The Qt Company

8 things to know about WebAssembly

The **quick guide for the basics** of Qt for WebAssembly

The Qt Company

Table of Contents

01.

What is Qt for WebAssembly?

WebAssembly (abbreviated Wasm) is a bytecode representation meant to be targeted by high-level programming languages such as C++ and to be executed in a virtual machine in a web browser. **Qt for WebAssembly lets you run Qt applications on the web.** With Qt for WebAssembly, **you can distribute your application as a web application that runs in a browser sandbox**. This approach is suitable for web-distributed applications that do not require full access to host device capabilities.

From Qt's perspective, WebAssembly is just another target platform. Starting with Qt 6.4, **it is an officially supported target platform** for selected, relevant modules. You can download the binary builds on Linux, macOS, and Windows host platforms and build your Qt applications to be run inside a web browser.

Qt for WebAssembly lets you run Qt applications on the web

02.

Is WebAssembly targeted only for the web?

The **Qt WebAssembly** offering is mainly geared towards the client, i.e., running in the browser. However, there is widespread usage of WebAssembly on the server side using technologies like wasi and wasmer.

WebAssembly can be used to implement unique use cases, which required much more work previously. With WebAssembly, anybody can take their Qt application to run in a web browser by recompiling it for WebAssembly as a target platform. It is also possible to reuse parts of a Qt application when building all new web applications combining JavaScript and WebAssembly components. **This will reduce the development effort significantly**. It also might improve usability because UI elements are used across different platforms.

Furthermore, applications using WebAssembly technology run in a web browser with zero installation. If Qt applications should be taken to mobile devices and the app store publishing is too cumbersome, then **WebAssembly is a good way to bypass app stores**. One should mention that WebAssembly comes with restrictions because it runs in a sandboxed environment in the web browser. WebAssembly does not have access by default to the low-level hardware capabilities of the target device. This has been done to increase the security of WebAssembly applications. **It also allows the applications to run on multiple platforms and many web browsers without modifications.**

Remote UIs for embedded devices are becoming more popular. **WebAssembly is a great way to implement remote UIs**. Reusing large parts of an existing User Interface implementation allows embedded system makers to create an additional remote display or even replace the in-built display altogether.

03.

For which use cases is Qt for WebAssembly useful?

Qt for WebAssembly has several key use cases:

1. Taking native apps to the web by recompiling Qt C++ code to WebAssembly – often requires working around platform limitations depending on the application and what lower-level functionality is needed. This approach allows you to have a web story by reusing your existing code and development resources.
2. Deploying apps without app stores – zero install, this could be especially interesting in e.g. embedded products.
3. Sharing e.g. embedded designs with stakeholders by sharing only an URL that allows running the design in a browser – please see DesignViewer in Qt Design Studio and https://qt-webassembly.io/designviewer/
4. Embedded devices with no or limited display, occasionally needing a rich UI – reuse your existing code, with a possibility for a companion app.
5. Remote control and monitoring of devices in the field - Again you have the opportunity to reuse existing code, running on standard tablets and desktop computers.

04.

What to build with WebAssembly?

WebAssembly, in general, has an extensive range of potential use cases. It also has superior performance to pure javascript, which is helpful in more complex web applications. From the Qt perspective, there are exciting use cases listed in this guide. Additionally, there are options for the architecture of the code as listed by webassembly.org:

- Entire codebase in WebAssembly
- Mainframe in WebAssembly, but the UI in JavaScript / HTML
- Reuse of existing code by targeting WebAssembly, embedded in a larger JavaScript / HTML application.

