The Qt Company

WHITEPAPER

組み込み開発者のための ゼロインストールガイド

本ホワイトペーパーでは、ゼロインストールがいかに進化を遂げてきたかを説明するとともに、典型的なゼロインストールソリューションを構成するコンポーネントをご紹介します。また、ゼロインストールを実現するのに必要なテクノロジーを概説し、組み込み製品開発プロジェクトでゼロインストール機能を搭載する際に検討するべきポイントについても論じます。

目次

はじめに／用語 3
ゼロインストールの実用 4
ゼロインストールの進化 5
ユーザーが享受するメリット 5
メーカーが享受するメリット 6

ゼロインストールを避けるべきケース 7
非ネイティブUI 7
ユーザーインタラクションがない 7
TCP/IPがない 7
ストレージがない 7
接続性が悪い 7
デバイスアクセスが制限されている 7

最新ゼロインストールテクノロジーポートフォリオ 8
HTML5 8
WebGL 8
WebAssembly 8
WebSocket API（WebSockets） 8

Qtを使ったゼロインストールソリューション 8
WebGL 9
WebAssembly 9
WebGLとWebAssemblyの比較 11
WebGL 11
WebAssembly 11
仕組み 11
既存のQtアプリケーションの変換 11
アプリケーションの速度 11
UIの反応性 11
初回起動時間 11
同時クライアントの接続数 11
リバースエンジニアリングの難しさ 11
制約 11

Qtゼロインストールテクノロジースタック 12
ゼロインストールとクラウド 12
機能的安全性とサイバーセキュリティ 13
まとめ 13

はじめに／用語

ゼロインストールとは、組み込みデバイスのユーザーにブラウザベースのアプリケーションを提供することで、それらのアプリケーションをダウンロードまたはインストールしなくても実行できるようにする機能のことです。通常のインストールプロセスが不要になるため、面倒な設定もいらず、能率的にアプリケーションの運用を開始することができます。また、標準化されたブラウザテクノロジーを基盤とするため、1つのアプリケーションをWindows、Mac、またはLinuxのデスクトップ、AndroidまたはAppleのスマートフォン、各種タブレットなど、あらゆるクライアントデバイスでも実行することができます。

ゼロインストールとは、組み込みデバイスのユーザーが専用アプリケーションをインストールしなくても、スマートフォンやタブレット、デスクトップのウェブページからデバイスを制御できるようにする機能のことです。

ゼロインストールは、0Installのような固有のソフトウェアパッケージやディストリビューションツールの製品名ではなく、アプリケーションをインストールすることなくブラウザベースで実行する機能の総称です。ゼロインストールの定義は複数存在しますが、本ホワイトペーパーでは、「組み込みデバイスがホストするブラウザベースのコンパニオンプログラムを通してそのデバイスを制御する機能」と定義します。（なお本書の後半では、クラウドホストのアプリケーションで組み込みデバイスを制御するゼロインストールもご紹介します。）ただ、ゼロインストールをどのように定義するにしても、ゼロインストールソリューションは次のような複数のコンポーネントで構成されます。

組み込みデバイス：ゼロインストールソリューションの第一のコンポーネントは、スマートホームコントロールセンターや産業ロボット、X線機器などのハードウェアです。ユーザーは通信によって組み込みデバイスを制御し、デバイスのステータスを確認します。

通信機能：上記の組み込みデバイスが、WiFiやEthernet、モバイルデータ通信経由でTCP/IP通信機能を提供します。IoTデバイスでは一般的なインターネット通信が利用されますが、必ずしもインターネット通信を用いる必要はありません。インターネットを経由しない専用ネットワーク通信やダイレクトなネットワーク通信でも代替できます。これは特に高度なセキュリティの安全性や機能的安全性が求められるアプリケーションでは推奨されるアプローチです。

ユーザーデバイス（クライアントデバイス）：デスクトップ／ラップトップPCやスマートフォン、タブレットなどのクライアントデバイスで、リモートユーザーインターフェースを提供します。リモートアプリケーション用に十分な空きメモリを有し、近代的なブラウザを実行できる必要があります。

