Date Published: 22/01/2023

Version: 1.0

The Carbon Impact of Web Standards

A systematic investigation into the carbon impact of Web Standards such as front-end HTML and CSS; which have a cumulative impact upon how high the emissions of a website, service, or web application are.

Author:

Alexander Dawson, alex@hitechy.com, https://alexanderdawson.com

Thanks:

Sustainable Web Design Community Group, W3C.

Table of Contents

Abstract

When analyzing carbon emissions in digital data, current best-guess methodologies involve a kWh per GB analysis [1]. What this fails to take into account is the diversity of emissions generated between differing Web standards and the variance of individual components within them [2]. This study systematically examines the energy cost of front-end HTML, CSS, Media Formats, and a host of other (in-use) specifications to help Web developers best determine how to optimize their code for eco-efficiency. This study uses a series of test suites to analyze the variables CPU (MS), GPU (MS), RAM (MB) and physical data transferred (bytes) to get a realistic "cost" impact value for each feature within each measured specification. Variables were chosen as each impacts carbon output irrespective of it's environment [3]. This research aims to make recommendations while adhering to best practices such as progressive enhancement [4], and Web accessibility [5]. Results from the study showing high carbon output for a particular HTML or CSS feature have been earmarked as bad practice (only if it can be avoided or prevented), or flagged to respective browser vendors to improve upon efficiency (if this can be realistically achieved) [6]. However, no language is free from a carbon impact as data is transferred and rendered. Therefore, to emit less carbon, developers must code carefully and responsibly.

Abbreviations

CSS	Cascading StyleSheets
CPU	Central Processing Unit
GB	Gigabyte
GPU	Graphics Processing Unit
HTML	Hypertext Markup Language
JS	JavaScript
KB	Kilobyte
kWa	Kilowatts Per Hour
MB	Megabyte
MS	Millisecond
RAM	Random Access Memory
W3C	World Wide Web Consortium
W	Watt
WASM	Web Assembly

Introduction

The Internet is a world leader in carbon pollution **[7].** With an ever increasing awareness of the importance of sustainable web design **[8],** it's critical that web developers and designers craft new strategies to deal with the growing climate emergency (and in doing so, improve the user-experience for visitors to their services).

The benefits of sustainable web design and development for consumers (and business owners) are multi-fold:

- Reducing the carbon footprint by way of performance improvements will reduce bandwidth costs for those who host websites (leading to fewer overhead costs in maintaining a website or app) **[9]**. There is an added benefit to visitors as with fewer bytes being sent "over-the-wire", less time is required to wait in order to interact with pages on the first load; leading to potential higher conversion rates **[10]**. With pages loading faster added benefits are that as content is more likely to work with older devices, a potential wider market of visitors able (and willing) to engage with the services is being offered **[11]**.
- Of course, speed improvements aren't the only way to make a website sustainable. Having a project hosted on a sustainable (green) host may come at a potential equal or slightly higher price tag, but for an ethical business, it could gain additional customers who choose them over a rival for this reason **[12]**.
- Additionally, taking advantage of innovations like print stylesheets **[13]**, making a service accessible (to avoid wasted time and sessions) **[14]**, ensuring source code is semantic **[15]**, and (of course) having a great user-experience to reduce "stuck-sessions" where the visitor is trapped with choice (or attention paralysis) all help **[16]**. Not to mention ensuring your site is accessible to avoid potential lawsuits and reduce wasted bandwidth from frustrated users unable to complete actions (plus it widens the reach of your platform). It's great for visitors (and for businesses), it makes for few complaints, opens new markets and reduces the chances of visitors giving up and going elsewhere due to errors or focus loss.

Naturally any of these areas would be interesting fields of study to examine how much carbon they produce, and the overall impact they have in relation to the rest of the Web; however, as my formal education is in code (being that I'm a Web Developer), I'm going to examine the markup languages of the Web. The benefit of studying Web standards such as HTML and CSS is that aside from little being known about the impact they have on the overall carbon output of a website, the results of analyzing particular variables will offer explicit quantitative results (which can be put into action by stakeholders). When possible in the discussion of the results I shall provide qualitative feedback to further enhance the researches usefulness using evidence based analysis of the specifications, third party research and my own experience of performance, accessibility and Web standards guiding how this may tie into the carbon efficiency of a website during the rendering process.

Scope of the Problem

It's been stated by Greenpeace in a recent report that if the Internet were a nation, it would be one of the top ten carbon emitting countries on earth **[17]**. While I'm generally not in favor of using metaphors to illustrate a point, this one does evoke a striking image of the scale of dependency the Web has upon electricity. From mining cryptocurrencies **[18]**, to channeling emails (most of which are spam **[19]**), from streaming video and audio **[20]**, to rendering websites and apps **[21]**, from gaming **[22]**, to firing data between Internet connected devices, it all uses energy **[23]**. Not just device usage, but the creation of tools and products themselves **[24]**.

Developers and designers haven't been entirely on the wrong side of the fight against climate change, from innovations such as caching, CDN's, compression algorithms (which constantly improve), better image and video and audio formats, linting, tree shaking, accessibility and better web browsers; web workers have made websites and applications faster and more efficient (leading to less energy being required to power them). We even have future scope for improvements with HTTP3 and (wider) Web Assembly uptake on the horizon.

Unfortunately, despite all of the improvements being made, page sizes continue to increase year-on-year **[25]** as visitors expect more from websites than ever before. Twenty years ago, images were the only things being requested, fifteen years ago it was audio files, ten years ago it was standard definition video and now visitors expect either high definition or 4-8K ultra high definition footage is served. Granted designers shouldn't refuse them such improvements in quality, but questions should be asked whether it's necessary in every situation.

Because the situation is such, I decided to find ways in which energy use could be measured. As it currently stands there are a few tools on the market which attempt to examine how much energy a website uses [26]. The main issue with these tools are that they only examine a single page (upon load), and calculate a figure based upon the total page size (adjusted for possible caching scenarios) against an energy use profile [27].

The limitations of this approach are as follows:

- Examining a single page fails to take into account the total scope of a website.
- Calculating against the page size alone misses important variables such as more energy dependent code (consider for example how CPU intensive CSS animations can be, or GPU intensive canvas is).
- Adjusting for caching alone doesn't account for additional byte saving measures (for examine GZIP) which have their own carbon footprint ratio to consider.

While this research paper does not aim to resolve all of these unanswered questions, it will aim to examine more closely the second limitation. This is because during this research I was unable to find any prior study or examination into the carbon impact of HTML, CSS or other semantic features of the Web. This is an important field of study to examine because while much is known of the semantic, accessibility, and performance implications of using HTML elements or CSS properties [28]; little to nothing is known about the sustainability of using such elements. Such knowledge would have wide implications for tooling (being able to realistically calculate a websites green credentials). Also it would help potential carbon traps be found in specifications which should empower browser vendors to fix flawed implementations (thereby saving energy for visitors).

Existing Literature

Within the subject of sustainability, there is little to no literature covering web syntax and the implications of using specific elements, attributes, properties, selectors, values, media queries, etc. Despite the lack of existing research, it's known that adding content to a page adds to the overall bloat; therefore, performance of a page will take a hit. With performance being affected, the load on the visitors computer will increase as will power use in order to render the websites content to the visitors display. Web performance literature may shed light on potential energy savings [29-31], holding additional insight into features that may prove carbon costly [32].

Plan of Investigation

Firstly I plan on creating an HTML boilerplate document to calculate the carbon footprint of getting a minimal page (with just mandatory tags) to the screen. Once this has been accomplished I'll run through every HTML element and attribute calculating it's footprint (minus the boilerplate). With CSS my boilerplate will include all the HTML required to run any CSS test successfully (and run the tests as with the HTML). Finally I'll approach testing other formats such as media types and other web assets. In doing so I'll build up an inventory of the cost (approximate) each element will factor into a document. This can be converted into a formula which can be used to calculate other sites sustainability credentials with higher accuracy than using existing techniques (based simply on page weight alone), which could prove beneficial to others. It should also be noted that I will consider sustainability benefits which cannot be directly calculated (as it's considered qualitative data) and there will be variables I shall not account for (such as the complexities of JavaScript or embedded programs such as compiled WASM). They will require further study - but I'll make some evidence based (performance) recommendations in the discussion about such features to attempt to be feature complete in my analysis.

Hypothesis

I can realistically predict (based on extrapolated existing performance metrics) that certain HTML elements will require more energy (carbon) in order to render at the client-side than others. For example, it's not out of the question to expect a paragraph tag or heading tag to require less energy than an image or video element. In addition, it's more than likely that certain aspects of CSS will impact carbon emissions more than others (such as print stylesheets or dark mode). As such I should be able to identify recommendations in relation to where sustainability improvements can be made within a website (in relation to quantity and quality of semantics). Additionally certain features will require certain hardware more than others. Examples of such differences may include the canvas element requiring GPU cycles, whereas the script element will require CPU cycles instead.

Methods

I shall begin this study by focusing my research upon the web standards which I feel will most benefit from immediate analysis (and it will be cost-effective to examine quickly). Because of it's simplistic nature HTML is an obvious choice as it offers developers a range of different "render-to-screen" elements with attributes that can affect the layout and performance of the tag **[33]**. I shall also prime my attention towards each of the CSS specifications as they have a lot to offer (and cause confusion within the developer community due to issues like the cascade) **[34]**. This makes examining any pitfalls in sustainability or any margins for improved usage performance worthwhile. I shall also examine different media formats such as audio and video files and image formats as they are widely used **[35]**, and some formats like JSON and XML **[36]** as they are in common use and can potentially impact a website in unusual ways. As an additional feature, I shall also study obsolete and deprecated HTML and CSS variables as these in many cases not only have active browser support but they also may have individuals using them (against advice) on live sites. This provides me with my parameters.

Creation of a Boilerplate

Each language has it's own set of constants of which I would like to test against. For HTML it's elements and attributes, for CSS it's media queries, selectors, etc. In order to test them I need to accomplish several things. Firstly I need to create a boilerplate to get the base HTML or CSS rendered to the test device. This will be referred to as the boilerplate document and will contain no-more than the required content to render a test. From there I shall take a baseline measurement using my profile methodology (mentioned below) in order to eliminate the boilerplates numbers from any further analysis. Then, whenever I add code into a test (based upon the boilerplate), any added energy requirements can only be explained as a result of the new variable.

