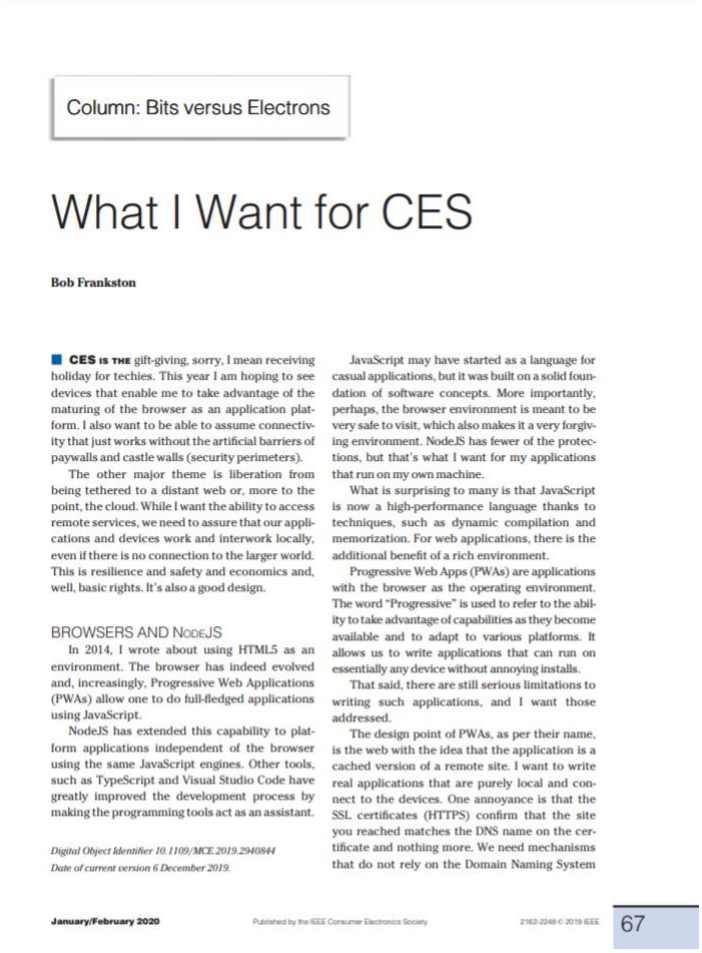
What I want for CES



Column: Bits versus Electrons

What I Want for CES

Bob Frankston

CES is the gift-giving, sorry, I mean receiving holiday for techies. This year I am hoping to see devices that enable me to take advantage of the maturing of the browser as an application platform. I also want to be able to assume connectivity that just works without the artificial barriers of paywalls and castle walls (security perimeters).

The other major theme is liberation from being tethered to a distant web or, more to the point, the cloud. While I want the ability to access remote services, we need to assure that our applications and devices work and interwork locally, even if there is no connection to the larger world. This is resilience and safety and economics and, well, basic rights. It's also a good design.

BROWSERS AND NODEJS

In 2014, I wrote about using HTML5 as an environment. The browser has indeed evolved and, increasingly, Progressive Web Applications (PWAs) allow one to do full-fledged applications using JavaScript.

NodeJS has extended this capability to platform applications independent of the browser using the same JavaScript engines. Other tools, such as TypeScript and Visual Studio Code have greatly improved the development process by making the programming tools act as an assistant.

Digital Object Identifier 10.1109/MCE.2019.2940844
Date of current version 6 December 2019.

JavaScript may have started as a language for casual applications, but it was built on a solid foundation of software concepts. More importantly, perhaps, the browser environment is meant to be very safe to visit, which also makes it a very forgiving environment. NodeJS has fewer of the protections, but that's what I want for my applications that run on my own machine.

What is surprising to many is that JavaScript is now a high-performance language thanks to techniques, such as dynamic compilation and memorization. For web applications, there is the additional benefit of a rich environment.

Progressive Web Apps (PWAs) are applications with the browser as the operating environment. The word "Progressive" is used to refer to the ability to take advantage of capabilities as they become available and to adapt to various platforms. It allows us to write applications that can run on essentially any device without annoying installs.

That said, there are still serious limitations to writing such applications, and I want those addressed.

The design point of PWAs, as per their name, is the web with the idea that the application is a cached version of a remote site. I want to write real applications that are purely local and connect to the devices. One annoyance is that the SSL certificates (HTTPS) confirm that the site you reached matches the DNS name on the certificate and nothing more. We need mechanisms that do not rely on the Domain Naming System

January/February 2020 Published by the IEEE Consumer Electronics Society 2162-2248 © 2019 IEEE 67

This column is now available at the IEEE via it's DOI.

CES is the gift-giving, sorry, I mean receiving holiday for techies. This year I'm hoping to see devices that enable me to take advantage of the maturing of the browser as an application platform. I also want to be able to assume connectivity that just works without the artificial barriers of paywalls and castle walls (security perimeters).

The other major theme is liberation from being tethered to a distant web or, more to the point, the cloud. While I want the ability to access remote services, we need to assure that our applications and devices work and inter-work locally even if there is no connection to the larger world. This is resilience and safety and economics and, well, basic rights. It's also good design.

Browsers and NodeJS

In 2014 (https://rmf.vc/IEEEHTML5) I wrote about using HTML5 as an environment. The browser has indeed evolved and, increasingly, Progressive Web Apps (PWAs) allow one to do full-fledged applications using JavaScript.

NodeJS has extended this capability to platform apps independent of the browser using the same JavaScript engines. Other tools such as TypeScript and Visual Studio Code have greatly improved the development process by making the programming tools act as an assistant.

JavaScript may have started as a language for casual applications, but it was built on a solid foundation of software concepts. More important, perhaps, the browser environment is meant to be very safe to visit, which also makes it a very forgiving environment. NodeJS has fewer of the protections, but that's what I want for my applications that run on my own machine.