Some of the most recent **Qt for WebAssembly examples** include:

DesignViewer

QtQuick 3D

QML Editor

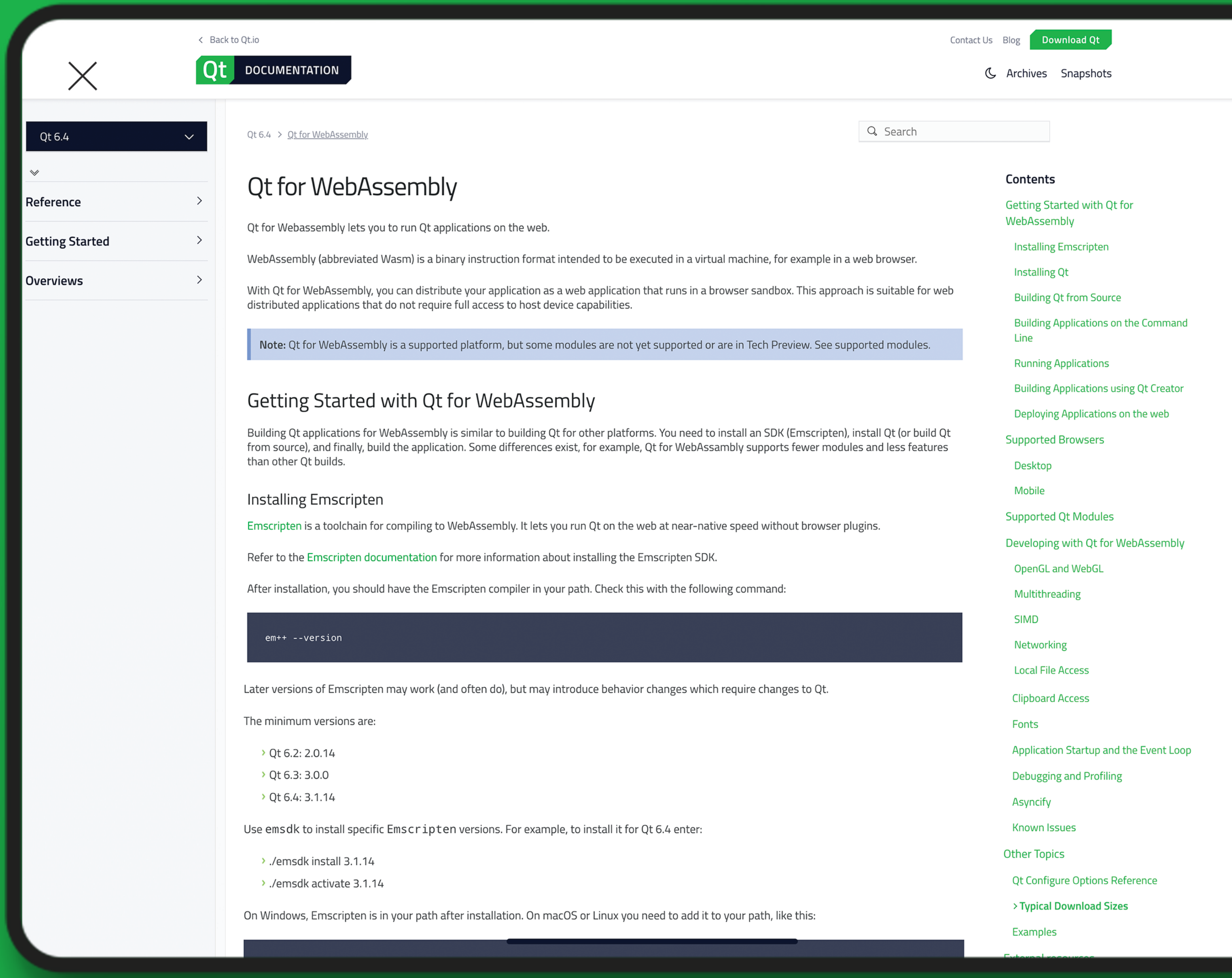
Slate

05.

How to build applications with Qt for WebAssembly?

Building Qt applications for WebAssembly is similar to building Qt for other platforms. You need to install an SDK (Emscripten), install Qt (or build Qt from source), and finally, build the application. Some differences exist. For example, Qt for WebAssembly supports fewer modules and features than other Qt builds.

Please see our technical page for more details on building:
https://doc.qt.io/qt-6/wasm.html


Back to Qt.io

Qt DOCUMENTATION

Contact Us Blog Download Qt

Archives Snapshots

Qt 6.4

Reference

Getting Started

Overviews

Qt 6.4 > Qt for WebAssembly

Search

Qt for WebAssembly

Qt for Webassembly lets you to run Qt applications on the web.