If there is an instance where a variable requires itself to be placed within another tested variable (say an input element within a form element) then the parents measurements shall be eliminated from the results of the other elements score (in place of the boilerplate) to provide an accurate energy rating. But if the element can be purely rendered from within the boilerplate (and it won't affect it semantically), it will be done instead. Working like this, I can ensure that every HTML5 and CSS3 feature is given a fair representation of how it would function in the real world, but also precisely calculate (and record) it's total energy requirements.

Bias Elimination

This study will remove as many biases as it is possible to eliminate in order to provide a fair study. Some biases such as case sensitivity are automatically taken care of by the browser. With case sensitivity, HTML5 and CSS3 (the current living standards) are case-insensitive, not strictly typed, or parsed by rendering engines. The only time where case-sensitivity may be an issue for developers is when altering code using a strictly typed language like JavaScript, but this lies outside the scope of this study in identifying any environmental impact.

By hosting the test suite on two different providers and testing it within two different web browsers (and their respective rendering engines), it should ensure that results aren't skewed to favor a single variable. Hopefully, this will result in actionable metrics and the results can be taken as valid. Areas where bias can be introduced could include DOM code injection (this study deliberately does not examine JavaScript's ecological impact as it's expansive enough to need dedicated research of it's own), plus there could be bias found in the choice of examples used within the test-suites. I have however tried to ensure they are based on real-world, semantic, and accessible examples to ensure realism within the results (and to ensure they will render accurately).

Randomization & Variance

To maintain a double-blind analysis, the hosting providers, the variable being tested against, and also the browser being tested against will be chosen at random (during the trial from a pre-determined selection). The only thing being undertaken by the tester is the benchmark (which will be computed by the machine and use the host selected from the data) and the recording of any results. This should help to reduce biased recording and maintaining a randomized control of the data. Testing was undertaken during UK working hours to ensure that peak usage time (stress test) effects would apply to the data being transferred for real-world assurance.

To be able to accomplish this, a script has been created which generates (at random) the filename of the test that should be benchmarked. This is loaded into a headless state (so that the server nor browser is visible), from there the test is performed. Upon completion the test window is automatically closed and an alert provides the data to be entered (and where), thus ensuring prejudice cannot play a part in the process.

It should be noted that any variance in the results (for individuals who repeat these tests) could be down to future improvements in the Web browsers (and their various rendering engines), different equipment being used to test the baseline measurements (for this study a M1 MacBook Air is being utilized with 16gb RAM, and my line speed is 40mb down, 8mb up). Additionally the availability and stability of the website (latency, etc) or conditions outside of the control of this experiment (including variables not being measured against) can also influence the outcome. As such any results recorded whilst not biased by the author can be impacted.

Two-Location Hosting for Validation

For the purposes of this research, The research shall host the collection of test suites in two different locations using different host providers: Github and Netlify **[37,38]**. The justification for this is that in order to accurately measure how much rendering time a website requires, by using multiple hosts you can avoid introducing bias caused by issues exclusive to a single provider. As such, by recording data between different sources (and averaging the result) will gain a more accurate representation of real-world results. It's also worth noting that as the host data is random at recording, bias is subsequently avoided using a double-blind methodology. As an additional point, in cases where a host (most notably Github Pages) doesn't support a particular format, a third host will be introduced to fill in the gaps and provide a comparative result to avoid a singular source.

Testing Browsers for Variance of Results

Alongside hosting requirements, it's as important than ever to recognize that not all web browsers are alike in how they render content (Internet Explorer in the past **[39]**, Apple Safari in the present **[40]**) and thus it's just as important today to test measurements from within multiple rendering engines. The two I have picked for the purposes of this study are Chromium (Google Chrome v108) as it's the most popular web browser on the market **[41]**, and Mozilla Firefox v106 (due to it's unique browser engine, Gecko) **[42]**. I chose against using Safari despite it having the most variance in browser technology due to it having the worst browser tooling for measuring energy performance (making it unreliable for this study) and because it's rendering engine is too closely related to Chrome as Webkit is forked into Blink (the underlying engine of the Chromium project. It's also worth noting that different browsers have differing levels of support for HTML and CSS at the bleeding edge, which means that results may vary; though it was still worth testing against such variables as lack of support still leaves a shadow impact as the browser has to compensate and either fall back or fail the render.

Using these browsers I shall utilize a clean session for each (incognito / private mode), resourcing the browser developer tools within (the performance tab (with the memory checkbox ticked if necessary). I shall include all activity categories (EG: Loading, Painting, Rendering, Scripting) excluding any idle time. Being that not all browsers (outside Chrome) allow direct measuring of GPU usage within the developer tools, I shall use the closest alternative means to accurately record this information such as the browsers task manager or a local JS script which has been natively coded to log a single value (the GPU peak using available sensor information).

Profile Recording Methodology

For the purposes of recording I have setup a spreadsheet which includes four primary variables I wish to test as a measurement of energy efficiency. The first of these is CPU usage. The justification of this is that when the processor is pushed hard performing math calculations and rendering content or crunching through a heavy JavaScript document, it drains the battery more freely **[43]**. The second variable which I'm going to test against within this study is GPU usage. This more than CPU usage can have a dramatic impact upon battery life as any gamer will tell you; performing heavy rendering tasks involving graphics, video or related media on-screen requires a lot of hardware interaction and thus consumes a great deal of energy **[44]**. The third factor I'll account for in this study is RAM usage. The reason for this is that if a website is particularly bloated or uses a lot of system resources it (again) can lead to high energy use, high network use (due to page refreshes from loss of cache), and potential memory leaks which can be a sign of the site being unoptimized (and a battery drainer) **[45]**. The final variable I will account for in my study are the physical bytes transferred to the visitors machine. Just like with RAM usage, page bloat is an important factor in detecting latency, cache, and other sustainability issues **[46]**. In addition, notes will be made about special conditions that exist which make a certain test case more sustainable or potentially give it a higher carbon footprint (needing direct attention).

The equation to calculate a profile is as follows:

$$v = \left(\; S1(C+F) + S2(C+F)\right)$$

$$profile = \left(\frac{CPU(v)[ms]}{4} + \frac{GPU(v)[ms]}{4} + \frac{RAM(v)[mb]}{4} + \frac{Data(v)[kb]}{4}\right)$$

S1 & S2 refer to the host Github and Netlify. C and F refer to the browser Chromium and Firefox.

To get an average for each of the variables (CPU, GPU, RAM and Data), each of the testing guidelines (the two hosts and the two browsers) are checked and their values measured. Once this has occurred their results are added together and divided by four to get a single figure which presents an overall average for the page. This means that if a page performs poorly in one browser but scores well in another and on both hosts it will not be punished for a potential error or faulty recording. Equally, if it scores poorly on three out of the four testing pathways (say both hosts and a browser) it will indicate a real issue with it's energy management policy.

To turn a profile (score) into kWh, both CPU and GPU time must be turned into watts. As the average CPU **[47,48]** (mobile and desktop) can use between 7w and 125w, just take the median value of 66 as the average to make allowances for transitioning from old to new tech. With GPU's the rule is the same aside from the average GPU using between 25w and 350w **[47,49]** as GPU's render process heavy games. The median of this value is therefore 185 (if averaged up). Then divide those two numbers by milliseconds per hour to get w/ms. If the CPU and GPU variables (combined) are multiplied by our result the energy use in watts will be known.

For every 8GB of RAM, 3W will be used **[47,50]**. So in order to get it down to a "watt per mb" value, just divide 8GB by the 3 (reducing it to per watt) and then by 1000 (thereby cycling it to per mb) leaving the value of RAM energy use in w/mb. Finally, just multiply the baseline by the RAM variable to get the memory usage in watts.

Turning the watts into kWh is simply a matter of taking the watt calculations (above), multiplying them by the energy used (xPU and RAM variables), and dividing the value by 1,000. Because many hardware SoC's decide to co-locate (and multitask) CPU's and GPU's, the numbers are combined into a single variable known as xPU.

Hard drives are a low power device, especially with SSD's (consuming 0.5-9w). As they are constantly in use, it's worth examining data transfer (on first-load) as a reliable method of energy consumption. For this use KB notation (bytes that are rounded to the nearest 1,000). I'll use the 0.12 kWh/GB measurement this study aims to expand upon **[51,52,53]** as a baseline measurement, so cycling the GB to KB means a 120,000 kWh/KB final figure to work with. I chose KB as the average website is less likely to be over 1GB in (total) size at this time. As this study will be examining first-load only, further examination could be done into pre-cached (warm) data.

To apply this to your own website, you can use the results table (attached to this research) to work out the cost of using a piece of the specification. Remember to get the average between servers and browsers as denoted in the above equation. For quick results get the CPU, GPU, RAM and Data for the whole page; for more accurate results, you'll want to add up all the bits of your code (and their cost as referenced in the table).

The equation to turn variables into actual energy use is as follows, it uses the previously explained calculation:

$$kWh = \left(\frac{\left(\left(\frac{66+185}{3{,}600{,}000} \times xPU\right) + \left(\frac{8 \div 3}{1{,}000} \times RAM\right)\right) \times (xPU + RAM)}{1{,}000}\right) + \left(\frac{Data}{120{,}000}\right)$$

xPU refers to CPU & GPU combined, hence why the medians for both values are used in the equation.

Results

Each of the following sections contains the conclusions of this study once all of the variables have had their individual calculations (as per the methodology) profiled. Results with any similarities will have been grouped together (for brevity) and any interesting points of note made clear. It's worth emphasizing early on that any further analysis can be achieved on the results (beyond this study) by examining the spreadsheet of research I have undertaken which contains all of the figures in full (and will accompany this document upon publication).

It's worth noting for reference purposes that this study created a suite comprised of:

- **293** HTML element & attribute tests
- **516** CSS rule, selector & property tests
- **51** Media & other specification tests

With each test running over four separate conditions, this equated to **13,760** benchmarks.

HTML Specification

First let's examine CPU usage, as that seems like a decent place to start. The below table arranges which of the elements and attributes (on average) ended up with the best CPU score in MS. If the variable took longer than 7.5 ms then it's a bit more sluggish than average (but not noticeably), however a rare few took 10ms+ which highlight the disparity between performance of various semantic features (take it into consideration if using).