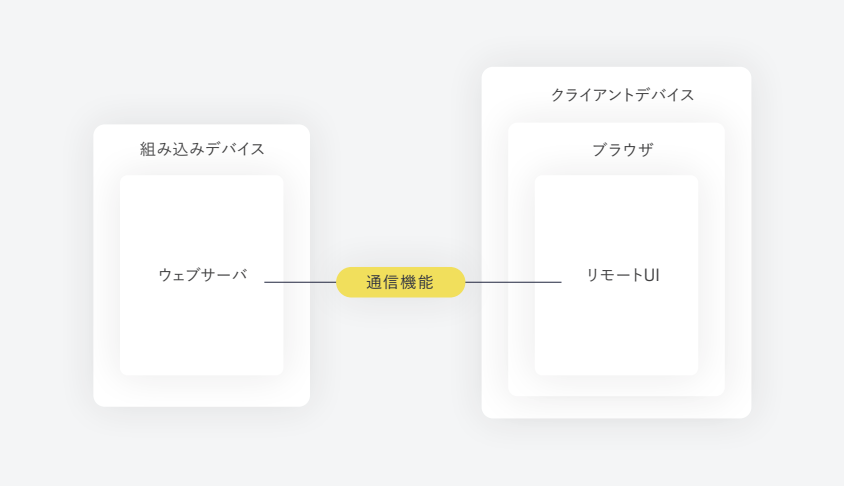
リモートUI（リモート／コンパニオンアプリ）：クライアントデバイスのブラウザで実行し、組み込みデバイスを制御するソフトウェアです。通常はデバイスのメインとなるUIで、物理的なディスプレイやボタンを備えている場合もあります。

ブラウザ：クライアントデバイス上でウェブブラウザを実行し、アプリケーションを実行する仮想環境を提供します。

ウェブサーバ：組み込みデバイスのウェブサーバで、アプリケーションをブラウザに提供します。

ゼロインストールの実用

ゼロインストールテクノロジーは、ユーザーにもメーカーにも数多くのメリットをもたらします。そのため、以下のようなさまざまな領域ですでに積極的に導入されています。



ゼロインストールソリューションの主なコンポーネント

産業オートメーション：複数の企業が、プロセス／オートメーション制御ソフトウェアにゼロインストールソリューションを活用しています。たとえばボッシュは、組立ラインのキャリブレーション装置にゼロインストールソリューションを用いています。

自動車：自動車用モジュールにゼロインストール機能を搭載すれば、ディーラーやメカニックが簡単に自動車の問題点を診断できます。

ヘルスケア：ヘルスケア業界の多くのリーディングカンパニーが、医療用画像ビューアやカルテ管理、診断情報の共有などにゼロインストールソリューションを活用しています。

スマートホーム：ゼロインストールのスマートホームハブなら、ユーザーアプリよりも高度なUIを利用でき、技術者がホームシステムと通信して設定を行うことができます。

高度な組み込みシステムで構成され、ユーザー（オペレーター）や技術者が監視・制御・診断などを行うような製品のほぼすべてにおいて、ゼロインストールソリューションの実装が可能です。ゼロインストールは、高度な製造業や航空宇宙、農業、建設、家電、医療、資源採取、ロボティクス、運輸など、あらゆる産業領域への導入が可能です。

ゼロインストールの進化

以前はプリンターを購入するとインストールディスク、設定用ソフトウェア、ドライバーが付属していたのを覚えている方もいらっしゃるでしょう。このような付属品を使ったインストールプロセスはすべて面倒なものですが、1つのハードウェアを大勢で共有するような場合、プロセスは極めて煩雑になります。そこで初期のゼロインストールソリューションは、組織のネットワーク上で全従業員のソフトウェアを保守する手間を軽減するために、プリンターやネットワークストレージデバイスといった共有デバイスにウェブサーバやウェブアプリケーションを提供していました。また、ストレージのコストがかかるため、ウェブページ設定機能は高価なデバイスにしか備わっていませんでした。当時と比べると、現在のゼロインストールソリューションはさまざまな点で進化を遂げています。