What is surprising to many is that JavaScript is now a high-performance language thanks to techniques such as dynamic compilation and memorization. For web applications, there is the additional benefit of a rich environment.

PWAs are applications with the browser as the operating environment. The word "Progressive" is used to refer to the ability to take advantage of capabilities as they become available and to adapt to various platforms. It allows us to write applications that can run on essentially any device without annoying installs.

That said, there are still serious limitations to writing such applications, and I want those addressed.

The design point of PWAs, as per their name, is the web with the idea that the application is a cached version of a remote site. I want to write real apps that are purely local and connect to devices. One annoyance is that the SSL certificates (HTTPS) confirm that the site you reached matches the DNS name on the certificate and nothing more. We need mechanisms that don't rely on the DNS and which can be associated with other properties, such as who owns the device. This needs design work but for now, all I want is a simple way to use HTTPS for local devices since the PWA specifications require that all connections are made using SSL.

I also want a browser file system that works across applications. Today I can share files across applications by using a cloud service such as iCloud, Dropbox, Google Drive, etc. Here too, we see the limitations of the web as the design point. Such applications can't cooperate in the absence of a remote connection.

These file systems have more sophisticated trust models and replication mechanisms than traditional file systems. It is time to bring these powerful ideas back to local systems.

This would help make the browser not simply an operating environment independent of the particular platform but one that is a meta operating system in the sense of working across platforms as well as being independent of the particular browser.

For devices, I just need NodeJS capabilities rather than a full browser. I'm working with ECMA TC53 which is focused on JavaScript for small devices. Companies like Moddable and Espruino are providing JavaScript-based development platforms.

Devices

The modern smartphone is a powerful platform. For under $100, I can buy a device with the browser as a programming environment complete with a display screen, multiple sensors, and interfaces. The Raspberry Pi is another example of a very general-purpose programmable device at a low price.

The very term "smartphone" is misleading since the device has become a cognitive prosthetic like eyeglasses in helping us see and interact with the world. QR codes are part of this being visible links to rich information.

I don't just use the devices as viewers. I build new "things" such as a lighting controller. (Formerly a light switch). Why do I need to fish a device out of my pocket when I can have one mounted on the wall that has the appropriate form and functionality for the place? I can use the browser as a programming environment to make it be a simple light switch or a sophisticated motion sensor. By using software to define my device I can share in the benefits of generic technology sold in the billions.

Even something as simple as the battery becomes a problem when a device is mounted on the wall. The battery can become bloated if overcharged. This is an increasingly common use case, and the devices should have an always-on mode which manages the battery. I don't want to get rid of the battery since it is useful to have a device that works even when the power is out. At very least I now have flashlights readily available. I mount the devices as soft infrastructure using 3M Command Strips which means I can take it off the wall and use it as a flashlight.

I want additional form factors. For small wrist-mounted devices I already have the option of round devices. I should also have that option for a wall-mounted device so I can, if I choose, emulate a round thermostat. Emulate is the wrong word here – with the proper software, it becomes a thermostat. Almost…

I also want the capabilities I have on a Raspberry Pi – the ability to attach sensors and other gear. And to fully access the built-in capabilities from browser apps (with suitable permission).

I currently have an array of home control apps on my phone. Fortunately, I can unify them and control the devices through my browser app coupled with a node app I've written and generic interface platforms like Hubitat. (https://hubitat.com) This is more reason to have APIs available for the devices rather than tying them all to Alexa and other such silos.

Finally, one box I want is the generic browser box that replaces specialized TV boxes. As with PWAs, why do we need specialized apps and specialized boxes? Same for cordless phones and intercoms within our homes.

Assuming Connectivity

In order to interconnect the devices, I need to be able to assume connectivity. There shouldn't be a need for onboarding just to interconnect my devices. Today I have over 200 IP (Internet) connected devices in my house (many are light bulbs). It simply isn't feasible to change the SSID. If a visitor wants to print, I shouldn't have to worry about which network the printer is on. I should just be able to give permission.

The power of IP packets is that they are fungible so that any capacity I added improves all applications. Any routers I use to extend the range benefits all devices. This isn't true of protocols like Bluetooth, Zigbee, etc. And WiFi direct doesn't have routing. We need to build on the idea of generic connectivity and improve the ability to route locally using any available wired or wireless segments without depending on single access points nor connectivity through a distant backbone.

I also want innovations in radios with techniques like spreadspectrum. There are some good ideas in Bluetooth such as low energy and location beacons we can build on.

This generic connectivity applies to USB devices. USB has two faces. One is that it is a special-purpose data path. It would be better to use generic IP connectivity with device relationships defined in software with trust based on shared secrets rather than requiring specialized wiring.

This would also benefit televisions. Imagine if we didn't need a pile of HDMI cables to connect all the TV boxes. Perhaps no wires at all as with WiFi.

Power to the device!

Broadcasting power isn't yet feasible beyond a short-range, but I can take advantage of the fact that USB has become the universal power supply. At least for a class of devices. USB-C's ability to negotiate power has enabled it

to reach up to 20 volts and 100 watts. This covers most devices we have. Most importantly, LED lights and portable computers.

It should be easy to provide USB outlets throughout the house but to take advantage of this I want USB power cords that I can tape on the wall and paint over with tiny connectors at the ends.

DC makes it practical to share and integrate disparate sources including batteries and solar and, using software protocols, to manage the flows. This is a more general model than the distribution-based one that is today's norm.

Modest Wishes

As we become dependent upon these new powerful technologies, we need to see beyond today's use cases and imagine what is possible. Because I can write my own apps, I can experience the future. I look forward to a merry CES … or, at least, I can dream of one when I land in Las Vegas.