MS	Elements	Attributes
10+	embed, label, noembed	autocomplete, data-*
7.5-9	a, abbr, address, article, b, bdo, big, br, button, center, cite, code, colgroup, dd, details, dfn, dialog, div, em, font, form, header, hgroup, input checkbox, input color, input date, input email, input file, input radio, input reset, input submit, input tel, input text, input time, input url, input week, input, ins, kbd, keygen, legend, li, main, mark, marquee, math, menu, menuitem, meter, nav, noscript, object, optgroup, option, param, portal, progress, q, rt, s, select, slot, small, span, strike, style, sub, summary, table, td, textarea, tfoot, thead, time, xmp	accept, bgcolor, challenge, cite, cols, colspan, datetime, default, dir, disabled, form, formenctype, high, id, inputmode, integrity, label, language, loading, loop, low, maxlength, media, min, minlength, novalidate, pattern, ping, readonly, scope, size, srcdoc, summary, tabindex, target, translate, wrap
5-7.4	acronym, applet, aside, audio, base, bdi, bgsound, blockquote, canvas, caption, command, content, data, datalist, del, dt, fieldset, footer, hr, i, input button, input datetime-local, input hidden, input month, input number, input password, input range, input search, link, meta, ol, output, p, plaintext, pre, rb, rbc, rp, rtc, ruby, samp, script src, script, section, shadow, spacer, sup, tbody, th, tr, track, tt, u, ul, var, wbr	accept-charset, accesskey, action, autocapitalize, autoplay, capture, checked, class, codebase, contenteditable, contextmenu, controls, data, dirname, draggable, enctype, enterkeyhint, formaction, formmethod, formnovalidate, headers, hidden, href, hreflang, http-equiv, icon, itemscope, itemtype, keytype, kind, lang, list, manifest, max, multiple, name, open, optimum, radiogroup, required, reversed, role, rows, rowspan, scoped, spellcheck, src, srclang, start, step, style, title, type, value
<5	area, blink, col, dir, dl, figcaption, figure, frame, frameset, h1, h2, h3, h4, h5, h6, iframe, image, img, input image, map, nobr, noframes, picture, source, strong, svg, template, video	align, allow, alt, async, autofocus, background, border, charset, code, color, content, coords, crossorigin, csp, decoding, defer, download, for, formtarget, height, importance, intrinsicsize, ismap, itemprop, method, muted, placeholder, poster, preload, referrerpolicy, rel, sandbox, selected, shape, sizes, slot, span, srcset, usemap, width

Next let's examine the GPU results. The testing methodology was very similar to CPU so it's interesting to see how differently some elements performed than others when measuring their graphics usage when rendering. Anything up to 7.5 ms should be considered fast. Anything over 10 ms is a bit sluggish but there are a few that take over 20 ms which could be a sign that the element (or attribute) is part of a render blocking chain.

MS	Elements	Attributes
20+	embed, iframe, noembed	autoplay, csp, sandbox
10-19	area, audio, bdo, canvas, center, cite, col, colgroup, del, dfn, dir, em, figcaption, figure, font, image, main, map, meta, nav, noscript, param, picture, plaintext, portal, pre, s, samp, script src, script, shadow, source, strike, style, summary, td, tfoot, thead, time, tt, u, wbr, xmp	allow, charset, cite, class, cols, colspan, contenteditable, coords, data, importance, kind, loop, manifest, muted, poster, preload, srclang
7.5-9	a, acronym, b, bgsound, big, br, code, dd, details, frameset, hgroup, i, input file, input range, input reset, input submit, input time, input week, ins, kbd, label, link, mark, marquee, math, nobr, object, ol, option, rt, ruby, section, select, slot, small, sub, sup, tr, track, ul	alt, border, crossorigin, defer, enterkeyhint, formenctype, formnovalidate, headers, height, high, hreflang, http-equiv, id, integrity, intrinsicsize, language, low, maxlength, min, novalidate, pattern, scope, sizes, srcset, summary, target, translate, usemap, wrap
5-7.4	abbr, address, applet, article, aside, base, bdi, blink, blockquote, button, content, datalist, dialog, div, footer, form, header, hr, img, input button, input checkbox, input color, input date, input datetime-local, input email, input hidden, input month, input password, input radio, input search, input tel, input text, input url, input, keygen, legend, menu, menuitem, meter, optgroup, output, p, progress, q, rbc, rp, rtc, span, strong, svg, table, tbody, textarea, th, var, video	accept, accept-charset, accesskey, action, align, async, autocapitalize, autofocus, capture, challenge, controls, data-*, datetime, decoding, default, dir, disabled, download, draggable, form, formmethod, formtarget, hidden, inputmode, ismap, keytype, label, lang, loading, max, media, minlength, multiple, optimum, ping, placeholder, radiogroup, readonly, referrerpolicy, required, reversed, rows, selected, shape, size, spellcheck, src, title, type, value, width
<5	caption, command, data, dl, dt, fieldset, frame, h1, h2, h3, h4, h5, h6, input image, input number, li, noframes, rb, spacer, template	autocomplete, background, bgcolor, checked, code, codebase, color, content, contextmenu, dirname, enctype, for, formaction, href, icon, itemprop, itemscope, itemtype, list, method, name, open, rel, role, rowspan, scoped, slot, span, srcdoc, start, step, style, tabindex

Next we have RAM usage. As you can tell from the results below, most of the elements and attributes are very good at not utilizing additional memory beyond the physical rendering of the object (0.1 MB overhead from the boilerplate measurement). But there are a few elements that do need a bit of extra memory due to having complex components, heavy content, or data which processes heavily on the hardware (graphics or audio).

MB	Elements	Attributes
1+	embed, noembed	data, preload
0.2-0.9	applet, audio, datalist, iframe, input date, input datetime-local, input email, input month, input password, input text, input time, input week, input, link, marquee, object, option, track, video	allow, autocapitalize, autofocus, autoplay, capture, contenteditable, controls, csp, default, kind, label, list, muted, pattern, poster, required, sandbox, selected, size, srcdoc, srclang, step, value
0.1	a, abbr, acronym, address, area, article, aside, b, base, bdi, bdo, bgsound, big, blink, blockquote, br, button, canvas, caption, center, cite, code, col, colgroup, command, content, data, dd, del, details, dfn, dialog, dir, div, dl, dt, em, fieldset, figcaption, figure, font, footer, form, frame, frameset, h1, h2, h3, h4, h5, h6, header, hgroup, hr, i, image, img, input button, input checkbox, input color, input file, input hidden, input image, input number, input radio, input range, input reset, input search, input submit, input tel, input url, ins, kbd, keygen, label, legend, li, main, map, mark, math, menu, menuitem, meta, meter, nav, nobr, noframes, noscript, ol, optgroup, output, p, param, picture, plaintext, portal, pre, progress, q, rb, rbc, rp, rt, rtc, ruby, s, samp, script src, script, section, select, shadow, slot, small, source, spacer, span, strike, strong, style, sub, summary, sup, svg, table, tbody, td, template, textarea, tfoot, th, thead, time, tr, tt, u, ul, var, wbr, xmp	accept, accept-charset, accesskey, action, align, alt, async, autocomplete, background, bgcolor, border, challenge, charset, checked, cite, class, code, codebase, color, cols, colspan, content, contextmenu, coords, crossorigin, data-*, datetime, decoding, defer, dir, dirname, disabled, download, draggable, enctype, enterkeyhint, for, form, formaction, formenctype, formmethod, formnovalidate, formtarget, headers, height, hidden, high, href, hreflang, http-equiv, icon, id, importance, inputmode, integrity, intrinsicsize, ismap, itemprop, itemscope, itemtype, keytype, lang, language, loading, loop, low, manifest, max, maxlength, media, method, min, minlength, multiple, name, novalidate, open, optimum, ping, placeholder, radiogroup, readonly, referrerpolicy, rel, reversed, role, rows, rowspan, scope, scoped, shape, sizes, slot, span, spellcheck, src, srcset, start, style, summary, tabindex, target, title, translate, type, usemap, width, wrap

Next we have the results from the physical data analysis, which notably excludes the HTML boilerplates KB usage so is a pure rendition of the space taken on the disk (and in transfer) to cache and render an element or attribute. As with RAM, a 0.1 result means a very low overhead as less than 100 bytes is required. Up to 0.5 means half a KB is required for that object, and any higher will add to that loading time. It should be reminded that in some nations, very slow internet speeds still exist and pushing pages this hard can make a difference not only in the user experience but also it reduces the energy required to transfer and render data to a screen.

KB	Elements	Attributes
0.5+	*None*	*None*
0.2-0.4	a, abbr, acronym, area, audio, b, bdo, bgsound, big, blockquote, br, canvas, center, cite, code, content, datalist, del, dfn, em, font, frame, i, iframe, input date, input datetime-local, input file, input month, input number, input password, input radio, input tel, input time, input url, input week, input, ins, kbd, mark, math, nobr, noframes, optgroup, option, output, p, plaintext, q, rtc, s, samp, script, shadow, small, source, span, strike, strong, style, sub, sup, time, track, tt, u, var, video, wbr, xmp	allow, autocapitalize, autoplay, capture, challenge, cite, codebase, controls, coords, csp, datetime, default, dir, draggable, enterkeyhint, hidden, hreflang, kind, label, lang, language, list, media, multiple, muted, pattern, ping, poster, preload, rel, required, sandbox, scoped, selected, shape, size, srcdoc, srclang, step, tabindex, target, title, translate, value
0.1	address, applet, article, base, bdi, blink, button, caption, col, colgroup, command, data, dd, details, dialog, dir, dt, embed, fieldset, figcaption, figure, frameset, image, input button, input checkbox, input color, input email, input hidden, input image, input range, input reset, input search, input submit, input text, keygen, label, legend, li, link, main, map, marquee, menuitem, meta, meter, noembed, noscript, object, param, picture, portal, pre, progress, rb, rbc, rp, rt, script src, section, select, slot, spacer, summary, svg, table, tbody, td, template, textarea, tfoot, th, thead, tr	accept, accept-charset, accesskey, action, align, alt, async, autocomplete, autofocus, background, border, charset, checked, code, cols, colspan, content, contextmenu, crossorigin, data-*, decoding, defer, dirname, disabled, download, enctype, for, form, formaction, formenctype, formmethod, formnovalidate, formtarget, headers, height, high, href, http-equiv, icon, importance, inputmode, integrity, intrinsicsize, ismap, itemprop, itemscope, itemtype, keytype, loading, loop, low, manifest, max, maxlength, method, min, minlength, name, novalidate, open, optimum, placeholder, radiogroup, readonly, referrerpolicy, reversed, role, rows, rowspan, scope, sizes, slot, span, spellcheck, srcset, start, style, summary, type, usemap, width, wrap

Finally we have the overall picture which is drawn from the score derived using the profile formula earlier in the study. This takes into account CPU (MS), GPU (MS), RAM (MB) and Data (KB) and balances them fairly to give a nuanced view of which elements and attributes are the most sustainable and which are the least.