WebAssembly (abbreviated Wasm) is a binary instruction format intended to be executed in a virtual machine, for example in a web browser.

With Qt for WebAssembly, you can distribute your application as a web application that runs in a browser sandbox. This approach is suitable for web distributed applications that do not require full access to host device capabilities.

Note: Qt for WebAssembly is a supported platform, but some modules are not yet supported or are in Tech Preview. See supported modules.

Getting Started with Qt for WebAssembly

Building Qt applications for WebAssembly is similar to building Qt for other platforms. You need to install an SDK (Emscripten), install Qt (or build Qt from source), and finally, build the application. Some differences exist, for example, Qt for WebAssambly supports fewer modules and less features than other Qt builds.

Installing Emscripten

Emscripten is a toolchain for compiling to WebAssembly. It lets you run Qt on the web at near-native speed without browser plugins.

Refer to the Emscripten documentation for more information about installing the Emscripten SDK.

After installation, you should have the Emscripten compiler in your path. Check this with the following command:

```
em++ --version
```

Later versions of Emscripten may work (and often do), but may introduce behavior changes which require changes to Qt.

The minimum versions are:

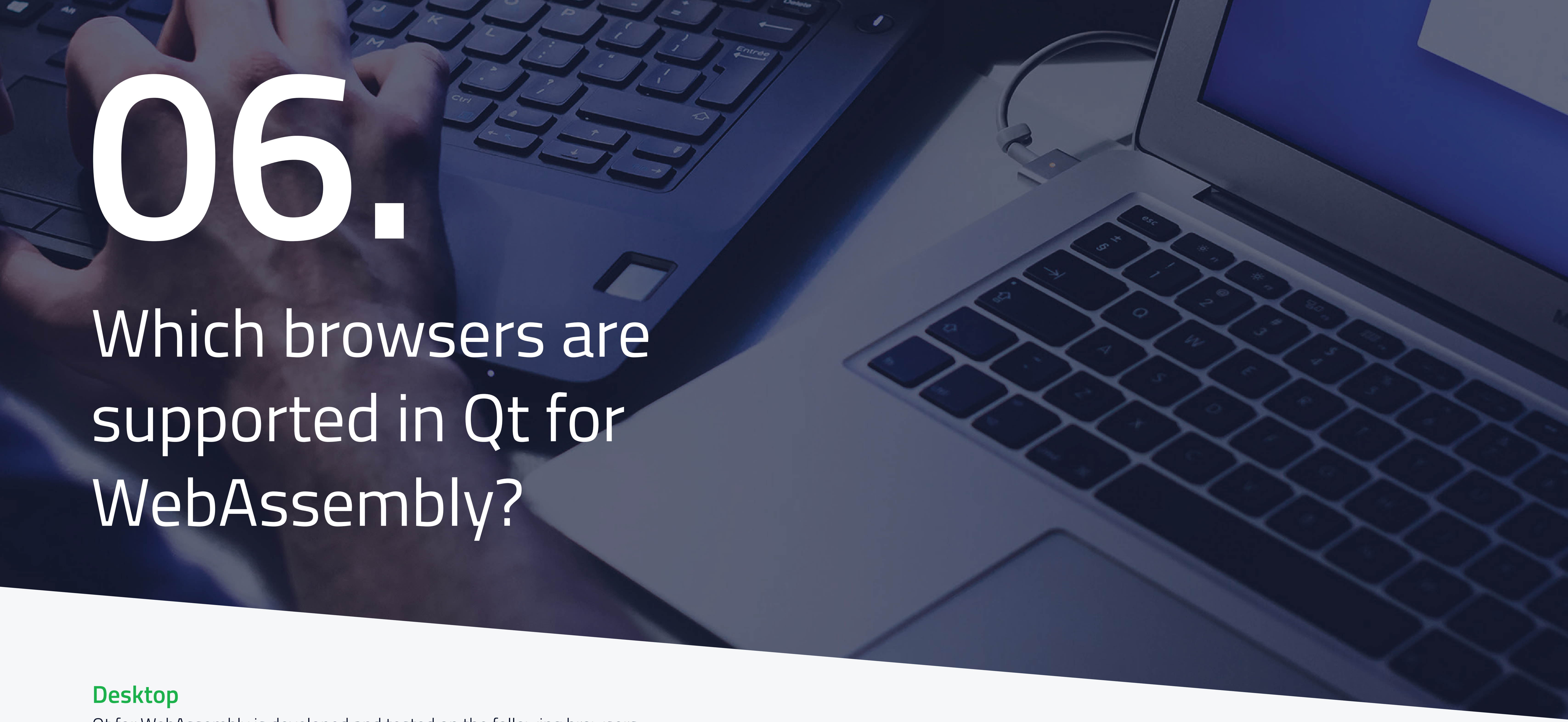
- Qt 6.2: 2.0.14
- Qt 6.3: 3.0.0
- Qt 6.4: 3.1.14

