push notifications in a Blazor PWA application

---

1. What Are Push Notifications?

Push notifications are **server-initiated messages** delivered to a user's device **even when the application is not open**. In a Blazor PWA, they are handled by the **browser**, not by .NET directly.

Key characteristics:

- Work when the app is **closed**
- Delivered via the **browser's push service**
- Displayed using a **Service Worker**
- Require **explicit user permission**

---

2. Why Push Notifications in Blazor PWA?

Blazor PWAs use push notifications for:

- Order status updates (e-commerce)
- Alerts and reminders
- Background system notifications
- Re-engagement of users

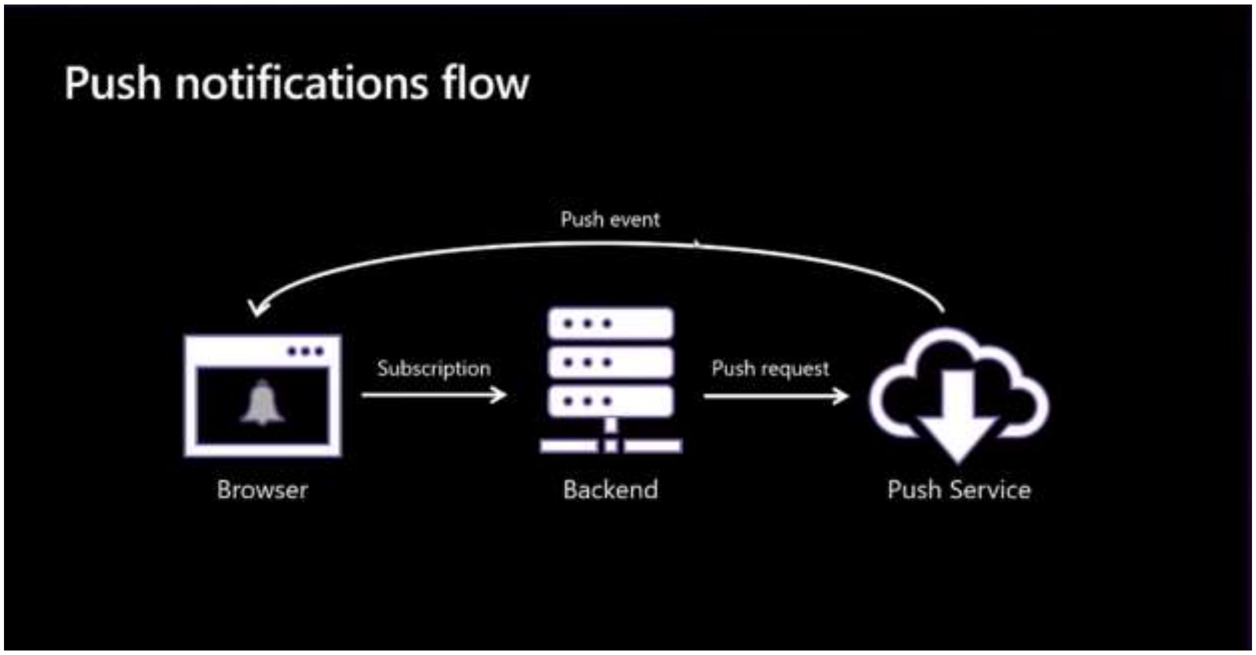
Because Blazor runs on WebAssembly, **push notifications are implemented using standard Web APIs**, not Blazor-specific APIs.

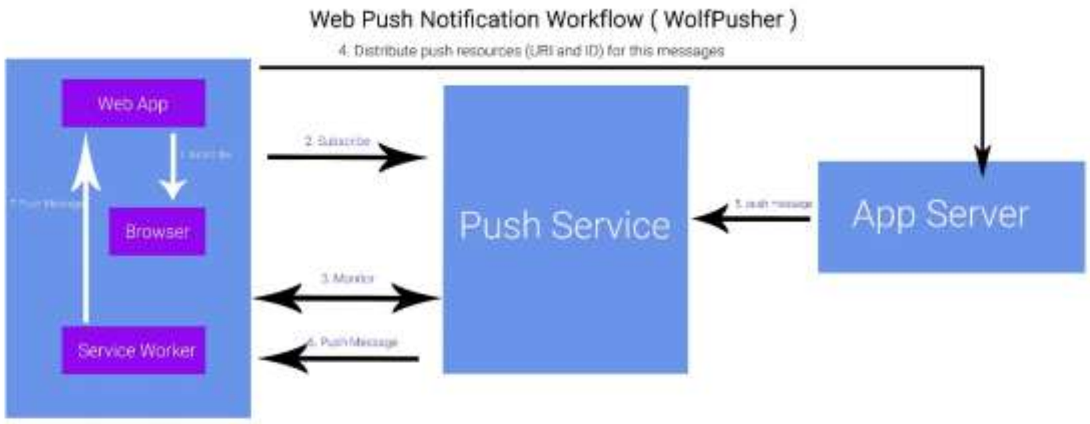
---

3. Core Architecture

Push notifications in a Blazor PWA involve **four actors**:

1. **Blazor WebAssembly App**
2. **Service Worker**
3. **Browser Push Service** (Chrome, Edge, Firefox)
4. **Application Server** (ASP.NET Core API)





---

4. Key Components Explained

4.1 Service Worker (Mandatory)

A **Service Worker** is a background JavaScript file that:

- Runs independently of the Blazor app
- Receives push events
- Displays notifications

In Blazor PWA, this is typically:

wwwroot/service-worker.js

Without a service worker, push notifications **cannot work**.

4.2 Push Subscription

A **push subscription** uniquely identifies a browser + device.

It contains:

- Endpoint URL
- Public key
- Authentication secret

This subscription is:

- Created in the browser
- Sent to your backend
- Stored for later use

4.3 VAPID Keys

VAPID (Voluntary Application Server Identification) keys are used to:

- Authenticate your server with the browser push service
- Prevent push abuse

They include:

- **Public key** (shared with client)
- **Private key** (kept on server)

---

5. High-Level Flow (Step-by-Step)

Step 1: User Opens Blazor PWA

The app loads and registers the service worker.

Step 2: Request Permission

The browser prompts:

“Allow notifications?”

Step 3: Create Push Subscription

If allowed:

- Browser generates a subscription
- Blazor app sends it to backend API

Step 4: Store Subscription

Backend saves subscription in database.

Step 5: Server Sends Push Message

Backend sends a message using:

- Subscription endpoint
- VAPID private key

Step 6: Service Worker Displays Notification

The browser wakes the service worker and shows the notification.

---

6. Where Blazor Fits In

Blazor’s role is **orchestration**, not delivery.

Blazor handles:

- Requesting permission
- Calling JavaScript via JS interop
- Sending subscription data to backend

JavaScript handles:

- Push APIs
- Service Worker events

Server handles:

- Message generation
- Secure push delivery

---

7. JavaScript Responsibilities

Push notifications rely on **Web Push APIs**, which are **not available directly in .NET**.

JavaScript is responsible for:

- Notification.requestPermission()
- serviceWorkerRegistration.pushManager.subscribe()
- Handling push events
- Displaying notifications

Blazor calls these via **JavaScript Interop**.

---

8. Service Worker Push Handling (Conceptual)

Inside service-worker.js:

```
self.addEventListener("push", event => {
  const data = event.data.json();


  self.registration.showNotification(data.title, {
    body: data.message,
    icon: "/icon-192.png"
  });
});
```

This runs **even if Blazor is not loaded**.

---

9. Backend Role (ASP.NET Core API)

The backend:

- Stores push subscriptions
- Sends notifications using Web Push protocol
- Uses VAPID private key
- Can trigger notifications from:
  - Background jobs
  - Business events
  - Scheduled tasks

Blazor **never sends push messages directly**.

---

10. Limitations & Constraints

Important constraints to understand:

Limitation	Explanation
HTTPS required	Push works only over HTTPS
Browser support	Not supported on iOS Safari (limited support improving)
User permission	Mandatory and revocable
No guarantee	Delivery is best-effort
Payload size	Typically < 4 KB

---

11. Blazor Server vs Blazor WASM

Feature	Blazor WASM PWA	Blazor Server
Push notifications	Fully supported	Not applicable
Offline support	Yes	No
Service Worker	Required	Not used

Feature	Blazor WASM PWA	Blazor Server
Background execution	Yes	No

Push notifications are **only practical in Blazor WebAssembly PWA**.

---

12. Common Use Cases in Blazor PWA

- Order shipped notification
- Session expiry warning
- New message alerts
- System maintenance notices
- Daily reminders

---

13. Summary

- Push notifications in Blazor PWA rely on **browser standards**, not Blazor APIs
- **Service Workers** are the backbone
- Blazor coordinates via **JavaScript Interop**
- Backend handles **secure message delivery**
- Works even when the app is closed