Score	Elements	Attributes
30+	embed, noembed	autoplay
20-29	bdo, colgroup, dfn, dir, iframe, nav, noscript, param, script src, shadow, style, u, wbr	csp, data, kind, sandbox, srclang
10-19	a, abbr, acronym, address, applet, area, article, aside, audio, b, base, bdi, bgsound, big, blink, blockquote, br, button, canvas, caption, center, cite, code, col, content, data, datalist, dd, del, details, dialog, div, dt, em, fieldset, figcaption, figure, font, footer, form, frameset, header, hgroup, hr, i, image, input button, input checkbox, input color, input date, input datetime-local, input email, input file, input hidden, input image, input month, input number, input password, input radio, input range, input reset, input search, input submit, input tel, input text, input time, input url, input week, input, ins, kbd, keygen, label, legend, li, link, main, map, mark, marquee, math, menu, menuitem, meta, meter, nobr, object, ol, optgroup, option, output, p, picture, plaintext, portal, pre, progress, q, rb, rbc, rp, rt, rtc, ruby, s, samp, script, section, select, slot, small, source, span, strike, strong, sub, summary, sup, table, tbody, td, textarea, tfoot, th, thead, time, tr, track, tt, ul, var, xmp	accept, accept-charset, accesskey, action, allow, alt, autocapitalize, autocomplete, autofocus, bgcolor, border, capture, challenge, charset, cite, class, codebase, cols, colspan, contenteditable, contextmenu, controls, coords, crossorigin, data-*, datetime, decoding, default, defer, dir, disabled, download, draggable, enctype, enterkeyhint, form, formenctype, formmethod, formnovalidate, formtarget, headers, height, hidden, high, href, hreflang, http-equiv, id, importance, inputmode, integrity, intrinsicsize, ismap, keytype, label, lang, language, loading, loop, low, manifest, max, maxlength, media, method, min, minlength, multiple, muted, name, novalidate, optimum, pattern, ping, placeholder, poster, preload, radiogroup, readonly, referrerpolicy, required, reversed, role, rows, scope, scoped, selected, shape, size, sizes, spellcheck, src, srcdoc, srcset, step, summary, tabindex, target, title, translate, type, usemap, value, width, wrap
<10	command, dl, frame, h1, h2, h3, h4, h5, h6, img, noframes, spacer, svg, template, video	align, async, background, checked, code, color, content, dirname, for, formaction, icon, itemprop, itemscope, itemtype, list, open, rel, rowspan, slot, span, start, style

CSS Specifications

First lets examine the CPU results. With the CSS specification the results have been triggered slightly differently. Because the HTML document for every test is identical (containing a variety of HTML elements of which every test can be performed), it will have an initial rendering impact (though this will be offset by the boilerplate which uses the same document in order to calculate the variables before any additional code is applied). As such, the same rules apply as with the HTML tests and the expected results should be fairly consistent.

MS	Selectors	Properties
10+	@container, @font-palette-values, @keyframes	animation, contain-intrinsic-height, contain-intrinsic-width, offset
7.5-9	:empty, :first, :last-of-type, :optional, @font-face	border-block-start, border-left-width, border-width, break-after, break-before, break-inside, font-feature-settings, font-variant-east-asian, hyphenate-character, list-style, max-inline-size, min-height, overflow-x, scale, scrollbar-color, top
5-7.4	::before, ::cue-region, ::file-selector-button, ::first-letter, ::first-line, ::grammar-error, ::part, ::placeholder, ::selection, ::slotted, ::spelling-error, ::target-text, :active, :any-link, :autofill, :blank, :checked, :default, :defined, :dir, :disabled, :enabled, :first-child, :first-of-type, :focus-visible, :focus-within, :fullscreen, :host, :host-context, :host-function, :hover, :in-range, :indeterminate, :invalid, :is, :lang, :last-child, :left, :link, :modal, :nth-child, :nth-last-child, :nth-last-of-type, :only-child, :only-of-type, :out-of-range, :paused, :picture-in-picture, :playing, :read-only, :read-write, :required, :right, :root, :scope, :target, :user-invalid, :user-valid, :where, @color-profile, @counter-style, @font-feature-values, @layer, @media, @property, @scroll-timeline, @supports, @viewport, Attribute selectors, Child combinator, Class selectors, Column combinator, Descendant combinator, General sibling combinator, ID selectors, Selector list, Type selectors, Universal selectors	-webkit-text-stroke, -webkit-text-stroke-color, -webkit-text-stroke-width, align-content, align-items, align-tracks, all, animation-composition, animation-delay, animation-direction, animation-duration, animation-fill-mode, animation-iteration-count, animation-name, animation-play-state, animation-timeline, animation-timing-function, appearance, background, background-attachment, background-blend-mode, background-color, background-image, background-origin, background-position, background-position-x, background-repeat, background-size, border, border-block, border-block-color, border-block-end, border-block-end-color, border-block-end-style, border-block-end-width, border-block-start-color, border-block-start-style, border-block-start-width, border-block-width, border-bottom-color, border-bottom-left-radius, border-bottom-right-radius, border-end-end-radius, border-end-start-radius, border-image, border-inline-end, border-inline-end-color, border-inline-end-style, border-inline-end-width, border-inline-width, border-left, border-left-color, border-left-style, border-radius, border-right, border-right-style, border-right-width, border-spacing, border-start-end-radius, border-start-start-radius, border-style, border-top, border-top-color, border-top-left-radius, border-top-right-radius, border-top-style, border-top-width, bottom, box-decoration-break, box-shadow, box-sizing, caret-color, column-fill, column-span, column-width, contain-intrinsic-size, container, container-name, container-type, flex-basis, font-family, font-language-override, font-optical-sizing, font-size, font-size-adjust, font-stretch, font-style, font-synthesis, font-variant, font-variant-alternates, font-variant-caps, font-variant-emoji, font-variant-ligatures, font-variant-numeric, font-variant-position, font-variation-settings, font-weight, forced-color-adjust, grid-area, grid-auto-flow, grid-column-end, grid-row-start, grid-template, hyphens, hyphenate-limit-chars, image-orientation, initial-letter, inset, inset-block-end, inset-block-start, inset-inline, inset-inline-end, inset-inline-start, isolation, justify-items, justify-self, line-break, list-style-image, list-style-type, margin-block-start, margin-top, margin-trim, mask-border, mask-border-mode, mask-border-outset, mask-border-repeat, mask-border-slice, mask-border-source, mask-border-width, mask-clip, mask-composite, mask-image, mask-mode, mask-origin, mask-position, mask-repeat, mask-size, mask-type, masonry-auto-flow, math-depth, math-shift, math-style, max-height, max-width, min-block-size, min-inline-size, min-width, mix-blend-mode, object-fit, object-position, offset-anchor, opacity, order, outline-color, outline-style, outline-width, overflow-clip-margin, overscroll-behavior, overscroll-behavior-block, overscroll-behavior-inline, overscroll-behavior-x, overscroll-behavior-y, padding-block-end, padding-bottom, padding-right, padding-top, page-break-after, page-break-inside, perspective-origin, place-content, place-items, place-self, pointer-events, rotate, ruby-align, ruby-position, scroll-behavior, scroll-margin, scroll-margin-block, scroll-margin-block-end, scroll-margin-block-start, scroll-margin-bottom, scroll-margin-inline, scroll-margin-inline-end, scroll-margin-inline-start, scroll-margin-left, scroll-margin-right, scroll-margin-top, scroll-padding, scroll-padding-block, scroll-padding-block-end, scroll-padding-block-start, scroll-padding-bottom, scroll-padding-inline, scroll-padding-inline-start, scroll-padding-left, scroll-padding-right, scroll-padding-top, scroll-snap-align, scroll-snap-stop, scroll-snap-type, scrollbar-gutter, scrollbar-width, shape-margin, shape-outside, tab-size, table-layout, text-align, text-align-last, text-combine-upright, text-decoration, text-decoration-color, text-decoration-line, text-decoration-skip, text-decoration-skip-ink, text-decoration-style, text-decoration-thickness, text-emphasis, text-emphasis-color, text-emphasis-position, text-emphasis-style, text-indent, text-justify, text-orientation, text-overflow, text-rendering, text-shadow, text-size-adjust, text-transform, text-underline-offset, text-underline-position, touch-action, transform, transform-box, transform-origin, transform-style, transition, transition-delay, transition-duration, transition-property, transition-timing-function, translate, user-select, visibility, white-space, widows, width, will-change, word-break, word-spacing, writing-mode, z-index

MS	Selectors	Properties
<5	::after, ::backdrop, ::cue, ::marker , :current, :focus, :future, :has, :local-link, :not, :nth-col, :nth-last-col, :nth-of-type, :past, :placeholder-shown, :target-within, :valid, :visited, @charset, @import, @namespace, @page, Adjacent sibling combinator	-webkit-line-clamp, -webkit-text-fill-color, accent-color, align-self, aspect-ratio, backdrop-filter, backface-visibility, background-clip, background-position-y, block-size, border-block-style, border-bottom, border-bottom-style, border-bottom-width, border-collapse, border-color, border-image-outset, border-image-repeat, border-image-slice, border-image-source, border-image-width, border-inline, border-inline-color, border-inline-start, border-inline-start-color, border-inline-start-style, border-inline-start-width, border-inline-style, border-right-color, caption-side, clear, clip, clip-path, color, color-scheme, column-count, column-gap, column-rule, column-rule-color, column-rule-style, column-rule-width, columns, contain, contain-intrinsic-block-size, contain-intrinsic-inline-size, content, content-visibility, counter-increment, counter-reset, counter-set, cursor, direction, display, empty-cells, filter, flex, flex-direction, flex-flow, flex-grow, flex-shrink, flex-wrap, float, font, font-display, font-kerning, gap (grid-gap), grid, grid-auto-columns, grid-auto-rows, grid-column, grid-column-start, grid-row, grid-row-end, grid-template-areas, grid-template-columns, grid-template-rows, hanging-punctuation, height, image-rendering, image-resolution, ime-mode, initial-letter-align, inline-size, inset-block, justify-content, justify-tracks, left, letter-spacing, line-height, line-height-step, list-style-position, margin, margin-block, margin-block-end, margin-bottom, margin-inline, margin-inline-end, margin-inline-start, margin-left, margin-right, mask, max-block-size, offset-distance, offset-path, offset-position, offset-rotate, orphans, outline, outline-offset, overflow, overflow-anchor, overflow-block, overflow-inline, overflow-wrap, overflow-y, padding, padding-block, padding-block-start, padding-inline, padding-inline-end, padding-inline-start, padding-left, page-break-before, paint-order, perspective, position, print-color-adjust, quotes, resize, right, row-gap, scroll-padding-inline-end, shape-image-threshold, unicode-bidi, vertical-align

Next we shall take a look at the GPU results, the results will be much alike HTML due to the calibration of the boilerplate and as such, best practices should be drawn from using the least intensive rendering objects (ms).