ユーザー: デバイスと通信する必要があるのはITチームだけではありません。デバイスの種類により、診断の専門家や特殊技能を持ったオペレーター、消費者などもデバイスとの通信を必要とすることがあります。

製品: これまで、ゼロインストールは高額製品に限定された機能でした。しかしほぼあらゆる組み込みデバイスに通信機能が装備されるようになった現在では、インターネット通信機能とウェブサーバ実行容量を持ったIoTデバイスでも、ゼロインストールが可能になっています。

ルック&フィール: 旧世代の組み込みデバイスは、見た目の冴えない原始的なUIが当たり前でした。現在はウェブテクノロジーが標準化され、UIも高性能になり、見た目も改善されています。

最終成果: 現在のゼロインストールソリューションは、かつてのように設定や監視といった特定のタスクに用途が限定されません。デザイン面でも性能面でもデスクトップアプリケーションに負けない、多様な機能を備えたブラウザベースのアプリケーションを提供します。

ユーザーが享受するメリット

現代のゼロインストールソリューションは、製品の各種機能へすぐにアクセスできるという大きなメリットをユーザーにもたらします。従来新しい製品を使いだして数か月の間に生じがちであった問題は、見過ごせるものではありません。これはユーザーエクスペリエンスと製品の有用性を低下させ、ひいてはブランド力に悪影響を及ぼします。製品に「きちんと動く」ことを期待するのは人の本能と言っても過言ではありません。従って、一般消費者向けであれ、技術者向けであれ、優れたUXは製品に不可欠な要素だと言えます。

従来のゼロインストールソリューションは、リモートデスクトップからデバイスを監視する方法を提供することで主に評価されてきました。しかしながら現在、ユーザーはデバイスをオフィスでも、自宅でも、移動中でも制御できる環境を必要としています。ユーザーが持つモバイルデバイスで製品を制御するのを可能にすることで、ゼロインストールは工場の専用コンピューターからインターネットカフェに至るまで、ユーザーがどこにいてもアクセスできる環境を提供します。

ユーザーがモバイルデバイスをUIとして使えるということは、追加の機器も不要になるということです。ゼロインストールソリューションでは、製品の診断や設定、修理用の専用ツールを導入しなくても、製品の各種機能を使うことができるのです。

ゼロインストールはユーザーが自由に移動し、どこからでもアクセスできる環境を提供します。

メーカーが享受するメリット

デバイスメーカーが製品にゼロインストール機能を搭載するべき理由は多数あります。第一に、ゼロインストールアプリケーションは当然ながらクロスプラットフォームであるため、1つのソリューションを開発するだけで製品の全ユーザーに対応することができ、WindowsやMac、iOS、Android、Linux向けに個別に開発する必要がありません。また、個々のユーザーのプラットフォームに対応しつつ、面倒なインストールプロセスを省くことで、より合理的で高品質なユーザーエクスペリエンスを提供でき、製品の差別化を図ることができます。さらに、共有ライブラリの互換性がない、ユーザー認証が適切でないといった、インストールプロセスで生じがちなトラブルも防げます。

ゼロインストールを避けるべきケース

ゼロインストールがUXのフリクションを軽減し、高機能製品の開発プロセスを容易にするなら、直近システム開発でゼロインストールを採用しない手はないはずです。しかし、ゼロインストールを避けるべきケースがあるのも事実です。

ゼロインストールはUXのフリクションを軽減し、高機能製品の開発プロセスを容易にします。

非ネイティブUI
すべてのユーザーに共通のアプリケーションを提供するということは、すべてのデバイスでUIも共通になるということです。多くの場合、このような一貫性はプラスに働きますが、UIにネイティブなルック&フィールを持たせることはできなくなります。また、ブラウザ環境外でUIとネイティブデバイスのテクノロジーを統合することもできません。従って、プラットフォームに適したUIやプラットフォームに固有のセンサーが不可欠な製品を開発するなら、ゼロインストールソリューションは避け、プラットフォームごとにコンパニオンアプリを開発するべきでしょう。

