Blazor

Module 9: Progressive Web Applications

Student Lab Manual

Instructor Edition (Book Title Hidden Style)

**Conditions and Terms of Use**

**Microsoft Confidential**

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2020 Microsoft Corporation. All rights reserved.

**Copyright and Trademarks**

© 2020 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at  
<http://www.microsoft.com/en-us/legal/intellectualproperty/Permissions/default.aspx>

Internet Explorer, Microsoft, SQL Server, Visual Studio, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Contents

[Lab 9: Progressive Web Applications 1](#_Toc38805748)

[Exercise 1: Add PWA Support 3](#_Toc38805749)

# Lab 9: Progressive Web Applications

#### Introduction

The aim of this exercise is to explore implementing an application using the new application model introduced under ASP.Net Core 3 called Blazor.

Important

**Blazor WebAssembly in preview**

Blazor Server is supported in ASP.NET Core 3.0. Blazor WebAssembly is in preview for ASP.NET Core 3.1.

#### Objectives

After completing this lab, you will be able to:

* Execute JavaScript code from within Blazor Components

#### Prerequisites

None

#### Scenario

In this scenario, we will explore implementing an online Pizza Delivery application.

#### System Requirements

* Follow the instructions found [on this page](https://docs.microsoft.com/en-us/aspnet/core/blazor/get-started?view=aspnetcore-3.1&tabs=visual-studio) to get started

#### Estimated Time to Complete This Lab

30 minutes

Exercise 1: Add PWA Support

#### Objectives

In this exercise, you will:

* Add support for PWA features

#### Scenario

Users will be able to install their application on their desktop and mobile device home screen.

Task 1: Adding a Service Worker

1. As a prerequisite to most of the PWA-type APIs, your application will need a *service worker*. This is a JavaScript file, usually quite small, that provides event handlers that the browser can invoke outside the context of your running application, for example when fetching resources from your domain, or when a push notification arrives. You can learn more about service workers in Google's [Web Fundamentals guide](https://developers.google.com/web/fundamentals/primers/service-workers).
2. Even though Blazor applications are built in .NET, your service worker will still be JavaScript because it runs outside the context of your application. Technically it would be possible to create a service worker that starts up the Mono WebAssembly runtime and then runs .NET code within the service worker context, but this is a lot of work that may be unnecessary considering that you might only need a few lines of JavaScript code. To add a service worker, create a file called service-worker.js in your client app's wwwroot directory, containing:

self.addEventListener('install', async event => {

console.log('Installing service worker...');

self.skipWaiting();

});

self.addEventListener('fetch', event => {

// You can add custom logic here for controlling whether to use cached data if offline, etc.

// The following line opts out, so requests go directly to the network as usual.

return null;  
});

This service worker doesn't really do anything yet. It just installs itself, and then whenever any fetch event occurs (meaning that the browser is performing an HTTP request to your origin), it simply opts out of processing the request so that the browser handles it normally. If you want, you can come back to this file later and add some more advanced functionality like offline support, but we don't need that just yet.

Enable the service worker by adding the following <script> element into your index.html file, for example beneath the other <script> elements:

<script>navigator.serviceWorker.register('service-worker.js');</script>

If you run your app now, then in the browser's dev tools console, you should see it log the following message:

Installing service worker...

Note that this only happens during the first page load after each time you modify service-worker.js. It doesn't re-install on each load if that file's contents (compared byte-for-byte) haven't changed. Try it out: check that you can make some trivial change to the file (such as adding a comment or changing whitespace) and observe that it reinstalls after those changes, but not at other times.

This might not seem to achieve anything yet, but is a prerequisite for the following steps.

Task 2: Making Your App Installable

1. Next, let's make it possible to install Blazing Pizza into your OS. This uses a browser feature in Chrome/Edge-beta on Windows/Mac/Linux, or Safari/Chrome for iOS/Android. It may not yet be implemented on other browsers such as Firefox. First add a file called manifest.json in your client app's wwwroot, containing:

{

"short\_name": "Blazing Pizza",

"name": "Blazing Pizza",

"icons": [

{

"src": "img/icon-512.png",

"type": "image/png",

"sizes": "512x512"