MS	Selectors	Properties
20+	@container, @font-palette-values, @keyframes, :current	animation, offset
10-19	::cue-region, ::grammar-error, ::spelling-error, :host, :only-child, :root, :where, @font-face, @font-feature-values, @import, @layer, @media	animation-duration, animation-play-state, border, border-inline-start-width, border-width, bottom, box-decoration-break, box-shadow, columns, font-variant-alternates, font-variant-caps, font-variant-east-asian, font-variant-ligatures, font-variant-numeric, font-variant-position, font-variation-settings, inset, inset-inline, inset-inline-end, mask-border-width, max-block-size, max-inline-size, min-block-size, mix-blend-mode, overscroll-behavior-block, perspective-origin, place-content, scroll-snap-stop, scrollbar-color, transform, transform-box, transform-style, transition, transition-delay, translate
7.5-9	::cue, ::first-line, :future, ::slotted, :active, :enabled, :focus-visible, :modal, :nth-last-of-type, :past, :read-only, :target-within, :user-valid, :valid, @color-profile, @page, @property	animation-fill-mode, animation-name, animation-timing-function, aspect-ratio, background-size, border-block, border-block-start, border-collapse, border-inline, border-inline-start, border-right, break-after, break-before, break-inside, clip, contain-intrinsic-height, content, font-optical-sizing, font-variant, font-weight, forced-color-adjust, grid-row, grid-template, height, hyphenate-limit-chars, inset-block-end, inset-block-start, inset-inline-start, line-height-step, list-style, margin, max-height, min-height, min-inline-size, min-width, object-fit, object-position, overscroll-behavior, overscroll-behavior-inline, overscroll-behavior-x, overscroll-behavior-y, ruby-position, scroll-snap-align, scroll-snap-type, scrollbar-gutter, scrollbar-width, shape-margin, text-align, text-orientation, text-overflow, text-underline-offset, text-underline-position, top, transform-origin, transition-property, visibility, widows, width, word-spacing, z-index

MS	Selectors	Properties
5-7.4	::backdrop, ::before, ::file-selector-button, ::first-letter, ::placeholder, ::selection, ::target-text, :any-link, :autofill, :checked, :default, :defined, :dir, :disabled, :first, :first-child, :focus-within, :fullscreen, :host-context, :host-function, :in-range, :indeterminate, :invalid, :is, :last-child, :last-of-type, :left, :link, :local-link, :not, :nth-child, :nth-last-col, :nth-last-child, :only-of-type, :optional, :out-of-range, :paused, :picture-in-picture, :placeholder-shown, :read-write, :right, :scope, :target, :user-invalid, :visited, @counter-style, @scroll-timeline, @supports, @viewport, Child combinator, Column combinator, General sibling combinator, Universal selectors	-webkit-text-fill-color, -webkit-text-stroke, -webkit-text-stroke-color, -webkit-text-stroke-width, align-items, align-tracks, all, animation-composition, animation-direction, animation-iteration-count, animation-timeline, appearance, background-blend-mode, background-color, background-image, background-origin, background-position, background-position-y, background-repeat, border-block-color, border-block-end, border-block-end-color, border-block-end-style, border-block-start-width, border-block-style, border-bottom-color, border-end-end-radius, border-image, border-image-outset, border-inline-end-style, border-left, border-left-color, border-left-style, border-left-width, border-radius, border-right-width, border-spacing, border-top-left-radius, border-top-right-radius, border-top-width, box-sizing, color, color-scheme, column-count, column-fill, column-rule, column-width, contain, contain-intrinsic-block-size, contain-intrinsic-inline-size, contain-intrinsic-size, contain-intrinsic-width, container-type, content-visibility, container, counter-increment, counter-reset, counter-set, cursor, direction, empty-cells, filter, float, font, font-feature-settings, font-kerning, font-language-override, font-size, font-size-adjust, font-stretch, font-variant-emoji, grid-column-end, grid-column-start, grid-row-end, grid-row-start, grid-template-areas, grid-template-columns, grid-template-rows, hanging-punctuation, hyphenate-character, hyphens, image-rendering, inset-block, isolation, justify-items, justify-self, justify-tracks, letter-spacing, line-break, list-style-image, list-style-position, list-style-type, margin-bottom, margin-inline, margin-right, mask, mask-border-mode, mask-border-outset, mask-border-repeat, mask-border-slice, mask-border-source, mask-composite, mask-image, mask-origin, mask-position, math-depth, math-shift, math-style, max-width, offset-anchor, offset-distance, offset-position, opacity, outline, outline-color, outline-offset, outline-width, overflow, overflow-block, overflow-inline, overflow-wrap, overflow-x, overflow-y, padding, padding-block-end, padding-inline-start, page-break-before, paint-order, perspective, place-self, position, print-color-adjust, right, rotate, row-gap, ruby-align, scale, scroll-behavior, scroll-margin-block, scroll-margin-block-end, scroll-margin-block-start, scroll-margin-inline-end, scroll-margin-right, scroll-padding, scroll-padding-block, scroll-padding-block-end, scroll-padding-right, shape-image-threshold, shape-outside, tab-size, table-layout, text-align-last, text-combine-upright, text-decoration-color, text-decoration-line, text-decoration-skip, text-decoration-skip-ink, text-decoration-style, text-emphasis, text-emphasis-color, text-emphasis-position, text-indent, text-justify, text-rendering, text-size-adjust, text-transform, touch-action, transition-duration, transition-timing-function, vertical-align, white-space, word-break, writing-mode
<5	::after, ::marker, ::part, :blank, :empty, :first-of-type, :focus, :has, :hover, :lang, :nth-col, :nth-of-type, :playing, :required, @charset, @namespace, Adjacent sibling combinator, Attribute selectors, Class selectors, Descendant combinator, ID selectors, Selector list, Type selectors	-webkit-line-clamp, accent-color, align-content, align-self, animation-delay, backdrop-filter, backface-visibility, background, background-attachment, background-clip, background-position-x, block-size, border-block-end-width, border-block-start-color, border-block-start-style, border-block-width, border-bottom, border-bottom-left-radius, border-bottom-right-radius, border-bottom-style, border-bottom-width, border-color, border-end-start-radius, border-image-repeat, border-image-slice, border-image-source, border-image-width, border-inline-color, border-inline-end, border-inline-end-color, border-inline-end-width, border-inline-start-color, border-inline-start-style, border-inline-style, border-inline-width, border-right-color, border-right-style, border-start-end-radius, border-start-start-radius, border-style, border-top, border-top-color, border-top-style, caption-side, caret-color, clear, clip-path, column-gap, column-rule-color, column-rule-style, column-rule-width, column-span, container-name, display, flex, flex-basis, flex-direction, flex-flow, flex-grow, flex-shrink, flex-wrap, font-display, font-family, font-style, font-synthesis, gap (grid-gap), grid, grid-area, grid-auto-columns, grid-auto-flow, grid-auto-rows, grid-column, image-orientation, image-resolution, ime-mode, initial-letter, initial-letter-align, inline-size, justify-content, left, line-height, margin-block, margin-block-end, margin-block-start, margin-inline-end, margin-inline-start, margin-left, margin-top, margin-trim, mask-border, mask-clip, mask-mode, mask-repeat, mask-size, mask-type, masonry-auto-flow, offset-path, offset-rotate, order, orphans, outline-style, overflow-anchor, overflow-clip-margin, padding-block, padding-block-start, padding-bottom, padding-inline, padding-inline-end, padding-left, padding-right, padding-top, page-break-after, page-break-inside, place-items, pointer-events, quotes, resize, scroll-margin, scroll-margin-bottom, scroll-margin-inline, scroll-margin-inline-start, scroll-margin-left, scroll-margin-top, scroll-padding-block-start, scroll-padding-bottom, scroll-padding-inline, scroll-padding-inline-end, scroll-padding-inline-start, scroll-padding-left, scroll-padding-top, text-decoration, text-decoration-thickness, text-emphasis-style, text-shadow, unicode-bidi, user-select, will-change

Now we come onto the RAM section. Interestingly with the CSS results, browsers were highly optimized into delivering content without adding additional memory resources to that property or selector (with exception to the physical content on the page such as an image). As such no variable in the CSS spec gave a baseline reading that could be measured above the boilerplate in terms of physical memory usage (in megabytes).

MB	Selectors	Properties
0.1+	*None*	*None*

Next we come onto KB transferred and utilized by the variables. As many of the selectors, pseudos, and properties were so tiny in size they didn't add enough bytes (100) to measure 0.1 of a KB, they have been excluded from the results by default. Those which have a measurable (minor) impact are listed below.

KB	Selectors	Properties
0.2+	@container, @counter-style, @font-face, @font-feature-values, @font-palette-values, @keyframes	animation, offset
0.1	:is, :modal, :where, @color-profile, @property, @scroll-timeline	-webkit-line-clamp, align-items, align-self, animation-composition, animation-delay, animation-direction, animation-duration, animation-fill-mode, animation-iteration-count, animation-name, animation-play-state, animation-timeline, animation-timing-function, background-attachment, background-blend-mode, background-origin, background-position, background-position-x, background-position-y, background-repeat, background-size, border-block-color, border-block-end-style, border-block-end-width, border-block-start-color, border-block-start-style, border-block-start-width, border-block-style, border-image-repeat, border-image-slice, border-image-source, border-image-width, border-inline-end-color, border-inline-end-style, border-inline-end-width, border-inline-start, border-inline-start-color, border-inline-start-style, border-inline-start-width, border-inline-style, border-inline-width, counter-increment, counter-set, font, font-display, font-family, grid-area, grid-auto-columns, grid-auto-flow, grid-auto-rows, grid-column, grid-column-end, grid-column-start, grid-row, grid-row-end, grid-row-start, grid-template, justify-tracks, mask, mask-border, mask-border-source, place-self, rotate, shape-outside, transition, transition-property, translate

Finally we come onto the calculated score based upon the profile equation devised previously in the study to estimate how sustainable an object was based upon multiple variables (CPU, GPU, RAM and Data).