ユーザーインタラクションがない
センサーやエッジデバイスなど、デバイスによってはユーザーと直接通信する必要がないものもあります。こうした小型のデバイスは大規模システムの一部であることが多く、ゲートウェイデバイスやクラウドアプリケーションに接続されています。また、これらの小型デバイスに個別にアクセスする必要性もないので、UIも不要です。小型デバイスがMQTTやSNMPといった標準プロトコルを用いている場合も同様です。標準プロトコルに準拠した既存のダッシュボードアプリケーションと小型デバイスの通信環境が整っていればよいので、さらにゼロインストールアプリケーションをホストする意味はありません。

TCP/IPがない
WiFiやEthernet、モバイルデータ通信をサポートしない組み込みデバイスのケースでは、ゼロインストールは避けるべきです。BTLEを用いる低電力デバイスや、ZigbeeやZ-Waveなどのメッシュネットワークで通信するデバイスでも同じことが言えます。

ストレージがない
オンボードのマイクロコントローラフラッシュメモリを使う小型デバイスや低コストのデバイスでは、ウェブサーバやウェブアプリケーションをホストできる十分なストレージがない場合があります。（デバイス自体に十分な容量がなく、それでもゼロインストール機能を搭載したい場合には、クラウドサーバでアプリケーションをホストするというアプローチがあります。）

接続性が悪い
環境によっては、接続性が悪い場合や、接続性に制約があるケースがあります。インフラの不足、セキュリティ上の問題、接続距離など、その原因にかかわらず、デバイスとそのネットワーク環境を踏まえた上で、接続性の問題を理由にゼロインストールソリューションを避けるべきか否かを判断する必要があります。

デバイスアクセスが制限されている
デプロイメントアーキテクチャによっては、ゼロインストールUIで組み込みシステムのハードウェアに直接アクセスができない場合があります。このような場合には抽象化レイヤーでUIとハードウェアを接続する必要が生じるため、開発時間が長くなり、複雑性が増し、ソリューションのレイテンシーが高くなってしまいます。（なお、Qt WebGLではこのような問題は生じません。詳細は後半でご説明します。）

最新ゼロインストールテクノロジーポートフォリオ

旧世代の組み込みデバイスの分かりにくく扱いにくいインターフェースは、すべて旧バージョンのHTTPやHTMLの仕様がデバイスの限界となっています。かつての標準的UIでは、もはやユーザーのニーズを満たすことはできません。使い勝手も見た目もよく、汎用性に優れたUIの進化により、ユーザーの期待が一層高まっているからです。しかし、ゼロインストールアプリケーションの開発では、さまざまな最新テクノロジーを活用することが可能です。

HTML5
HTML5マークアップ言語は、特にJavaScriptやCSS3と組み合わせた場合、ウェブ黎明期に比べて著しく機能が向上しています。HTML5は現在、Google DocsやAirtable、Pixlrといったインタラクティブアプリにも対応しています。

WebGL
グラフィカルな表示や3Dの表示には、WebGLを利用することができます。これはOpenGL ESを基盤とするブラウザベースの2D/3DグラフィックAPIで、グラフィックコンテンツをブラウザ表示することが可能です。Web GLアプリケーションの例として、Google Maps、3DプリントプラットフォームのThingiverse、BioDigital Human Platformがあります。

WebAssembly
WebAssemblyでは、ブラウザで実行できない言語（CやC++）をクロスコンパイルし、WebAssembly対応のブラウザと互換性のあるフォーマットに変換します。現在では、Google ChromeやMozilla Firefox、Opera、Microsoft Edge、Apple Safariといったメジャーなブラウザがすべてこれに対応しています。WebAssemblyは保護された仮想環境で動作しますが、ほぼネイティブな実行速度を目指しており、こちら（https://www.qt.io/qt-examples-for-webassembly）でいくつかの例をご覧いただけます。

