WebAssembly and the Wizards of Hogwarts

Daniel Burger

7. November 2020

Abstract

There is a new kind in the web development hood: WebAssembly. It is fast, portable, supported by the big players and should allow making the world wide web the biggest software platform in existence. It doesn't matter if you are an experienced software architect or a front-end developer — everyone will be able to profit from WebAssembly's existence.

In this article, we will look at what WebAssembly is, how it works and how it works alongside its sibling JavaScript.

Contents

1 Introduction **4**

2 Web Development Back Then **4**

3 JavaScript's Destiny **5**

1 Introduction

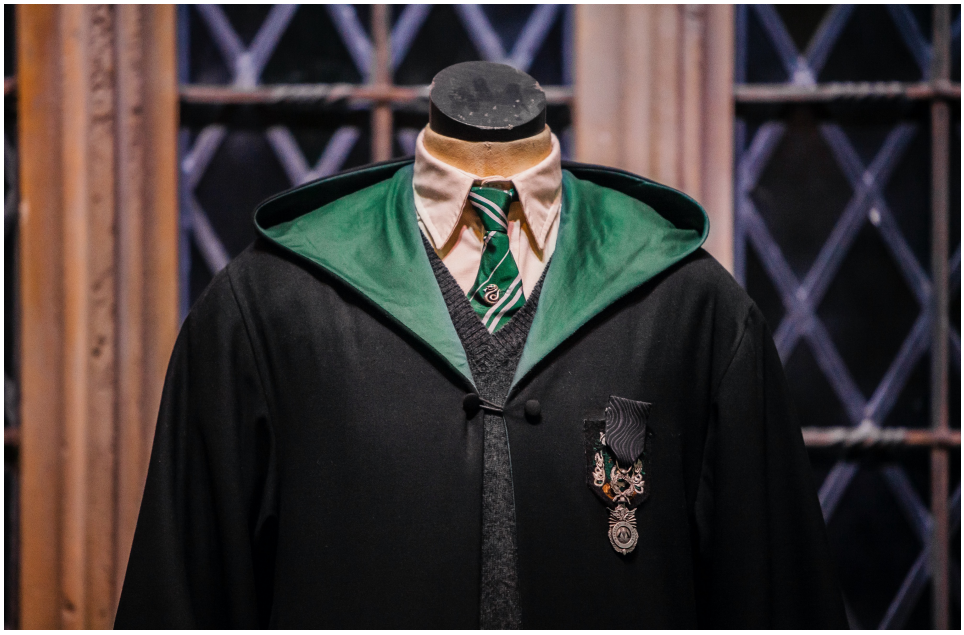

Figure 1: Harry Potter dress form with a black jacket (Unsplash, 2018)

On the 17th of June 2015, Brendan Eich — the inventor of JavaScript — and the teams behind Mozilla, Chromium, Edge and WebKit presented a new browser standard: WebAssembly, a portable and highly efficient byte-code compilation target for high-level languages such as C++ and Rust (Eich, 2015).

However, what does this mean? What is the reason that WebAssembly should exist in the first place? Should JavaScript developers be worried now? And what do the wizards of Hogwarts have to do with it?

2 Web Development Back Then

Back in the day when you could call yourself web developer because you only understood HTML, web development itself was a rather interactionless and static field of business. Netscape Communications, one of the establishing companies who built the world wide web as we know it today, soon realised that websites lacked to be interactive and dynamic (Cassel, 2018). They wanted the web to be a new form of a distributed operating system rather

than just a simple HTML document-accessing application on your computer.

Marc Andreessen, who was the founder of Netscape Communications, proposed that HTML needed some kind of "scripting language" that was approachable — something you could glue directly into the markup. A language that is easy to use by newbie programmers who didn't want to handle compiler errors or strictly-typed syntax.

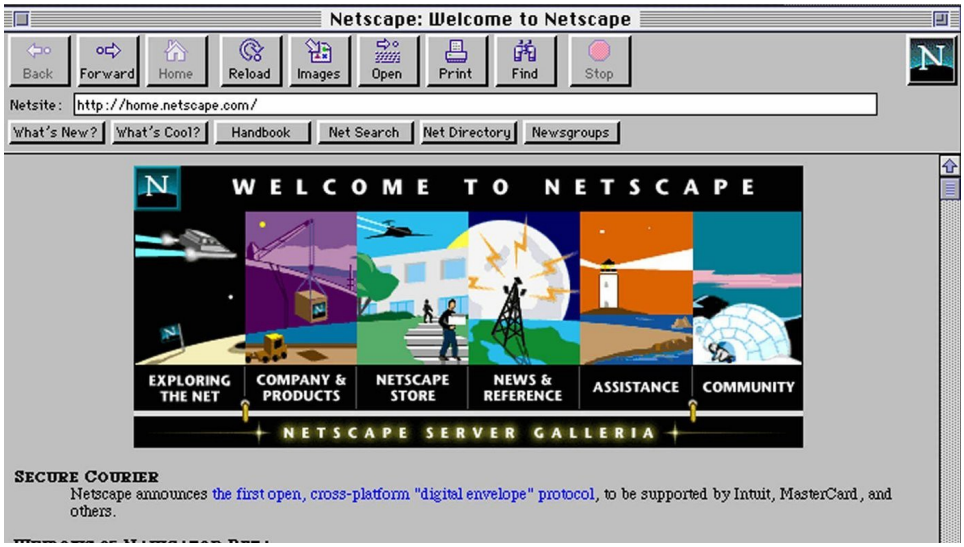

Figure 2: Screenshot of the Netscape start-page (NPR)

That was the reason they hired the experienced programming language and network code developer Brendan Eich. Brendan's first task was the nearly unachievable goal of creating such a scripting language for the web — due in 10 days. They (later) called it: JavaScript (Severance, 2012).

3 JavaScript's Destiny

I don't need to dig too deep into the history of the web to show you one crucial pain point of today's web technology standards: JavaScript is a scripting language for the browser to interpret. I repeat it: JavaScript is a scripting language for the browser to interpret — not some fancy multi-paradigm system programming language that focuses on speed, security or code maintainability. It was supposed and designed to be an easy-to-understand dynamically-typed scripting language to give your website some cool DOM manipulations and decorative animations.

Nevertheless, see what happened:

> *Any application that can be written in JavaScript, will eventually be written in JavaScript. – Jeff Atwood*

Bibliography

NPR (n.d.). Home Page Top Stories.
URL https://www.npr.org (Accessed at: 2022-11-19)

Unsplash (2018). Beautiful Free Images & Pictures | Unsplash.
URL https://unsplash.com (Accessed at: 2022-11-19)