Sie können jeweils Änderungen vornehmen, wir verwenden jetzt jedoch die grafische Ansicht.



1. Klicken Sie auf die vertikale Registerkarte **Toolbox** auf der linken Seite, um die Liste der UI-Steuerelemente zu öffnen. (Sie können auf das Reißzweckensymbol in der Titelleiste klicken, damit sie sichtbar bleibt.)



1. Erweitern Sie **Häufig verwendete XAML-Steuerelemente**, und ziehen Sie die **Schaltfläche** in die Mitte der Design-Canvas.



Wenn Sie das Fenster mit dem XAML-Code betrachten, sehen Sie, dass die Schaltfläche auch dort hinzugefügt wurde:

XAMLKopieren

<Button x:name="button" Content="Button" HorizontalAlignment="Left" Margin = "152,293,0,0" VerticalAlignment="Top"/>

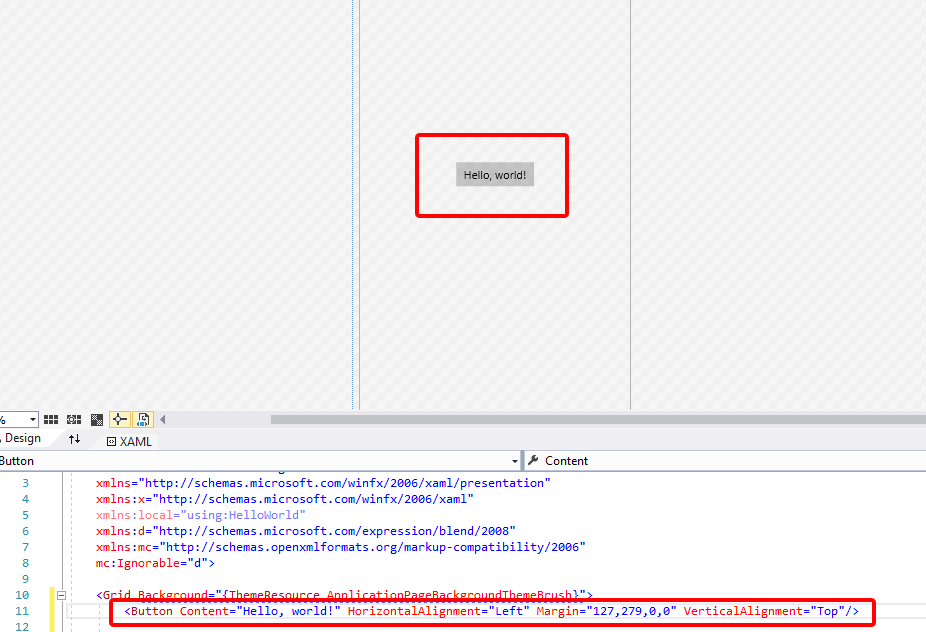
1. Ändern Sie den Text der Schaltfläche.

Klicken Sie in der XAML-Codeansicht, und ändern Sie den Inhalt von „Schaltfläche“ in „Hello, World!“.

XAMLKopieren

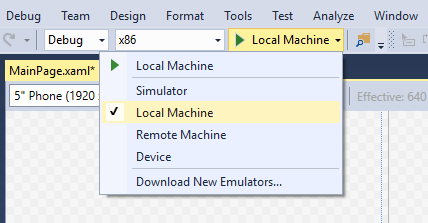
<Button x:name="button" Content="Hello, world!" HorizontalAlignment="Left" Margin = "152,293,0,0" VerticalAlignment="Top"/>

Beachten Sie, wie die in der Design-Canvas angezeigte Schaltfläche aktualisiert wird, um den neuen Text anzuzeigen.



Schritt 3: Starten der App

Sie haben nun eine sehr einfache App erstellt. Dies ist ein guter Zeitpunkt zum Erstellen, Bereitstellen und Starten Ihrer App, um sie in Aktion zu sehen. Sie können Ihre App auf dem lokalen Computer, in einem Simulator oder Emulator oder auf einem Remotegerät debuggen. Dies ist das Zielgerätmenü in Visual Studio.