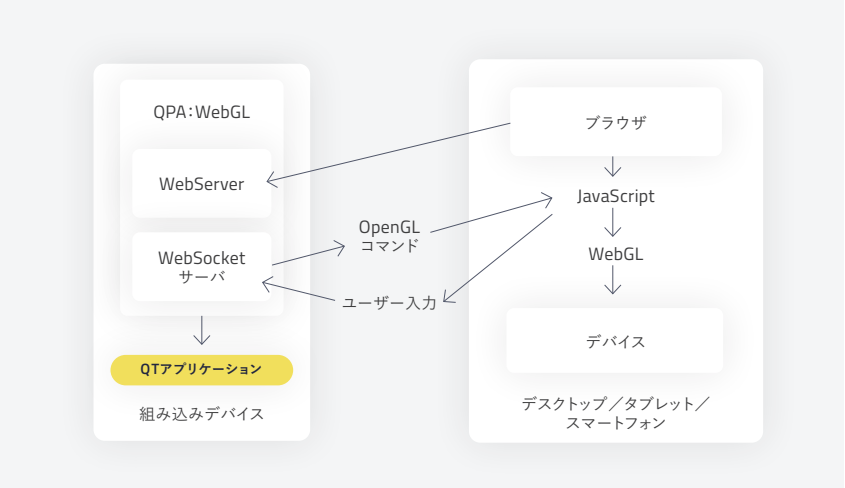
WebSocket API（WebSockets）
ブラウザアプリケーションの多くはHTTPで組み込みデバイスと通信しますが、これは本来、HTTPと組み込みアプリケーションサーバに備わった機能ではありません。WebSocketは2012年頃に誕生した新しいテクノロジーで、クライアント側のブラウザアプリが組み込みアプリケーションサーバと直接通信するため、パフォーマンスやレイテンシーの改善、セキュリティの向上、合理的な実装といったメリットが得られます。

Qtを使ったゼロインストールソリューション

読者の皆様はすでにQtフレームワークのさまざまな強みをご存じかと思います。たとえば、開発者、ライブラリ、ツールについて、大規模なエコシステムをもっている、クロスプラットフォームアプリケーションの開発が得意である、C++のネイティブパフォーマンスを提供できるといった強みがあります。Qtは極めて高機能なC++フレームワークと言えますが、では、ブラウザベースのゼロインストールアプリケーションの開発には適しているのでしょうか？
その答えは、もちろん「適している」です。C++のQtアプリケーションをブラウザベースバージョンに変換するには、WebGLを使うアプローチとWebAssemblyを使うアプローチがあります。では具体的にどのように行うのか、2つのテクノロジーについて詳しく見てみましょう。

WebGL

Qt WebGLアプローチでは、Qtレンダリングコマンドを WebGLストリームに変換後、リモートブラウザに送信して、組み込みデバイス上で表示を描画します。開発者にとってはごく簡単な作業で、たいていの場合、起動引数(-platform webgl)を追加するだけで完了します。



Qt WebGLブロック図

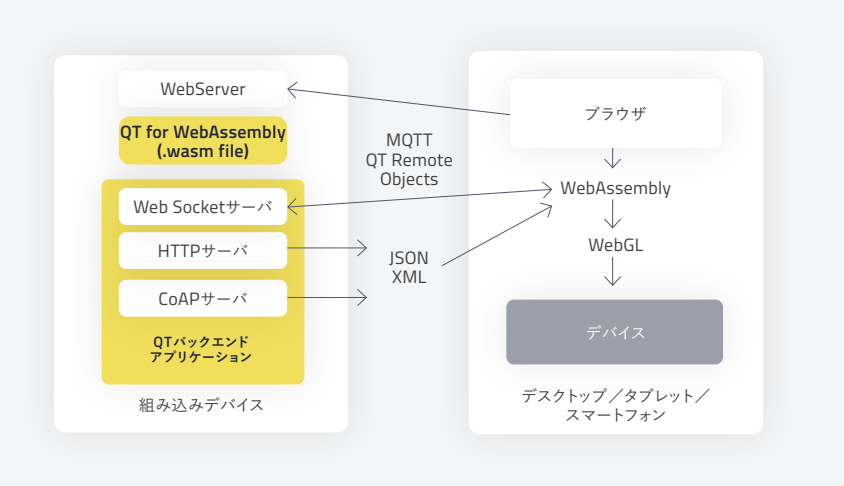
図の通り、このシンプルなコマンドを支援するために、バックグラウンドではさまざまなことが行われます。まず、組み込みデバイスがゼロインストールアプリケーション用の軽量ウェブサーバとコマンドの送受信を行うWebSocketサーバを起動します。ブラウザがウェブサーバに接続すると、サーバがJavaScriptの小さなフロントエンドアプリケーションを送信し、組み込みデバイスのWebSocketサーバへの接続が確立され、コマンドの送受信が行われます。

