Why Browser Engines ≠ Real Desktop Browsers ≠ Mobile Browsers

David Burns - @AutomatedTester

Agenda

- Specifications…
- Levels of ‘real-user’ testing
- Headless vs headful testing: Technicalities and examples
- Browser engines vs real browsers: Technicalities and examples
- Real mobile browsers vs simulated browsers: Technicalities and examples
- Q&A

Specifications

RFC2119

Status of this Memo

This document specifies an Internet Best Current Practices for the Internet Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Abstract

In many standards track documents several words are used to signify the requirements in the specification. These words are often capitalized. This document defines these words as they should be interpreted in IETF documents. Authors who follow these guidelines should incorporate this phrase near the beginning of their document:

> The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Note that the force of these words is modified by the requirement level of the document in which they are used.

1. **MUST** This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

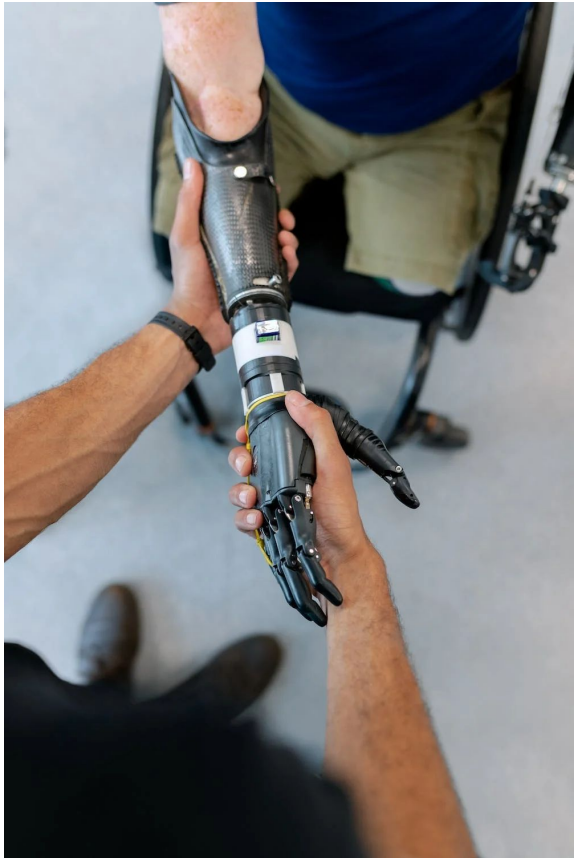
2. **MUST NOT** This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.

3. **SHOULD** This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

4. **SHOULD NOT** This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

Real User Testing

Real User Testing



https://www.pexels.com/photo/person-holding-prosthetic-arm-3912979/

https://www.pexels.com/photo/1-1-3-text-on-black-chalkboard-374918/

Why does this matter?

Headless vs headful testing

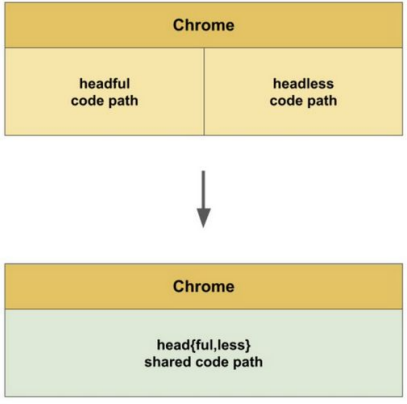
Mathias Bynens @mathias · Feb 22

Normal 2%

Did you know that “old” Chrome Headless (2017–2023) was a separate, alternate browser implementation that just happened to be shipped as part of the same Chrome binary? 🤯 It doesn’t share any of the Chrome browser code in //chrome!

Say hello to --headless=new in Chrome 112!

> **Chrome Developers** @ChromiumDev · Feb 22
>
> Normal 0%
>
> Chrome’s Headless mode just got a whole lot better! We made Headless more useful for developers by bringing it closer to Chrome’s regular “headful” mode.
>
> 👉 developer.chrome.com/articles/new-h...