Starten der App auf einem Desktop-Gerät

Standardmäßig wird die App auf dem lokalen Computer ausgeführt. Das Menü mit den Zielgeräten enthält mehrere Optionen zum Debuggen Ihrer App auf Geräten der Desktopfamilie.

* **Simulator**
* **Lokaler Computer**
* **Remotecomputer**

**So beginnen Sie mit dem Debuggen auf dem lokalen Computer**

1. Stellen Sie sicher, dass auf der **Standardsymbolleiste** im Menü mit den Zielgeräten (Menü „Debuggen starten“) die Option **Lokaler Computer** ausgewählt ist. (Dies ist die Standardeinstellung.)
2. Klicken Sie auf der Symbolleiste auf die Schaltfläche **Debuggen starten** (Schaltfläche „Debuggen starten“).

oder

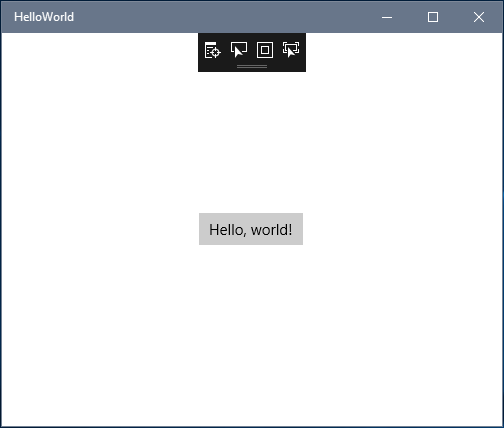
Klicken Sie im Menü **Debuggen** auf **Debuggen starten**.

oder

Drücken Sie F5.

Die App wird in einem Fenster geöffnet, und zuerst wird ein standardmäßiger Begrüßungsbildschirm angezeigt. Der Begrüßungsbildschirm setzt sich aus einem Bild (SplashScreen.png) und einer Hintergrundfarbe (in der Manifestdatei der App angegeben) zusammen.

Nach dem Ausblenden des Begrüßungsbildschirms wird Ihre App angezeigt. Sie sieht ungefähr wie folgt aus:



Drücken Sie die WINDOWS-TASTE, um das Menü **Start** zu öffnen, und zeigen Sie alle Apps an. Beachten Sie, dass beim lokalen Bereitstellen der App dem Menü **Start** die dazugehörige Kachel hinzugefügt wird. Wenn Sie die App später erneut ausführen möchten (nicht im Debugmodus), tippen oder klicken Sie im Menü **Start** auf die Kachel.

Viel zu bieten hat die App zwar noch nicht, aber trotzdem: Herzlichen Glückwunsch! Sie haben Ihre erste UWP-App erstellt!

**So beenden Sie das Debuggen**

Klicken Sie auf der Symbolleiste auf die Schaltfläche **Debuggen beenden** (Schaltfläche „Debuggen beenden“).

oder

Klicken Sie im Menü **Debuggen** auf **Debuggen beenden**.

oder

Schließen Sie das App-Fenster.

Starten der App in einem Emulator für mobile Geräte

Ihre App kann auf jedem Windows 10-Gerät ausgeführt werden. Wir sehen uns nun an, wie sie auf einem Windows Phone dargestellt wird.

Zusätzlich zu den Optionen zum Debuggen auf einem Desktopgerät enthält Visual Studio Optionen zum Bereitstellen und Debuggen Ihrer App auf einem physischen mobilen Gerät, das an den Computer angeschlossen ist, oder in einem Emulator für mobile Geräte. Sie können zwischen Emulatoren für Geräte mit unterschiedlichen Arbeitsspeicher- und Bildschirmkonfigurationen wählen.

* **Gerät**
* **Emulator WVGA 4 Zoll 512 MB**
* **Emulator WVGA 4 Zoll 1 GB**
* usw. (verschiedene Emulatoren mit anderen Konfigurationen)

(Werden die Emulatoren nicht angezeigt? Unter [Vorbereiten](https://docs.microsoft.com/de-de/windows/uwp/get-started/get-set-up) finden Sie Informationen dazu, wie Sie sicherstellen, dass die Entwicklungstools für universelle Windows-Apps installiert sind.)

