Extending W3C ML Work to Embedded Systems

September 10, 2020
For the “W3C Workshop on Web and Machine Learning”

Peter Hoddie
Co-Founder, Moddable Tech
Chair, Ecma TC53

About Me

- Not an expert in Machine Learning
- Extensive experience with JavaScript for embedded devices
  - Delivering commercial embedded products with JavaScript for over a decade
  - Chair of Ecma TC53 – ECMAScript Modules fro Embedded Systems – creating standard JavaScript APIs for embedded systems

Beyond the Web

- JavaScript's dominance on the web obscures its success in other domains
    - Node.js is part of the Web Platform, though it isn't part of the browser
- Many other devices can and do run JavaScript
    - Expanding the scope of the W3C Machine Learning APIs would accelerate overall ecosystem

“Resource constrained embedded device”

- Very low cost microcontrollers — a few dollars, at most
- Not powerful enough to run Linux or Node.js
- Powerful enough to run standard JavaScript (ECMAScript 2020!)
- On track to integrate hardware accelerated Machine Learning features in the coming years

What APIs for Machine Learning?

- Silicon manufacturers will likely provide their own proprietary native APIs
    - Optimized for their hardware
    - Not portable
- Standard JavaScript APIs would be welcome
    - Portable
    - Well designed
    - Bridge to the web

JavaScript ML API Considerations

- Is it realistic to use the same API on the web and embedded devices?
    - Web hosts are 100x more powerful than many embedded devices
- If they cannot be identical, is there a way to bridge the two worlds?

Pattern #1:
Same API on Web & Embedded

- Some APIs can be supported in both environments
- W3C Sensor API is a good example
    - Implemented in web browsers
    - Implemented on embedded devices
- Ecma TC53 supports through a low level JavaScript Sensor API
    - Data formats normatively compatible with W3C Sensors for simplicity

Pattern #2: Different APIs with a Bridge

- Serial supported by both Ecma TC53 and Chrome
    - Different APIs
    - Many of the same API conventions
    - Nearly identical functionality
- Chrome API is too heavy for embedded
- Two worlds are bridged by a JavaScript implementation of TC53 Serial using Chrome Serial
    - Allows serial client code sharing between embedded and web

Anti-Pattern: “Light” version of Web API

- Light versions of libraries are a common idea
  - Fewer features for less powerful devices
- They almost always are a failure
  - Developers expect the full version
  - Developers try to use the light version to do more than it was designed for
- Better approach is to have two separate APIs
  - Each optimized for its target device class
  - Avoiding needless differences (e.g. use the same terminology)

Closing

- Great potential in extending the W3C Machine Learning JavaScript API initiative to embedded
    - Expand the reach of web developers to edge devices
    - Different approaches presented here for how that might be done
- Thank you!

@phoddie
@moddabletech