CYBERSOFT ACADEMY

http://cybersoft.edu.vn

CYBERSOFT

ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH

Front-end Performance Checklist

INTRODUCTION

COMPONENTS OF CHECKLIST

PERFOMANCE TOOLS

REFERENCE

INTRODUCTION

INTRODUCTION

Performance (hiệu suất) is a huge subject, but it's not always a "back-end" or an "admin" subject: it's a Front-End responsibility too.

The Front-End Performance Checklist is an exhaustive list of elements you should check or at least be aware of.

INTRODUCTION

- There are many rules of this checklist, they’re divided into seven groups:

1. HTML
2. CSS
3. Fonts
4. Images
5. JavaScript
6. Server
7. JS Framework

INTRODUCTION

HOW TO USE ???

❖For each rule, you will have a paragraph explaining *why* this rule is important and *how* you can fix it.

❖For more deep information, after each rule there are some articles and links which you can refer to complete the checklist.

INTRODUCTION

RULE’S PRIORITY (độ ưu tiên)

❖All items in the Front-End Performance Checklist are essentials to achieve the highest performance score. But each rule has a different prioritized level. It’s noted for ease monitoring.

INTRODUCTION

RULE’S PRIORITY

There are 3 levels of priority:

❖**Low** the item has a low priority.

❖**Medium** the item has a medium priority. You shouldn't avoid tackling that item.

❖**High** the item has a high priority. You can't avoid following that rule and implement the corrections recommended.

COMPONENTS OF CHECKLIST

COMPONENTS OF CHECKLIST

HTML

COMPONENTS OF CHECKLIST - HTML

Minified HTML (tinh giảm HTML)

➢The HTML code is minified, comments, white spaces and new lines are removed from production files.

Why: Removing all unnecessary spaces, comments and break will reduce the size of your HTML and speed up your site's page load times and obviously lighten the download for your user.

How: Most of the frameworks have plugins to facilitate the minification of the webpages. You can use a bunch of NPM modules that can do the job for you automatically.

COMPONENTS OF CHECKLIST - HTML

Minified HTML

➢Tool:

- HTML minifier | Minifi Code - http://minifycode.com/html-minifier/
- Online HTML Compressor - http://refresh-sf.com/

➢Article:

- *Experimenting with HTML minifier — Perfection Kills http://perfectionkills.com/experimenting-with-html-minifier/#use_short_doctype*

COMPONENTS OF CHECKLIST - HTML

Remove unnecessary comments

➢Ensure that comments are removed from your pages.

Why: Comments are not really useful for the user and should be removed from production files. One case where you want to keep comments could be if you need to keep the origin for a library.

How: Most of the time, comments can be removed using an HTML minify plugin.

COMPONENTS OF CHECKLIST - HTML

Remove unnecessary comments

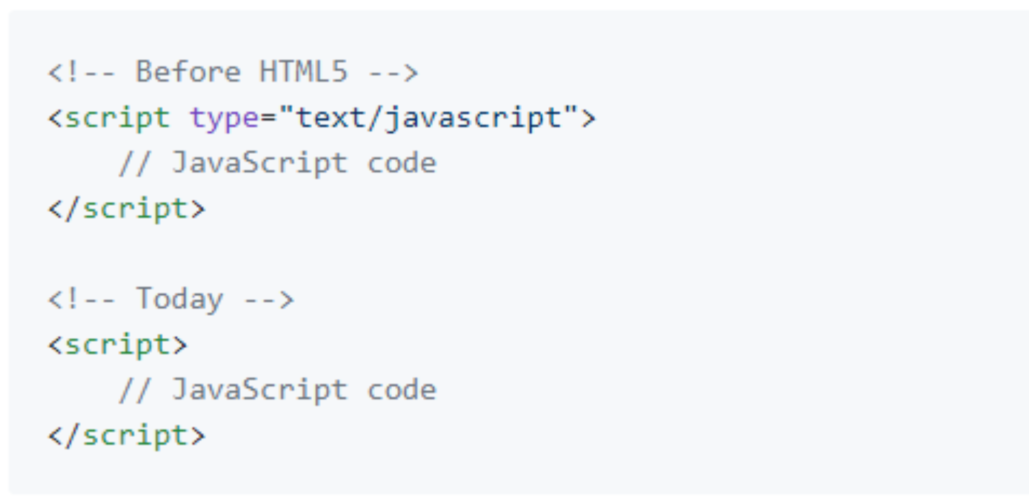
➢Tool:

- Remove-html-comments - npm (https://www.npmjs.com/package/remove-html-comments)

COMPONENTS OF CHECKLIST - HTML

Remove unnecessary attributes
(xóa thuộc tính không cần thiết)

➢Type attribute like type="text/javascript“ or type="text/css“ are not required anymore and should be removed



```
<!-- Before HTML5 -->
<script type="text/javascript">
    // JavaScript code
</script>

<!-- Today -->
<script>
    // JavaScript code
</script>
```

COMPONENTS OF CHECKLIST - HTML

Remove unnecessary attributes

Why: Type attributes are not necessary as HTML5 implies text/css and text/javascript as defaults. Unused code should be removed when not used by your website or app as they add more weight to your pages.