Score	Selectors	Properties
30+	@container, @font-palette-values, @keyframes	animation, offset
20-29	::grammar-error, :current, @font-face, @font-feature-values, @layer	font-variant-caps, scrollbar-color

Score	Selectors	Properties
10-19	::before, ::cue, ::cue-region, ::file-selector-button, ::first-letter, ::first-line, ::part, ::placeholder, ::selection, ::slotted, ::spelling-error, ::target-text, :active, :any-link, :autofill, :blank, :checked, :default, :defined, :dir, :disabled, :empty, :enabled, :first, :first-child, :first-of-type, :focus-visible, :focus-within, :fullscreen, :host, :host-context, :host-function, :hover, :in-range, :indeterminate, :invalid, :is, :lang, :last-child, :last-of-type, :left, :link, :modal, :not, :nth-child, :nth-last-child, :nth-last-of-type, :only-child, :only-of-type, :optional, :out-of-range, :past, :paused, :picture-in-picture, :placeholder-shown, :playing, :read-only, :read-write, :required, :right, :root, :scope, :target, :user-invalid, :user-valid, :valid, :visited, :where, @color-profile, @counter-style, @import, @media, @page, @property, @scroll-timeline, @supports, @viewport, Child combinator, Column combinator, Descendant combinator, General sibling combinator, ID selectors, Universal selectors	-webkit-text-fill-color, -webkit-text-stroke, -webkit-text-stroke-color, -webkit-text-stroke-width, align-content, align-items, align-tracks, all, animation-composition, animation-direction, animation-duration, animation-fill-mode, animation-iteration-count, animation-name, animation-play-state, animation-timeline, animation-timing-function, appearance, aspect-ratio, background, background-attachment, background-blend-mode, background-color, background-image, background-origin, background-position, background-position-x, background-position-y, background-repeat, background-size, border, border-block, border-block-color, border-block-end, border-block-end-color, border-block-end-style, border-block-end-width, border-block-start, border-block-start-style, border-block-start-width, border-bottom-color, border-bottom-left-radius, border-collapse, border-end-end-radius, border-end-start-radius, border-image, border-image-outset, border-inline, border-inline-end, border-inline-end-color, border-inline-end-style, border-inline-end-width, border-inline-start, border-inline-start-width, border-inline-width, border-left, border-left-color, border-left-style, border-left-width, border-radius, border-right, border-right-style, border-right-width, border-spacing, border-start-end-radius, border-start-start-radius, border-style, border-top, border-top-color, border-top-left-radius, border-top-right-radius, border-top-width, border-width, bottom, box-decoration-break, box-shadow, box-sizing, break-after, break-before, break-inside, clip, color, color-scheme, column-fill, column-rule, column-span, column-width, columns, contain, contain-intrinsic-height, contain-intrinsic-size, contain-intrinsic-width, container, container-name, container-type, content, content-visibility, counter-increment, counter-reset, counter-set, cursor, direction, empty-cells, flex-basis, float, font, font-feature-settings, font-kerning, font-language-override, font-optical-sizing, font-size, font-size-adjust, font-stretch, font-style, font-variant, font-variant-alternates, font-variant-east-asian, font-variant-emoji, font-variant-ligatures, font-variant-numeric, font-variant-position, font-variation-settings, font-weight, forced-color-adjust, grid-area, grid-column-end, grid-row, grid-row-end, grid-row-start, grid-template, grid-template-areas, grid-template-columns, grid-template-rows, height, hyphenate-character, hyphenate-limit-chars, hyphens, inset, inset-block-end, inset-block-start, inset-inline, inset-inline-end, inset-inline-start, isolation, justify-items, justify-self, justify-tracks, letter-spacing, line-break, line-height-step, list-style, list-style-image, list-style-position, list-style-type, margin, margin-block-start, margin-bottom, margin-inline, margin-right, margin-trim, mask, mask-border, mask-border-mode, mask-border-outset, mask-border-repeat, mask-border-slice, mask-border-source, mask-border-width, mask-clip, mask-composite, mask-image, mask-mode, mask-origin, mask-position, mask-repeat, mask-size, mask-type, masonry-auto-flow, math-depth, math-shift, math-style, max-block-size, max-height, max-inline-size, max-width, min-block-size, min-height, min-inline-size, min-width, mix-blend-mode, object-fit, object-position, offset-anchor, offset-distance, offset-position, opacity, outline-color, outline-offset, outline-width, overflow-inline, overflow-wrap, overflow-x, overflow-y, overscroll-behavior, overscroll-behavior-block, overscroll-behavior-inline, overscroll-behavior-x, overscroll-behavior-y, padding, padding-block-end, padding-bottom, padding-inline-start, page-break-before, paint-order, perspective, perspective-origin, place-content, place-self, position, right, rotate, row-gap, ruby-align, ruby-position, scale, scroll-behavior, scroll-margin, scroll-margin-block, scroll-margin-block-end, scroll-margin-block-start, scroll-margin-bottom, scroll-margin-inline, scroll-margin-inline-end, scroll-margin-inline-start, scroll-margin-left, scroll-margin-right, scroll-margin-top, scroll-padding, scroll-padding-block, scroll-padding-block-end, scroll-padding-inline-start, scroll-padding-left, scroll-padding-right, scroll-padding-top, scroll-snap-align, scroll-snap-stop, scroll-snap-type, scrollbar-gutter, scrollbar-width, shape-image-threshold, shape-margin, shape-outside, tab-size, table-layout, text-align, text-align-last, text-combine-upright, text-decoration, text-decoration-color, text-decoration-line, text-decoration-skip, text-decoration-skip-ink, text-decoration-style, text-decoration-thickness, text-emphasis, text-emphasis-color, text-emphasis-position, text-emphasis-style, text-indent, text-justify, text-orientation, text-overflow, text-rendering, text-shadow, text-size-adjust, text-transform, text-underline-offset, text-underline-position, top, touch-action, transform, transform-box, transform-origin, transform-style, transition, transition-delay, transition-duration, transition-property, transition-timing-function, translate, user-select, vertical-align, visibility, white-space, widows, width, will-change, word-break, word-spacing, writing-mode, z-index

Score	Selectors	Properties
<10	::after, ::backdrop, ::marker, :focus, :future, :has, :local-link, :nth-col, :nth-last-col, :nth-of-type, :target-within, @charset, @namespace, Adjacent sibling combinator, Attribute selectors, Class selectors, Selector list, Type selectors	-webkit-line-clamp, accent-color, align-self, animation-delay, backdrop-filter, backface-visibility, background-clip, block-size, border-block-start-color, border-block-style, border-block-width, border-bottom, border-bottom-right-radius, border-bottom-style, border-bottom-width, border-color, border-image-repeat, border-image-slice, border-image-source, border-image-width, border-inline-color, border-inline-start-color, border-inline-start-style, border-inline-style, border-right-color, border-top-style, caption-side, caret-color, clear, clip-path, column-count, column-gap, column-rule-color, column-rule-style, column-rule-width, contain-intrinsic-block-size, contain-intrinsic-inline-size, display, filter, flex, flex-direction, flex-flow, flex-grow, flex-shrink, flex-wrap, font-display, font-family, font-synthesis, gap (grid-gap), grid, grid-auto-columns, grid-auto-flow, grid-auto-rows, grid-column, grid-column-start, hanging-punctuation, image-orientation, image-rendering, image-resolution, ime-mode, initial-letter, initial-letter-align, inline-size, inset-block, justify-content, left, line-height, margin-block, margin-block-end, margin-inline-end, margin-inline-start, margin-left, margin-top, offset-path, offset-rotate, order, orphans, outline, outline-style, overflow, overflow-anchor, overflow-block, overflow-clip-margin, padding-block, padding-block-start, padding-inline, padding-inline-end, padding-left, padding-right, padding-top, page-break-after, page-break-inside, place-items, pointer-events, print-color-adjust, quotes, resize, scroll-padding-block-start, scroll-padding-bottom, scroll-padding-inline, scroll-padding-inline-end, unicode-bidi

Media Formats

Next we shall examine the results of the various web media formats (separated for ease of readability) to see how they perform under the different test suite metrics. Below are the CPU results, note that the amount of time to render is considerably higher than that for HTML and CSS due to the media being rendered to screen.

MS	Images	Audio	Video
30+		Ogg	
20-29		AAC, FLAC, MP3, MP4, Opus, WAV, WebM	AVI, MKV, MOV, MP4, MPEG, WMV, WebM
15-19	AVIF, BMP, GIF, JPEG, PNG, SVG, WebP		

Next up are the GPU results, these will be similar to the CPU results except notably as media is being rendered, its critically important to keep the MS down as faster media means a better user-experience and sustainability.

MS	Images	Audio	Video
40+		AAC, FLAC, MP3, MP4, Ogg, Opus, WAV, WebM	MPEG
30-39	BMP		AVI, MKV, MP4, WMV, WebM
20-29	AVIF, GIF, JPEG, PNG, SVG, WebP		MOV

Next up is the RAM usage results. As each media format requires the rendering of an internal media player within it's browser and the loading and playing of a file uses system resources, the results of this test will be really useful in determining the intensity of which different formats hit hardware on a temporary basis.

MB	Images	Audio	Video
6+			AVI, MP4, MPEG
5-5.9		AAC, FLAC, MP3, Ogg, Opus, WAV	WebM, WMV
4-4.9		MP4, WebM	MKV, MOV
3-3.9	AVIF, BMP, GIF, JPEG, PNG, SVG, WebP		

Now we have KB usage (bytes transferred). With media, this will encompass the non-cached total amount of the full media file (non-cached) excluding a boilerplate or internal media player (to ensure isolated accuracy).