Use `emsdk` to install specific `Emscripten` versions. For example, to install it for Qt 6.4 enter:

- ./emsdk install 3.1.14
- ./emsdk activate 3.1.14

On Windows, Emscripten is in your path after installation. On macOS or Linux you need to add it to your path, like this:

Contents



06.

Which browsers are supported in Qt for WebAssembly?

Desktop

Qt for WebAssembly is developed and tested on the following browsers:

- Chrome
- Firefox
- Safari
- Edge

Qt should run if the browser supports WebAssembly. Qt has a fixed WebGL requirement, even if the application does not use hardware-accelerated graphics. Browsers that support WebAssembly often support WebGL, though some browsers blacklist older or unsupported GPUs. s/qtloader.js provides APIs to check if WebGL is available.

Qt does not make direct use of operating system features, and it makes no difference if, for example, FireFox runs on Windows or macOS. Qt does use some operating system adaptations, for example for ctrl/cmd key handling on macOS. WebAssembly support in browsers is evolving rapidly. We currently recommend using Chrome or Firefox for the best possible experience.

Mobile

Qt for WebAssembly applications runs on mobile browsers such as mobile Safari and Android Chrome.

07.

What limitations need to be considered when assessing the use of Qt for WebAssembly?

There are some limitations due to the nature of WebAssembly technology, utilizing the Javascript sandbox for security. The most notable things to take into account are:

- **The use of multithreading** – needs to be enabled separately and is still experimental in WebAssembly. Additionally, multithreading requires browser support for SharedArrayBuffer. See caniuse sharedarraybuffer for the current supported status.
- **SIMD performance enhancement** – needs to be enabled separately on a need basis.
- **Networking** – WebSockets are the basic mechanism, although TCP/UDP sockets can also be used with limitations. HTTP requests can also be used to the web page origin server or to a server that supports CORS. This includes XMLHttpRequest from QML.
- **Local file access** – The usual dialogs don't work on WASM. Instantiating and running a QFileDialog will display the virtual filesystem instead of the user's real filesystem.
- **Clipboard access** – some differences due to the web sandbox. In general, clipboard access requires user permission, which can be obtained by handling an input event (e.g., CTRL+c) or using the Clipboard API. Browsers that support the Clipboard API are preferred. Note that a requirement for this API is that the web page is served over a secure connection (e.g., https) and that some browsers may require changing configuration flags.
- **Debugging and profiling** - Wasm debugging is done on the browser javascript console. Debugging applications on Wasm directly within Qt Creator is not possible.
- **Fonts** – only a few fonts are supported by default, but the app can enhance this.

For a full list please see our documentation:
https://doc.qt.io/qt-6/wasm.html

Browsers may also not support all the latest features, and some APIs are currently browser-specific.

08. Where to start exploring Qt for WebAssembly?

Our **documentation page** could be a good starting point for Qt-related matters:
https://doc.qt.io/qt-6/wasm.html

Qt WebAssembly demos:
https://www.qt.io/qt-examples-for-webassembly

Generic information about WebAssembly:
https://webassembly.org/

About The Qt Company

Qt Group (Nasdaq Helsinki: QTCOM) is a global software company with a strong presence in more than 70 industries and is the leading independent technology behind millions of devices and applications. Qt is used by major global companies and developers worldwide, and the technology enables its customers to deliver exceptional user experiences and advance their digital transformation initiatives.

Copyright © 2022 The Qt Company. All Rights Reserved.

www.qt.io