一方、組み込みデバイスはQtアプリケーションを開始し、QPAプラグインはUIの生成するOpenGL ES描画コマンドをインターセプトします。描画コマンドはすべてシリアライズされたWebGLストリームに変換され、WebSocket接続を介してJavaScriptアプリに送信されます。JavaScriptアプリは描画コマンドを受け取り、WebGL APIを使って、組み込みデバイスのUIをブラウザ内に描画します。このWebSocketチャネルはJavaScriptアプリがユーザーのマウス／キーボードコマンドを受け取る際にも使われ、コマンドはサーバに送り返されてユーザーインターフェースループが完成する仕組みです。

このソリューションでは、サーバ側で多くの作業が行われます。その点が、クライアント側のJavaScriptプログラミングで表示を描画する一般的なWebGLアプリケーションと違うところです。このソリューションのほうが簡素化できる側面もある一方、いくつかの制約が生じてしまうのも事実で、詳細は後半の比較の部分でご説明します。

WebAssembly

Qt WebAssemblyアプローチでは、標準的なQtアプリケーションをEmscriptenコンパイラでリコンパイルし、WebAssemblyバイナリを生成します。
これをクライアントに送ってブラウザのコンテナで実行する仕組みです。クライアントのブラウザがアプリ実行の処理能力を提供するため、多くのクライアントを同時にサポートするのが簡単になります。



Qt WebAssemblyの
ブロック図

WebAssemblyソリューションのブロック図は、WebGLと構造的にはよく似ています。ただしWebAssemblyソリューションでは、アプリケーションバイナリ用の軽量ウェブサーバと、クライアントとの接続用のサーバが必要です。

またQt WebGLソリューションと異なり、プログラマーがアプリケーションと組み込みデバイスのインタラクションを管理する必要性も生じます。解決方法としては、きめ細かいモジュラーサービスを提供するマイクロサービスアーキテクチャを利用し、クライアントアプリが実行する個々のタスクを処理する手法です。この方法では、クライアントのブラウザアプリケーションがリクエストをまとめ、組み込みデバイスに送信します。デバイスのサーバは受け取ったリクエストパケットを開き、実行します。（軽量な通信プロトコルとして、Qtクラスがあって使いやすいMQTTまたはJSONがあります。）もう1つの方法がQtRemoteObjectsで、Qtスロット／シグナルメカニズムで機器間の操作を担います。