}

],

"start\_url": "/",

"background\_color": "#860000",

"display": "standalone",

"scope": "/",

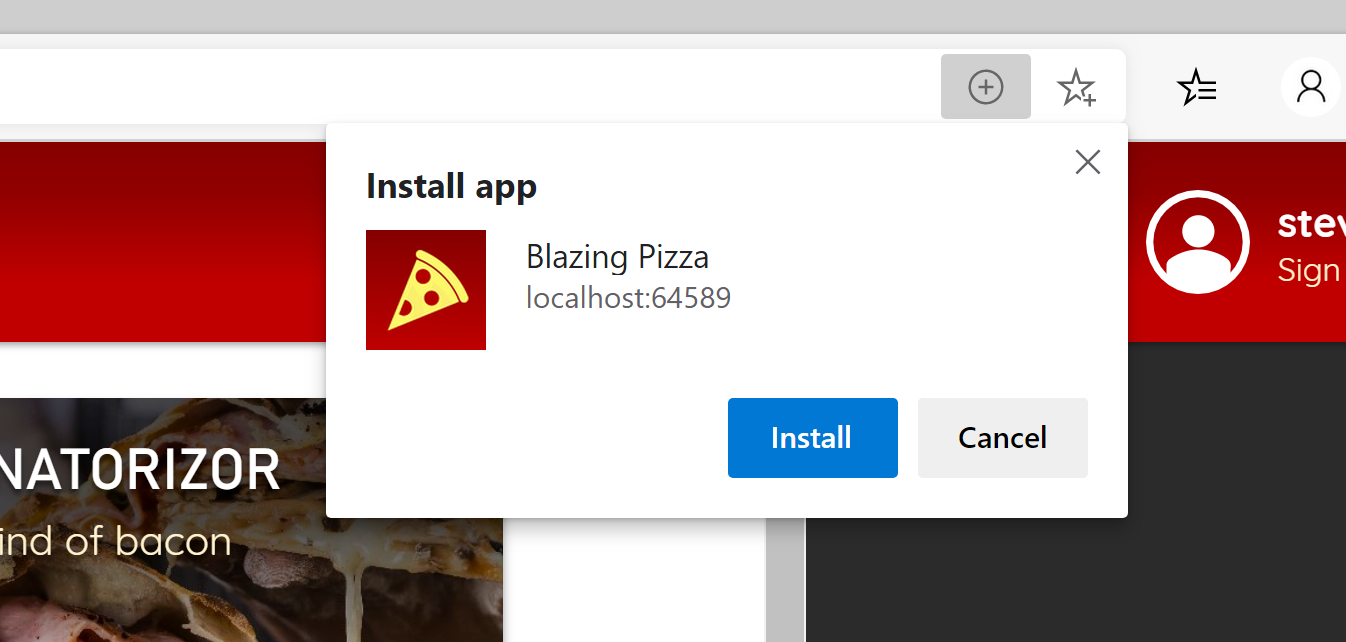
"theme\_color": "#860000"

}

1. You can probably guess what this information is used for. It will determine how your app will be presented to the user once installed into the OS. Feel free to change the text or colors if you want.
2. Next you'll need to tell the browser where to find this file. Add the following element into the <head> section of your index.html:

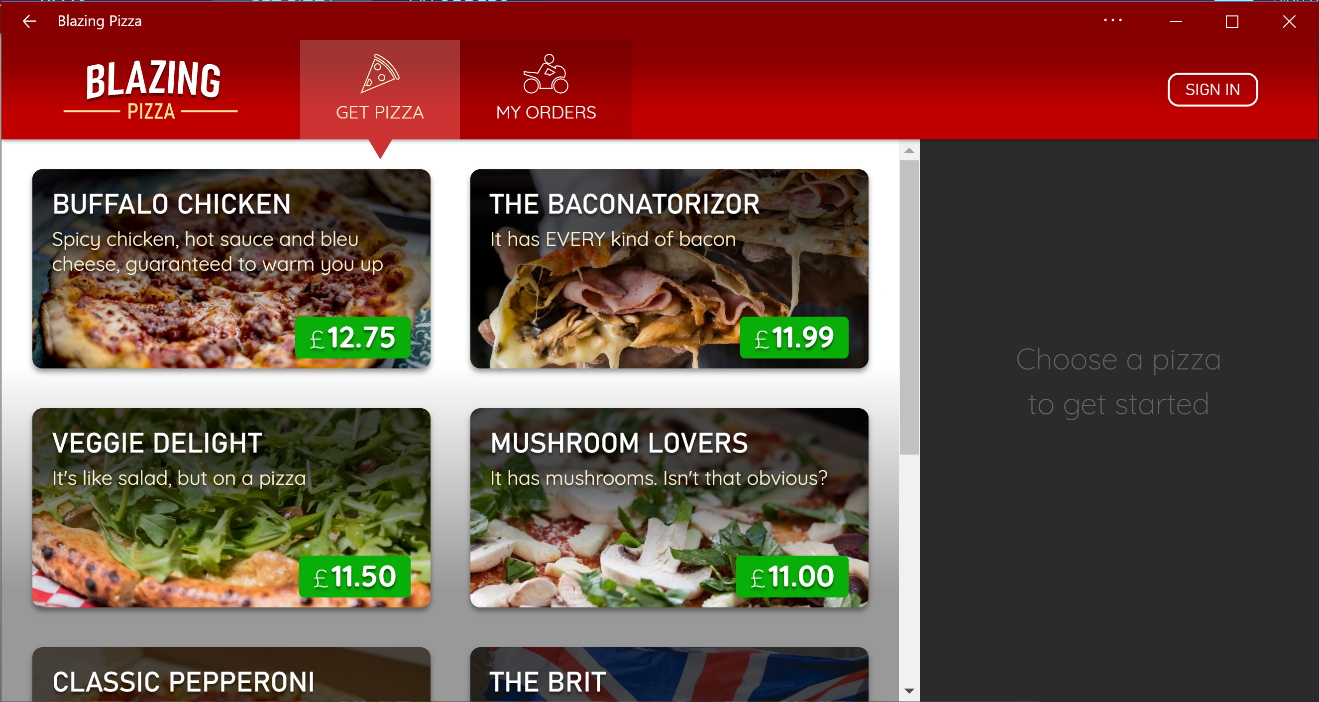
<link rel="manifest" href="manifest.json" />

... and that's it! Next time you load the site in Chrome or Edge beta, you'll see a new icon in the address bar that prompts users to install the app:

[](https://user-images.githubusercontent.com/1101362/66352975-d1eee900-e958-11e9-9042-85ea4ac0c56b.png)

Users on mobile devices would reach the same functionality via an option called Add to home screen or similar.

Once installed, the app will appear as a standalone app in its own window with no other browser UI:

[](https://user-images.githubusercontent.com/1101362/66356174-0024f680-e962-11e9-9218-3f1ca657a7a7.png)

Users on Windows will also find it on their start menu, and can pin it to their taskbar if desired. Similar options exist on macOS.