KB	Images	Audio	Video
1MB+			AVI, MP4, MPEG, WMV, WebM
500-1MB	BMP		MKV, MOV
250-500		FLAC, WAV	
100-250	GIF, PNG		
1-100	AVIF, JPEG, SVG, WebP	AAC, MP3, MP4, Ogg, Opus, WebM	

Finally it's time to examine the score which comprises the CPU, GPU, RAM and Data averages of the media formats as they underwent benchmarking within the test suite. The scores are very different to HTML and CSS.

Score	Images	Audio	Video
500+	BMP		AVI, MKV, MOV, MP4, MPEG, WMV, WebM
250-499		FLAC, WAV	
100-249	GIF, PNG	AAC, MP3, MP4, Ogg, Opus, WebM	
<100	AVIF, JPEG, SVG, WebP		

Other Specifications

Finally we move onto the other specifications tested. These include asset files like TXT, XML, JSON, and other useful files which are often inclusive of a semantic document but little is known about their carbon footprint. Rather than testing every element possible for these files, a simple boilerplate has been produced for each and as this will work in a live environment, it will produce an average case to calculate potential emissions.

First lets examine the CPU results (in MS):

MS	Asset
20+	atom.xml, browserconfig.xml, clientaccesspolicy.xml, dnt-policy.txt, feed.json, feed.opml, itunes.xml, opensearch.xml, p3p.xml, powder.xml, rss.xml, site.webmanifest, sitemap.xml, subtitles.vtt
10-19	ads.txt, carbon.txt, change.log, crossdomain.xml, favicon.ico, foaf.rdf, geo.kml, humans.txt, robots.txt, security.txt
<10	.htaccess, dublin.rdf, event.ics, geo.rdf, PICS.rdf, README, vcard.vcf

Next here are the GPU results (in MS):

MS	Asset
20+	ads.txt, atom.xml, browserconfig.xml, carbon.txt, change.log, clientaccesspolicy.xml, crossdomain.xml, dnt-policy.txt, dublin.rdf, event.ics, favicon.ico, feed.json, feed.opml, foaf.rdf, geo.kml, humans.txt, itunes.xml, opensearch.xml, p3p.xml, powder.xml, robots.txt, rss.xml, security.txt, site.webmanifest, sitemap.xml, subtitles.vtt, vcard.vcf
<20	.htaccess, geo.rdf, PICS.rdf, README

Following on, here are the RAM usage (in MB) results:

MB	Asset
4-4.5	atom.xml, browserconfig.xml, clientaccesspolicy.xml, crossdomain.xml, feed.json, itunes.xml, opensearch.xml, p3p.xml, site.webmanifest, sitemap.xml, subtitles.vtt
3.5-3.9	.htaccess, ads.txt, carbon.txt, change.log, dnt-policy.txt, favicon.ico, feed.opml, geo.rdf, humans.txt, powder.xml, robots.txt, rss.xml, security.txt, vcard.vcf
<3.5	dublin.rdf, event.ics, foaf.rdf, geo.kml, PICS.rdf, README

And next, here are the data transfer (disk usage) results (in KB):

KB	Asset
5+	dnt-policy.txt, favicon.ico
1.1-5	.htaccess, PICS.rdf
0.6-1	geo.kml, humans.txt, itunes.xml, powder.xml, vcard.vcf
0.2-0.5	atom.xml, browserconfig.xml, change.log, clientaccesspolicy.xml, crossdomain.xml, dublin.rdf, event.ics, feed.json, feed.opml, foaf.rdf, geo.rdf, opensearch.xml, p3p.xml, README, rss.xml, site.webmanifest, sitemap.xml
0.1	ads.txt, carbon.txt, robots.txt, security.txt, subtitles.vtt

Finally, here is the computed score based on CPU, GPU, RAM and Data:

Score	Asset
50+	dnt-policy.txt, itunes.xml, rss.xml, sitemap.xml
40-49	ads.txt, atom.xml, browserconfig.xml, carbon.txt, change.log, clientaccesspolicy.xml, crossdomain.xml, favicon.ico, feed.json, feed.opml, humans.txt, opensearch.xml, p3p.xml, powder.xml, robots.txt, security.txt, site.webmanifest, subtitles.vtt
30-39	.htaccess, dublin.rdf, foaf.rdf, geo.kml, geo.rdf, PICS.rdf
<30	event.ics, README, vcard.vcf

Discussion

Below are my thoughts on the results listed above. I will attempt to examine if my hypothesis was correct and make recommendations using the research I have undertaken. With this, web developers may be able to gain a greater understanding of the importance of precision coding and the dangers of technical debt. I shall also use this opportunity to point out any strengths and weaknesses in the research and if there are any potential future directions in which other researchers may want to examine in order to gain further insight into how web development could be made more sustainable (or at least how it can be more accurately measured).

Interpretation & Recommendations

Several conclusions can be drawn from the data in this study (as noted within the above tables), and further notes can be uncovered within the table of results which is attached to this research paper. Within the HTML specification one of the biggest triggers for CPU, GPU, RAM and Data usage was the (not surprising) use and embedding of interactive content. Media such as video and audio take time to render as do images, but also consider the effect of canvas, iframes, embeds, plugins, and other rendered materials as this is processor intensive. Another item which showed up time and again was that deprecated or non-standard code has a much higher chance of triggering issues with rendering (and causing performance issues) which in turn will act as a barrier to sustainability. Finally, elements that have additional "on-site" rendering requirements such as form inputs, and attributes that affect objects post paint include disabled, csp, sandbox and contenteditable.

On-site rendering requirements could include specialized inputs like password fields or fixed field inputs such as date fields (which display a calendar). Both of which have to validate their input and render the results and trigger a render burst. Below I've noted elements and properties which require special consideration as they can affect the sustainability of a product beyond this studies scope, as such recommendations are included.

- **Benefit:** async, defer, loading, preload, sizes
- **Disadvantage:** <marquee>, autoplay, loop

The CSS specification has a few more explicit items of interest. The first obvious conclusion is that using any form of animation or movement will impact the page greatly (keyframes being a prime example, as is offset and animation). However, it should be noted that using custom fonts also has a dramatic impact on all four variables being measured (and therefore should only be used in proportion). Interestingly, there are a few form based features like grammar-error and spelling-error which are resource taxing, plus changing your scrollbar can be quite carbon intensive (based upon the overall score metric). As can CSS Cascade layers and CSS containers alongside other bleeding edge techniques (possibly due to needing further optimization within individual web browsers either during the rendering process or how it handles lack of support).

- **Benefit:** @media print, @page, :target, image-resolution.
- **Disadvantage:** @font-face, @import, @container, @font-palette-values, @keyframes, :current, animation, cursor, offset.

When it comes to media formats, SVG is the best with AVIF slightly pushing ahead of WebP (though they are very close together. The worse offender by far was (no surprise) BMP. In audio, there was little to differentiate however WebM was very efficient above all others. The slowest was WAV purely because of it being lossless. With video there was little to differentiate however with all things being equal WebM again makes a solid statement for compatibility and speed, with MPEG being the slowest perhaps due to it's weaker encoding.

Each of the web asset formats performed well on the benchmarks, and the great news is they were small enough that the overhead made little difference to the overall impact of the website. If you find yourself wanting to improve the accessibility, UX, performance of sustainability of a site and one of these formats does apply to your work, consider investing in them as they do provide added benefits to your visitors worldwide.

Browser vendors may wish to examine the various results tables, find any loose nails (poor results which could be made more sustainable), perform additional testing to quantify the results and perhaps fix any flaws in their rendering engines. If the rendering process can be sped up in MS, it all adds up over billions of page-views.

Further recommendations beyond the results are harder to quantify as this research appears to back up most common best practices: HTML semantics are critically important (non-standard and obsolete code will affect your sustainability), accessibility is also useful to the process (a legible document flow, alt attributes, keyboard focus, progressive enhancement, etc; will all smooth the jagged edges of a green design), finally (and of the most critically important), ensuring a site is performant will push for sustainability as fewer resources used will result in less demand upon equipment at all ends of the chain (which benefits the worlds electricity supply).

This does however bring into context current practices which are engaged with by large vendors and the developers for existing websites. Existing evidence shows that websites are getting more bloated **[54]**, sites remain largely inaccessible **[55]**, and with JavaScript taking higher importance than basic semantics in tooling practices, the backbone of the web (HTML & CSS) have become clogged with un-performant code. The existing web needs a firm shake-up if it is to meet it's commitments to become sustainable (and user-friendly).

Strengths and Weaknesses

The main strengths of this research are in the results (being in many cases the first attempt at measuring the sustainability and performant reliability of existing web standards) and that it stands as a talking (or at least a jumping-off) point to begin more accurately measuring a websites carbon footprint; beyond the existing measurement techniques of estimating based on total page size (and little else). By having more variables to work with it gives a much wider view of how impactful a website can be sustainable, and where the pinch-points can be found within a design. Rather than focusing on a "data is all" basis of measurement, it gives equal focus to other important variables implying the credible importance that these have upon the web sustainability of an app or website. Additional strengths include a model for other researchers to produce original research to further identify sustainability issues, backup or disclaim any issues within this research, explicit websites, wider implementation of standards or to adapt it with inclusivity with additional variables to make the results even more reliable. The methodology of avoiding bias, and especially developing a model / equation which can be used to algorithmically calculate a sites sustainability (within the scope of the W3C standards) was of critical importance, as something that can be implemented publicly, has real-world value.

The main weaknesses are that being able to identify poorly performing aspects of a specification may be one thing, but actually enacting change as a result is much harder. There is a reason why those features exist within specifications and browsers and while optimizations can be made, sometimes it's just a fact of life that with the best will in the world; having certain necessary features on a product or service (say streaming video) will be conducive to heavy energy use both in data transfer and battery drain for the consumer. As such, knowing how to best optimize these pitfalls and to build defensively can soften the impact and potentially reduce the ever increasing drain upon scarce resources that the Internet currently relies so heavily upon. Another weakness of this study is that its main emphasis has just been placed on the primary technologies of the web (HTML and CSS) along with various media formats which get embedded within it. Due to the complex nature of JavaScript and server-side languages, they have largely been excluded, aside from the inclusion of HTML script tags. This is because calculating the impact of scripts is more difficult due to the way in which they process and render, unlike other Web languages which render "as-is" upon download.