WebGLとWebAssemblyの比較

	WebGL	WebAssembly
仕組み	デバイスがシリアライズされたWebGLデータストリームをクライアントブラウザに送り、リモートUIを描画	デバイスがWebAssemblyバイナリをクライアントブラウザに送り、アプリをブラウザ内で実行
既存のQtアプリケーションの変換	極めて簡単。クライアント側のアプリケーショングラフィックは自動的に処理され、ダイレクトなデバイス制御にも影響がない	シンプル。変換は主にWebAssemblyツールチェーンを使ったリコンパイルで行う。ハードウェアに直接アクセスするコンポーネントはマイクロサービスアーキテクチャ（または類似品）対応に変換する必要がある
アプリケーションの速度	組み込みハードウェアによって決まる	クライアントデバイスのハードウェアによって決まる
UIの反応性	UIは組み込みデバイス上で実行され、クライアントデバイスに送られる。従って反応性は、ネットワーク回線容量とパケットのレイテンシーによって決まる	UIはクライアントデバイスで直接実行される。従って反応性は、クライアントハードウェアの仕様とバックグラウンドで実行されるその他のクライアントアプリケーションによって決まる
初回起動時間	ほぼ瞬時と言える速さ。クライアントのブラウザが小さなスタブを受信するだけで最初のフレームを表示できる	数秒かかる。WebAssemblyアプリケーション全体をクライアントデバイスに送り、ブラウザでコンパイルしてから表示
同時クライアントの接続数	1台。マルチプロセスのQt WebGLを現在開発中	無制限
リバースエンジニアリングの難易度	クライアントデバイスがWebGLデータストリームしか受け取れないため、クライアント側のリバースエンジニアリングは不可能	クライアントデバイスに送られたWebAssemblyコードはリバースエンジニアリング可能。ただし、HTML5/ JavaScriptのリバースエンジニアリングに比べると難易度が高い
制約	アプリケーションがOpenGL描画でない場合は活用できない（例：Qt Widgets）	
	アプリケーションにサードパーティのソースのないモジュールが含まれる場合は活用できない（すべてのコードをWebAssemblyにリコンパイルする必要があるため）	

Qtゼロインストールテクノロジースタック

では、Qtでゼロインストールソリューションを開発するには具体的に何が必要でしょうか？

まずは、デザイナーがグラフィックを作るAdobe PhotoshopまたはAdobe Sketchが必要です。インポートしたグラフィックからアプリケーションの各画面用に宣言型言語のQMLを書くQt Design Studioも必要です。

Qtアプリケーション開発用のコンパイラやツール、ライブラリを提供するQt Creatorも必要です。この統合開発環境（IDE）を使い、デザイナーの作成したグラフィックからQMLまたはC++でQtアプリケーションの開発を行います。
（注：Qt for WebAssemblyでは、WebAssembly用にリコンパイルを行うQtソースをダウンロードする必要があります。）

UI開発用の多様なコンポーネントやツール（QML、Widgets、OpenGL、SVG、Qt 3D）を提供するQt for Application Developmentも必要です。Qt for Application Developmentは、ハードウェア通信の制御（ネットワーキング、WebSockets、Bluetooth、シリアルポート、CAN、Modbus、センサー）、およびデータ管理（SQL、XML、イメージフォーマット）用のコンポーネント／ツールも提供します。

Qt for Device Creationも必要です。組み込みデバイス上での開発、デバッギング、デプロイメントを行う各種コンポーネントを提供するパッケージで、クロスコンパイレーションツールチェーン、USB経由のデバッギング用ソフトウェア、Boot to Qtソフトウェアスタック、Qt for RTOS、参照用ターゲットプラットフォームで構成されます。Qt WebGLおよびQt for Web Assemblyコンポーネントも含まれるため、開発者はどちらのタイプのソリューションも構築することができます。

最後に、Qtゼロインストールソリューション開発に不可欠なコアパーツがQt for Automationです。Qt for Automationは組み込みデバイスとクライアントアプリケーションの通信に必須のQt MQTT、Qt OPC UA、Qt KNX、Qt CoAP、Qt Remote Objectsといったコンポーネントで構成されるテクノロジーです。

ゼロインストールとクラウド

組み込みデバイスにゼロインストール機能を搭載するもう1つのアプローチが、クラウドでホストするという方法です。柔軟性に優れたQtなら、必要なツールに変更を加えずにこのアプローチを採用することができます。クラウドベースのアプリケーションは、大きなデータ容量が必要となる複雑なアプリケーションをホスト／運用するための性能が組み込みハードウェアにない場合、もしくはアプリケーションがクラウド上のデータリソースにアクセスする必要がある場合に適したアプローチです。