17 131 621 205.7K

Browser engines vs real browsers


Safari Technology Preview

Bad Apple Safari update breaks IndexedDB JavaScript API, upsets web apps

Developers fed up with iGiant neglecting non-native software

Thomas Claburn

Wed 16 Jun 2021 // 07:29 UTC

Apple's WebKit team has managed to break the popular IndexedDB JavaScript API in the latest version of Safari (14.1.1) on macOS 11.4 and iOS 14.6.

The bug, first reported on June 2, 2021, only manifests when applications first try to use IndexedDB NoSQL manager to store data. Reloading a web page or app implementing the API resolves the issue, according to several bug reports.

Nonetheless, the situation is less than ideal for web developers and for anyone using the desktop or mobile versions of Safari. While there are a variety of storage APIs available to web developers, IndexedDB is one of two (the other being the Cache Storage API) that's recommended; the other options have specific use cases, shortcomings, or aren't widely supported.

Feross Aboukhadijeh, an open-source developer who runs Socket, on Monday said that the bug prevented his firm's web-based file transfer app Wormhole from working when initially loaded until a workaround was implemented.

"Opening an IndexedDB database fails 100 per cent of the time on the first try," he said via Twitter. "If you refresh, it starts working."

[BUG] drawImage doesn't work with WebM files #20489

Open **warrenseine** opened this issue on Jan 30 · 2 comments

warrenseine commented on Jan 30

Context:

- Playwright Version: 1.31.0-alpha-jan-29-2023
- Operating System: macOS 13.2
- Node.js version: 16.17
- Browser: WebKit 1783

Code Snippet

See repo: https://github.com/warrenseine/playwright-webm-test

Describe the bug

Following the resolution of #18423, I wanted to test my use case of copying WebM frames to a canvas, using `Canvas.drawImage()`. Unfortunately, it won't work with WebM video under the most recent Playwright WebKit version. It works fine with Safari, Playwright Chromium, Playwright Firefox.

Here is a demo of the bug that you can reproduce in WebKit: https://warrenseine.github.io/playwright-webm-test/

While playback now works fine with WebKit, you can see that drawing frames to a canvas from an MP4 video works, but it doesn't with WebM.

[BUG] SharedArrayBuffer should work in WebKit #14043

Open yury-s opened this issue on May 9, 2022 · 4 comments

yury-s commented on May 9, 2022 Member

This popped up in #13976, SharedArrayBuffer was reenabled in Safari https://webkit.org/blog/12140/new-webkit-features-in-safari-15-2/ but we disabled COOP handling in #9185 as it broke some other functionality in WebKit. We should reenable COOP to match Safari and make SharedArrayBuffer work.

2

Real mobile browsers vs simulated browsers

Real mobile browsers vs simulated browsers

- Covid-19 saw the boom of mobile usage across the globe
- Getting all the mobile devices is expensive
- Mobile development is hard... but thanks to browser devtools it’s not so hard

Real mobile browsers vs simulated browsers

- Mobile development is hard... but thanks to browser devtools it’s not so hard

OR IS IT...

Real mobile browsers vs simulated browsers

- Resizing a desktop browser is NOT a mobile...

[BUG] reports the wrong devicePixelRatio #20111

Open **greggman** opened this issue on Jan 13 · 4 comments

greggman commented on Jan 13

Context:

- Playwright Version: 1.29.2
- Operating System: Mac
- Node.js version: 16.17.0
- Browser: Chrome
- Extra: Make sure you're on DPR > 1 device This is true for most Macs but might not be if you're using an external monitor

Code Snippet

follow the docs for a setup with `npm init playwright@latest`

in `playwright.config.js` add `use: { headless: true, ...`

replace `example.spec.js` with

```
// @ts-check
const { test, expect } = require('@playwright/test');

test('test dpr', async ({ page }) => {
  await page.goto('data:text/html,<div id="d" style="display: inline-block;">foo</div>');
  await page.waitForLoadState('networkidle');
```

Risk Management

Q&A

Thank you!

David Burns - @AutomatedTester