Future Research Directions

There are many directions in which further research projects could attempt to follow. As my study has been primarily aimed at HTML and CSS, a closer examination at the carbon impact of JavaScript could be useful to Web developers. A more granular examination of the ES6+ could enhance our knowledge for the purposes of calculating energy usage; though a different form of measurement may need to be utilized as the rendering path for JavaScript differs to other Web languages. Additional routes of study could also examining the impact of other variables (outside of web standards) upon energy use. An example of this being accessibility tooling, and if being accessible makes a site carbon positive, neutral, or negative. If a researcher decides to carry out a study in relation to the work carried out within this paper, please get in touch. Web sustainability is an evolving field and requires study from it's engineers to better understand how climate change impacts the industry. By learning about how energy efficient source code is, visitors lives and experiences can be improved.

References

1. MightyBytes & Wholegrain Digital. (2022). Calculating Digital Emissions - Sustainable Web Design. [online] Available at: https://sustainablewebdesign.org/calculating-digital-emissions/
2. Harrabin, R. (2020). 'Ditch 4K video and new tech to fight climate change'. [online] BBC News. Available at: https://www.bbc.co.uk/news/technology-55164410
3. Lenox, J. (2019). How Improving Website Performance Can Help Save The Planet — Smashing Magazine. [online] Smashing Magazine. Available at: https://www.smashingmagazine.com/2019/01/save-planet-improving-website-performance/
4. Mozilla.org. (2022). Progressive Enhancement - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Glossary/Progressive_Enhancement
5. WAI. (2022). Introduction to Web Accessibility. [online] Web Accessibility Initiative (WAI). Available at: https://www.w3.org/WAI/fundamentals/accessibility-intro/
6. Wikipedia Contributors (2022). Comparison of browser engines. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Comparison_of_browser_engines
7. Jones, N. (2018). How to stop data centres from gobbling up the world's electricity. Nature, [online] 561(7722), pp.163–166. doi:10.1038/d41586-018-06610-y.
8. Wholegrain Digital. (2022). Sustainable Web Manifesto. [online] Available at: https://www.sustainablewebmanifesto.com/
9. Davison, B.D. (2001). A Web caching primer. IEEE Internet Computing, 5(4), pp.38–45. doi:10.1109/4236.939449.
10. Meder, Antonov, & Chang. (2017). Driving user growth with performance improvements - Pinterest Engineering Blog - Medium. [online] Medium. Available at: https://medium.com/pinterest-engineering/driving-user-growth-with-performance-improvements-cfc50dafadd7.
11. Russell, A. (2022). The Performance Inequality Gap, 2023. [online] Infrequently Noted. Available at: https://infrequently.org/2022/12/performance-baseline-2023/
12. Wikipedia Contributors (2022). Green hosting. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Green_hosting
13. Andrew, R. (2018). A Guide To The State Of Print Stylesheets In 2018 — Smashing Magazine. [online] Smashing Magazine. Available at: https://www.smashingmagazine.com/2018/05/print-stylesheets-in-2018/
14. Mozilla.org. (2022). Accessibility | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/Accessibility
15. W3.org. (2022). The W3C Markup Validation Service. [online] Available at: https://validator.w3.org/
16. Patel, N. (2017). The Paradox of Choice: Why Less is More in UX Design. [online] Getfeedback.com. Available at: https://www.getfeedback.com/resources/ux/paradox-choice-less-ux-design/
17. Vij, V. (2014). Nine tech companies helping build renewably powered internet | Reuters Events | Sustainable Business. [online] Available at: https://www.reutersevents.com/sustainability/environment/nine-tech-companies-helping-build-renewably-powered-internet
18. Hinsdale, J. (2022). Cryptocurrency's Dirty Secret: Energy Consumption. [online] State of the Planet. Available at: https://news.climate.columbia.edu/2022/05/04/cryptocurrency-energy/
19. McAfee. (2009). The Carbon Footprint of Email Spam Report Key Findings. (n.d.). [online] Available at: https://thegreenemail.com/assets/files/mcafee_report_carbon_footprint_of_email.pdf
20. Bedingfield, W. (2021). We finally know how bad for the environment your Netflix habit is. [online] WIRED UK. Available at: https://www.wired.co.uk/article/netflix-carbon-footprint
21. Powell, B. (2021). Reducing Carbon Emissions On The Web — Smashing Magazine. [online] Smashing Magazine. Available at: https://www.smashingmagazine.com/2021/09/reducing-carbon-emissions-on-web/

22. Aslan, J. (2022). Climate change implications of gaming products and services. [online] Available at: https://openresearch.surrey.ac.uk/esploro/outputs/doctoral/Climate-change-implications-of-gaming-products-and-services/99512335802346
23. Coroamă, V. (2021). Investigating the Inconsistencies among Energy and Energy Intensity Estimates of the Internet Metrics and Harmonising Values. (n.d.). [online] Available at: https://www.aramis.admin.ch/Default?DocumentID=67656&Load=true
24. Carbon Emissions Implications of ICT Re-use at the University of Edinburgh. (N.d.). [online] Available at: https://www.ed.ac.uk/files/atoms/files/pc-carbonfootprints-jh-ecci2.pdf
25. Everts, T. (2022). SpeedCurve | Ten years of page bloat: What have we learned? [online] Available at: https://www.speedcurve.com/blog/ten-years-page-bloat/
26. Website Carbon Calculator. (2022). How does it work? - Website Carbon Calculator. [online] Available at: https://www.websitecarbon.com/how-does-it-work/
27. Schneider, C. (2022). Digital Carbon Footprint: The Current State of Measuring Tools. [online] Marmelab.com. Available at: https://marmelab.com/blog/2022/04/05/greenframe-compare.html
28. Mozilla.org. (2022). Performance fundamentals - Web Performance | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/Performance/Fundamentals
29. Google Developers (2022). Fast load times. [online] web.dev. Available at: https://web.dev/fast/
30. Grigorik, I. (2013). High Performance Browser Networking (O'Reilly). [online] High Performance Browser Networking. Available at: https://hpbn.co/
31. Friedman, V. (2021). Front-End Performance Checklist 2021 (PDF, Apple Pages, MS Word) — Smashing Magazine. [online] Smashing Magazine. Available at: https://www.smashingmagazine.com/2021/01/front-end-performance-2021-free-pdf-checklist/
32. Lozano, K. (2019). Can the Internet Survive Climate Change? [online] The New Republic. Available at: https://newrepublic.com/article/155993/can-internet-survive-climate-change
33. WHATWG. (2022). HTML Standard. [online] Available at: https://html.spec.whatwg.org/multipage/
34. W3.org. (2015). All CSS specifications. [online] Available at: https://www.w3.org/Style/CSS/specs.en.html
35. Mozilla.org. (2021). Media type and format guide: image, audio, and video content - Web media technologies | MDN. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/Media/Formats
36. CSS-Tricks. (2021). Working With Web Feeds: It's More Than RSS | CSS-Tricks. [online] Available at: https://css-tricks.com/working-with-web-feeds-its-more-than-rss/
37. GitHub. (2022). Build software better, together. [online] Available at: https://github.com/
38. Netlify. (2022). Netlify: Develop & deploy the best web experiences in record time. [online] Netlify. Available at: https://www.netlify.com/
39. Giraudel, H. (2022). Browserhacks. [online] Browserhacks.com. Available at: http://browserhacks.com/
40. Russell, A. (2021). Infrequently Noted. [online] Infrequently Noted. Available at: https://infrequently.org/series/browser-choice-must-matter/
41. Google. (2017). Google Chrome - Download the Fast, Secure Browser from Google. [online] Available at: https://www.google.com/chrome/
42. Mozilla. (2022). Download the fastest Firefox ever. [online] Available at: https://www.mozilla.org/firefox/ [Accessed 30 Aug. 2022].
43. Webkit. (2019). How Web Content Can Affect Power Usage. [online] WebKit. Available at: https://webkit.org/blog/8970/how-web-content-can-affect-power-usage/
44. Klein, H. (2014). Impact of GPU Acceleration on Browser CPU Usage. [online] Helgeklein.com. Available at: https://helgeklein.com/blog/impact-gpu-acceleration-browser-cpu-usage/
45. Lawson, N. (2022). Memory leaks: the forgotten side of web performance. [online] Available at: https://nolanlawson.com/2022/01/05/memory-leaks-the-forgotten-side-of-web-performance/

46. Thiagarajan, N., Aggarwal, G., Nicoara, A., Boneh, D. and Singh, J. (n.d.). Who Killed My Battery: Analyzing Mobile Browser Energy Consumption. [online] Available at: https://mobisocial.stanford.edu/papers/boneh-www2012.pdf

47. buildcomputers.net. (2012). Typical Power Consumption of PC Components - Power Draw in Watts. [online] Available at: https://www.buildcomputers.net/power-consumption-of-pc-components.html

48. Apple (2019). Mac Pro: Power consumption and thermal output (BTU/h) information. [online] Apple Support. Available at: https://support.apple.com/en-us/HT201796

49. Walton, J. (2021). Graphics Card Power Consumption and Efficiency Tested. [online] Tom's Hardware. Available at: https://www.tomshardware.com/features/graphics-card-power-consumption-tested

50. Crucial. (2020). How much power does memory use? [online] Available at: https://www.crucial.com/support/articles-faq-memory/how-much-power-does-memory-use

51. Andrae A. S.G. (2019). New perspectives on internet electricity use in 2030. [online] PISRT. Available at: https://pisrt.org/psr-press/journals/easl-vol-3-issue-2-2020/new-perspectives-on-internet-electricity-use-in-2030/

52. Aslan, J., Mayers, K., Koomey, J.G. and France, C. (2017). Electricity Intensity of Internet Data Transmission: Untangling the Estimates. Journal of Industrial Ecology, [online] 22(4), pp.785–798. doi:10.1111/jiec.12630.

53. Kamiya, G. (2020). The carbon footprint of streaming video: fact-checking the headlines – Analysis - IEA. [online] IEA. Available at: https://www.iea.org/commentaries/the-carbon-footprint-of-streaming-video-fact-checking-the-headlines

54. SpeedCurve. (2022). SpeedCurve | Ten years of page bloat: What have we learned? [online] Available at: https://www.speedcurve.com/blog/ten-years-page-bloat/

55. WebAIM. (2022). WebAIM: The WebAIM Million - The 2022 report on the accessibility of the top 1,000,000 home pages. [online] Available at: https://webaim.org/projects/million/