クラウドでホストするアプローチの弱点は、接続性です。組み込みデバイスがクライアントアプリケーションをホストするなら、クライアントブラウザとデバイスの接続に問題はありません。しかし、クラウドベースのアーキテクチャでは、クライアント機器をプロキシやVPN、ファイヤウォールといった干渉のないインターネットに接続する必要があります。また、クライアントブラウザから組み込みデバイスへの接続も別途必要になります。この問題を解決するのは不可能ではありませんが、「フリクションがない」というゼロインストールならではのメリットは若干損なわれることになります。

クラウドアプローチの大きなメリットとして、ネットワーク上の多数のデバイスからの入力を1つのモデルに統合し、デジタルツインを作成できるという点が挙げられます。デジタルツインは実際のシステムやオブジェクトからセンサー入力を受け取り、リアルタイムの仮想レプリカを作ります。オブジェクトの先行保全や運用効率、教育情報などを遠隔でモニタリングすることを可能にするテクノロジーです。デジタルツインテクノロジーは現在、製造やヘルスケア、自動車、ビルオートメーション、宇宙、防衛といったさまざまな分野で導入が進んでいます。

機能的安全性とサイバーセキュリティ

サイバーセキュリティの懸念に対しては、ゼロインストールはどのように対処しているのでしょうか?ITチームは通常、アプリケーションソフトウェアのインストールを行う際、自社のファイヤウォールとユーザー認証の強度を下げることになります。しかしゼロインストールアプリケーションはブラウザ上で実行されるため、IT環境を確実にコントロールしながら、ユーザーにコンパニオンアプリへのアクセスを許可することができます。ブラウザが安全な仮想環境を提供するので、クライアントへの攻撃に対して、クライアントコンピューターを隔離した状態を維持できます。またハードウェアとクライアントの接続には、機器間のデータ通信に安全な暗号化／認証ソケットが用いられます。

ゼロインストールは、容易な導入、手軽な保守、クロスプラットフォーム、モビリティ機能など数多くのメリットをユーザーにもたらします。

ゼロインストールは、リバースエンジニアリングの防止という意味でも有効です。WebGL実装では、クライアント側からのリバースエンジニアリングは不可能です（アプリケーションデータは送信されず、WebGLレンダリングコマンドのみ送信されるため）。WebAssemblyアプリケーションはクライアントに実行ファイルを送信しますが、生成されるバイナリコードは、HTML5/JavaScriptの生成するコードと違って容易に解読することはできません。

Qtでは、Qt Safe Rendererを使って産業用（IEC 61508）、自動車用（ISO 26262）、医療用（IEC 62304）の認証取得可能なシステムを開発できます。ただしWebAssemblyを使ったQtゼロインストールでは、組み込みシステムに一切のレンダリングが不要です。ヘッドレスソリューションのため、認証もずっと簡単です。（残念ながらWebGLを使ったQtゼロインストールはそこまでシンプルではなく、認証の取得も容易ではありません。）

まとめ

ゼロインストールは、容易な導入、手軽な保守、クロスプラットフォーム、モビリティ機能など数多くのメリットをユーザーにもたらします。現在、多くの製品開発者がQtを使ってゼロインストールソリューションを構築しています。Qtは優れたユーザーエクスペリエンスの開発を可能にし、それと同時に、クロスプラットフォームならではの高い柔軟性、信頼性に優れた高性能ツール、大規模な開発者エコシステムを提供して、開発プロセスの短縮化と合理化も実現します。

さらに、Qt for WebGLまたはQt for WebAssemblyを使えば、既存のネイティブQtアプリケーションをブラウザベースのゼロインストールアプリケーションに再利用することも可能です。実装の容易さやシングルコードベースといった利点に加え、Qt基盤のゼロインストールアプリケーションはセキュリティやパフォーマンスの向上など、さまざまなメリットをもたらします。

次のプロジェクトではゼロインストールが最適解かもしれないとお考えの場合にはぜひ、The Qt Company（japan@qt.io）にお問い合わせください。具体的なアプローチの検討をお手伝いします。

お問い合わせ：
The Qt Company　日本オフィス
〒100-0005
東京都千代田区丸の内3-3-1新東京ビル2F

Web: https://www.qt.io/jp/
Email: japan@qt.io
TEL: 03-6264-4500