How: Ensure that all your <link> and <script> tags don't have any type attribute.

COMPONENTS OF CHECKLIST - HTML

Remove unnecessary attributes

➢Article:

- The Script Tag | CSS-Tricks (https://css-tricks.com/the-script-tag/)

COMPONENTS OF CHECKLIST - HTML

Place CSS tags always before JavaScript tags (đặt thẻ CSS trước thẻ JavaScript)

➢Ensure that your CSS is always loaded before having JavaScript code.

Why: Having your CSS tags before any JavaScript enables better, parallel download which speed up browser rendering time.

How: Ensure that <link> and <style> in your <head> are always before your <script>.

COMPONENTS OF CHECKLIST - HTML

Place CSS tags always before JavaScript tags

- Article:
  - Ordering your styles and scripts for pagespeed (https://varvy.com/pagespeed/style-script-order.html)

COMPONENTS OF CHECKLIST

CSS

COMPONENTS OF CHECKLIST - CSS

Minification (tinh giảm CSS)

➢All CSS files are minified, comments, white spaces and new lines are removed from production files.

Why: When CSS files are minified, the content is loaded faster and less data is sent to the client. It's important to always minify CSS files in production. It is beneficial for the user as it is for any business who wants to lower bandwidth costs and lower resource usage.

How: Use tools to minify your files automatically before or during your build or your deployment

COMPONENTS OF CHECKLIST - CSS

Minification

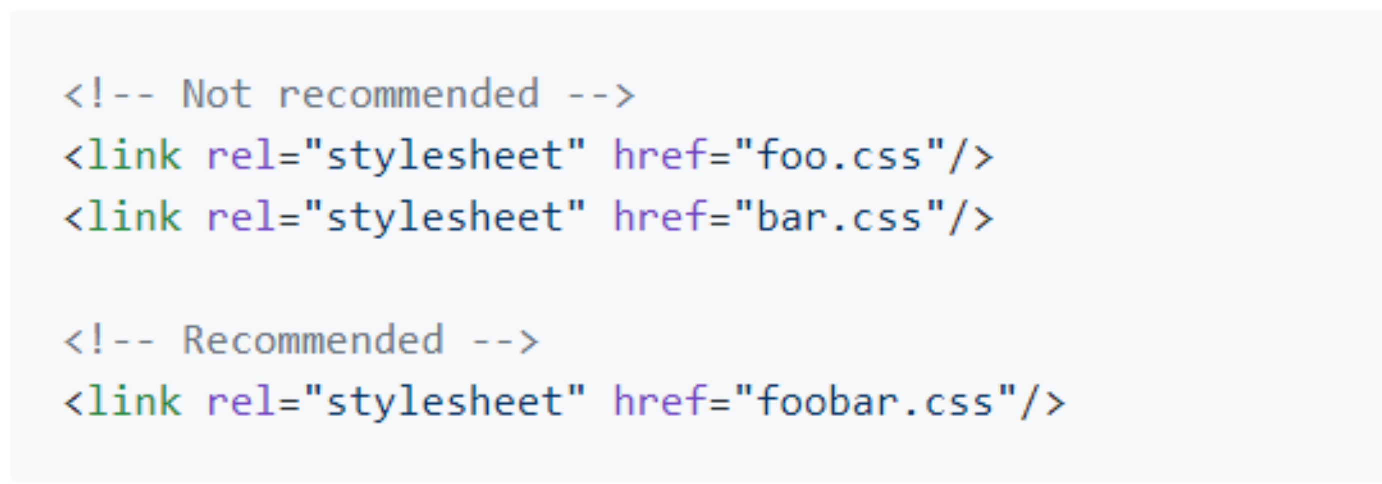
➢Tool:

- Cssnano: A modular minifier based on the PostCSS ecosystem. - cssnano (https://cssnano.co/)
- @neutrinojs/style-minify - npm (https://www.npmjs.com/package/@neutrinojs/style-minify)
- Online CSS Compressor (http://refresh-sf.com/)

COMPONENTS OF CHECKLIST - CSS

Concatenation (liên kết)

➢CSS files are concatenated in a single file *(Not always valid for HTTP/2)*.



```
<!-- Not recommended -->
<link rel="stylesheet" href="foo.css"/>
<link rel="stylesheet" href="bar.css"/>

<!-- Recommended -->
<link rel="stylesheet" href="foobar.css"/>
```

COMPONENTS OF CHECKLIST - CSS

Concatenation

Why:

If you are still using HTTP/1, you may need to still concatenate your files, it's less true if your server use HTTP/2 (tests should be made).

