Qt WebAssembly

- Kimmo Ollila

WebAssembly (Wasm)

- WebAssembly is a bytecode format intended to be executed in a web browser
  - Allows applications to be deployed to a device without any going through any explicit installation steps
  - Device needs to have a compliant web browser

- The applications will be running inside a secure sandbox in the web browser
  - Appropriate for apps that do not need full access to device capabilities but benefits from zero-installation process

Example use cases

- Shared HMI

- Car companion apps (targeted to passangers)
  - Benefits zero installation, temporary usage
  - Remote control for radio tuner, volume etc. from the rear seat
  - Diagnostics, vehicle information

WebAssembly and Qt

- Qt WebAssembly makes it possible to build Qt applications as WebAssembly modules
  - Target Qt applications to run on all major web browsers
  - Currently in development, a second tech preview is scheduled for release with Qt 5.12.0

WebAssembly and Qt

- Requires Emscripten
  - Toolchain for compiling to asm.js and WebAssembly
  - Built using LLVM compiler infrastructure
  - Known good version for Qt: 1.38.1

- Getting the code
  - The Qt sources can be downloaded from the Qt Account or checked out manually from git repositories

Supported Qt modules

- Qt modules for WebAssembly (5.11 TP)
  - QtBase (Qt Core, Qt Gui, Qt Graphics)
  - QtDeclarative (Qt Qml, Qt Quick)
  - Qt Graphical Effects
  - Qt Quick Controls
  - Qt Quick Controls 2
  - Qt Svg
  - Qt Web Sockets
  - Qt Charts

Building Qt for WebAssembly

- Get Emscripten
  - Supported host platforms: Linux, MacOS, Windows Subsystem for Linux
- Configure and build Qt
  - ./configure -xplatform wasm-emscripten -developer-build -release -static -no-feature-thread -nomake tests -nomake examples -no-dbus -no-headersclean -no-ssl -no-warnings-are-errors
  - make

Building and running Qt application

- Build like any other Qt application
  - /path/to/qmake && make
  - Generates .wasm, appname.js and appname.html

- Run your Qt application
  - Start a web server (python -m SimpleHTTPServer)
  - Open localhost:8000/appname.html in a web browser
  - Or use /path/to/emscripten/emrun –browser=firefox appname.html

Supported browsers

- Mainly developed and tested on desktop
  - Chrome, Firefox, Safari
  - Firefox (nightly) has the most performant wasm compiler
  - Some mobile testing is done in Android and iOS

Footprint (download size)

- Wasm modules can be large, but compress quite well
  - Compression is typically handled on the server side using standard compression features
    - Server compresses automatically
    - Provides pre-compressed version of the files
  - No need to have special handling of wasm files

Some known issues

- Disabled threading support in WebAssembly
- Nested event loops are not supported
  - No QDialog::exec() or new QEventLoop objects
- No access to system fonts
  - Apps need to distribute their fonts in .qrc for example
- Qt renders to canvas, not using any other DOM elements
  - Accecibility (screen readers) are not supported, text inputs won’t trigger VKBs
- All known issues
  - https://bugreports.qt.io/browse/QTBUG-63917

Demos

- QtWebAssembly demos available
  - https://msorvig.github.io/qt-webassembly-examples/


Qt