**So beginnen Sie mit dem Debuggen in einem Emulator für mobile Geräte**

1. Es empfiehlt sich, Ihre App auf einem Gerät mit kleinem Bildschirm und begrenztem Arbeitsspeicher zu testen. Wählen Sie daher im Menü (Debuggen starten) des Zielgeräts in der Symbolleiste **Standard** die Option **Emulator 10.0.14393.0 WVGA 4 inch 512MB**.
2. Klicken Sie in der Symbolleiste auf die Schaltfläche **Debuggen starten** (Schaltfläche „Debuggen starten“).

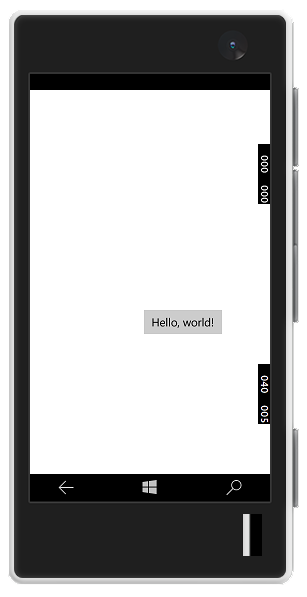
oder

Klicken Sie im Menü **Debuggen** auf **Debuggen starten**.

oder

Drücken Sie F5.

Visual Studio startet den ausgewählten Emulator, stellt dann die App bereit und startet sie. Beim ersten Start dauert dies möglicherweise etwas Zeit. Im Emulator für mobile Geräte sieht die App wie folgt aus.



Wenn Sie über ein Windows Phone mit Windows 10 verfügen, können Sie es auch mit dem Computer verbinden, die App bereitstellen und direkt darauf ausführen (zuerst müssen Sie jedoch den [Entwicklermodus aktivieren](https://docs.microsoft.com/de-de/windows/uwp/get-started/enable-your-device-for-development)).

Schritt 3: Ereignishandler

„Ereignishandler“ klingt kompliziert, dies ist jedoch nur ein anderer Namen für den Code, der aufgerufen wird, wenn ein Ereignis auftritt (z. B. wenn der Benutzer auf die Schaltfläche klickt).

1. Beenden Sie die Ausführung der App, sofern nicht bereits geschehen.
2. Doppelklicken Sie auf das Schaltflächen-Steuerelement auf die Design-Canvas, damit Visual Studio einen Ereignishandler für die Schaltfläche erstellt.

Natürlich können Sie den gesamten Code auch manuell erstellen. Oder Sie klicken zum Auswählen auf die Schaltfläche und suchen im Bereich **Eigenschaften** unten rechts. Wenn Sie zu **Ereignisse** (kleiner Gewitterblitz) wechseln, können Sie den Namen des Ereignishandlers hinzufügen.

1. Bearbeiten Sie den Ereignishandlercode in *MainPage.xaml.cs*, der CodeBehind-Seite. An dieser Stelle wird die Sache interessant. Der Standard-Ereignishandler sieht wie folgt aus:

C#Kopieren

private void button\_Click(object sender, RouteEventArgs e)

{

}

Wir ändern ihn, damit er wie folgt aussieht:

C#Kopieren

private async void button\_Click(object sender, RoutedEventArgs e)

{

MediaElement mediaElement = new MediaElement();

var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();

Windows.Media.SpeechSynthesis.SpeechSynthesisStream stream = await synth.SynthesizeTextToStreamAsync("Hello, World!");

mediaElement.SetSource(stream, stream.ContentType);

mediaElement.Play();

}

Geben Sie unbedingt auch das **async**-Schlüsselwort an, oder Sie erhalten beim Versuch, die App auszuführen, einen Fehler.

Was haben wir gerade gemacht?

Dieser Code verwendet Windows-APIs zum Erstellen eines Sprachsyntheseobjekts und gibt dann zu sprechenden Text an. (Weitere Informationen zur Verwendung von SpeechSynthesis finden Sie in den Dokumenten zum [SpeechSynthesis-Namespace](https://msdn.microsoft.com/library/windows/apps/windows.media.speechsynthesis.aspx).)

Wenn Sie die App ausführen und auf die Schaltfläche klicken, sagt Ihr Computer (oder das Handy) wörtlich „Hello, World!“.