Beyond the editor - The implications of M1 Apple Silicon for Web Developers - How architecture will affect daily development and what the future holds

Written by Seth Corker on Benevolent Bytes

If you've been on a social media blackout over the past few weeks then you may have missed the announcement, Apple's first generation of ARM SOCs are out. The M1 is now available in three Macs and first impressions are positive. The change will have a positive impact on consumers but what about web developers? What does this shift in architecture mean for developers working on the web with JavaScript, HTML and CSS? Here are a few implications that M1 will have on developers.

Availability of tools

Figure 1: Hammer and spanner cross emoji

Apple has started off strong by ensuring there is a good compatibility story when it comes to the transition from x86 to ARM based architecture. Most consumer facing applications will work through Rosetta 2, a translation layer, just fine and will likely see performance improvements once developers make Apple Silicon optimised builds. This covers many use cases but as a developer, there are some gaps.

Developer Tooling is incompatible on M1

Not everything is working on Apple Silicon and it will likely be that way for some time as developers get their hands on M1 equipped computers to test and rebuild tools. At the time of writing, here's how developer tools are fairing.

The Good Let's start with the positive, here are some relevant tools you might use as a web developer.

- **Electron**: if you're making cross platform apps with Electron, v11.0 has full native support.
- **Python**: I'm including this because it's a common language used for web servers, Python v3.8, v3.9 and even v2.7 are compatible.
- **Sublime Text**: This editor isn't native but it works via Rosetta 2

The Bad These are some common tools you might be using as part of daily development that aren't working.

- **Docker**: This is a big one if you rely on spinning up your infrastructure for local development. This is a big one if you rely on spinning up your infrastructure for local development. It's currently in available in the Preview Program.
- **Atom**: Still quite a popular editor but there's no support yet.

The Ugly Finally, the mixed bag. These two are very popular and have support coming but they just aren't ready yet for everyone.

- **Node**: v15 is supported but previous versions will require patches which aren't ready yet. If you're a node developer on an LTS release, you're out of luck right now.

- **VS Code**: Insider builds are working on M1 but so far, no final release. It's only a matter of time with Electron having M1 support.

Take a look at "Does it ARM" for compatibility of specific developer tools to see if they'll work on a shiny new M1 equipped Mac.

What does this mean for Linux?

There are implications across the industry because Apple is a large company with a lot of sway. A positive flow-on effect of putting powerful ARM processors in the hands of developers comes with an easier path for ARM tooling on other platforms. ARM isn't new and there are lots of popular devices that are built with these chips like the Raspberry Pi. The difference is that until now, the majority have been low powered devices. My hope is that with ARM processors in the hands of more developers, the barrier to making tools compatible with other devices will be lower than it is today. This might take the form of more optimised compilers, programming languages and tooling that is more efficient than what we see with x86 version today. This also introduces an issue, however, compatibility.

Incompatibility across systems

I've already discussed the issues that now exist when trying to run software on new Apple Silicon devices but what about the other way around? This migration away from what has been a common architecture for decades, will create incompatibility and limit interoperability between computers. When Apple moved to Intel based Macs, one compatibility advantage comes to mind immediately, video games. PC is still dominant in this field but due to the similarity between Macs and Windows PCs, developers have been more likely to release their games to both platforms. The barrier for entry is relatively low.

The new architecture shift is creating another platform for developer's to support and it's unlikely that this is a a market share large enough to pay attention to. It's also unclear if Apple Silicon will be powerful enough to deal with games that are more graphically intensive and visually demanding. Game developers will need to make a decision on what path they take for cross-platform titles and it might not be worth it right now.

New possibilities

Figure 2: The M1 promotional image

There are some possibilities for Apple Silicon devices that I think are going to shake things up in the industry. I'm not sure how many of them will reach web developers but they're tantalising all the same.

Media enhancements for safari

WebKit, the engine that powers Safari, now has support for a new media query. It's to detect if the display supports HDR (High Dynamic Range).

```
<style>
  @media only screen (dynamic-range: high) {
    /* HDR-only CSS rules */
  }
```

```
</style>

<script>
  if (window.matchMedia("dynamic-range: high")) {
    // HDR-specific JavaScript
  }
</script>
```

It's pretty mundane at the moment but there could be some great potential to expand in the future. Web developers have a knack for taking simple queries like these and crafting compelling experiences, one example that comes to mind is light and dark mode with CSS Media Queries, based on `prefers-color-scheme`.

Low power/high efficiency

The biggest advantage that Apple is toting with Apple Silicon is power efficiency but how does that affect web developers? We often have our machines plugged in to electricity the majority of the time. This is a win for users who can browse more websites with less battery drain while still having performant systems when needed. I'm really interested in if there will be more support for leveraging the different types of cores, either directly with an API or indirectly via the browser. There are two types of cores in M1 chips, high efficiency and high performance. High efficiency cores are for battery, they are lower power drain cores. What if we could offload background tasks to these cores? As web developers we can make websites that have smaller bundle size, more optimised images, more efficient code but we can't directly save someones battery life. If browsers start to utilise this efficiency or even better, allow developers to use these cores too, we can create a whole new metric for how performance on websites is measured.

Media encoding/decoding

The general purpose CPUs and GPUs of computer architecture we think of today have some downsides when compared with more dedicated chips. The M1 has dedicated support for ending and decoding media like video, images and audio with lower power consumption. It's unclear how this will impact web developers right now but I hope that this will lead to better video codecs and file formats that are more smaller even if they do require more processing on the client. With a wider range of options, as web developers, we can provide experiences that cater to low bandwidth environments more easily.

Conclusion

I'm looking forward to seeing how this shift plays out and that impacts it has on users and developers alike. There are some opportunities around performance and efficiency that I hope we can take advantage of soon but there are going to be some teething problems until the tooling we use everyday, is compatible with Apple Silicon. Let's see if developers take to the platform and if user's expectations of efficiency change too.

References

- See if your favourite app works on M1
- Checkout what Safari 14 has to offer
- Apple's press release on M1

Updates

This article was updated on 17/01/2021 to reflect the following changes. Since this article has been written, the following changes have been made:

Support for M1 has been added to Docker and is available in the Preview Program

Atom now works via Rosetta 2