How:

- Use online tool or any plugin before or during your build or your deployment to concatenate your files.
- Ensure, of course, that concatenation does not break your project.

COMPONENTS OF CHECKLIST - CSS

Concatenation

➢Article:

- HTTP: Optimizing Application Delivery - High Performance Browser Networking (O'Reilly) (https://hpbn.co/optimizing-application-delivery/#optimizing-for-http2)

- Performance Best Practices in the HTTP/2 Era (https://deliciousbrains.com/performance-best-practices-http2/)

COMPONENTS OF CHECKLIST - CSS

Non-blocking (không bị chặn)

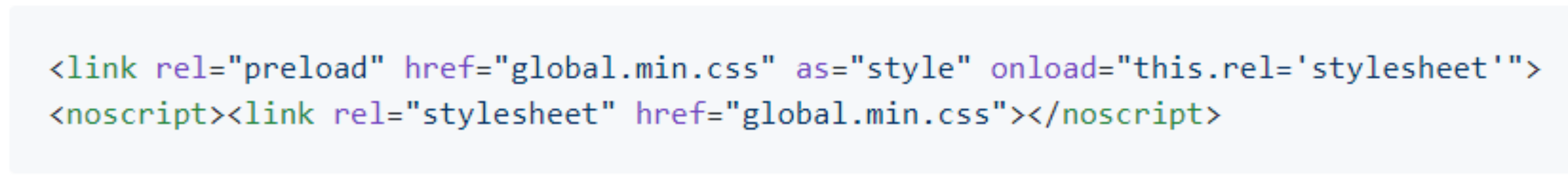
➢CSS files need to be non-blocking to prevent the DOM from taking time to load.

Why:

CSS files can block the page load and delay the rendering of your page. Using preload can actually load the CSS files before the browser starts showing the content of the page.

How:

You need to add the rel attribute with the preload value and add as="style" on the <link> element.



```
<link rel="preload" href="global.min.css" as="style" onload="this.rel='stylesheet'">
<noscript><link rel="stylesheet" href="global.min.css"></noscript>
```

COMPONENTS OF CHECKLIST - CSS

Non-blocking

Article:

- Example of preload CSS using loadCSS (https://gist.github.com/thedaviddias/c24763b82b9991e53928e66a0bafc9bf)
- Preloading content with rel="preload" (https://developer.mozilla.org/en-US/docs/Web/HTML/Preloading_content)
- Preload: What Is It Good For? — Smashing Magazine (https://www.smashingmagazine.com/2016/02/preload-what-is-it-good-for/)

COMPONENTS OF CHECKLIST - CSS

Length of CSS classes (độ dài tên class)

➢The length of your classes can have an (slight) impact on your HTML and CSS files (eventually)

Why:

Even performance impacts can be disputable, taking a decision on a naming strategy regarding your project can have a substantial impact on the maintainability of your stylesheets. If you are using BEM, in some cases, you can end up with classes having more characters than needed. It's always important to choose wisely your names and namespaces.

COMPONENTS OF CHECKLIST - CSS

Length of CSS classes

How:

Setting a limit in terms of number of characters could be interesting for some people, but ensuring that you broke down your website in components can help to reduce the amount of classes (and declarations) and the length of your classes.

➢Tool:

- long vs short class · jsPerf (https://jsperf.com/long-vs-short-class)

COMPONENTS OF CHECKLIST - CSS

Unused CSS (CSS không sử dụng)

➢The length of your classes can have an (slight) impact on your HTML and CSS files (eventually)

Why:

Removing unused CSS selectors can reduce the size of your files and then speed up the load of your assets.

How:

Always check if the framework CSS you want to use don't already has a reset / normalize code included. Sometimes you may not need everything that is inside your reset / normalize file

COMPONENTS OF CHECKLIST - CSS

Unused CSS

Tool:

- UnCSS Online (https://uncss-online.com/)
- PurifyCSS (https://github.com/purifycss/purifycss)
- PurgeCSS (https://github.com/FullHuman/purgecss)
- Chrome DevTools Coverage (https://developers.google.com/web/updates/2017/04/devtools-release-notes#coverage)

COMPONENTS OF CHECKLIST - CSS

CSS Critical (CSS cốt lõi)

Why:

Inlining critical CSS help to speed up the rendering of the web pages reducing the number of requests to the server.

How:

Generate the CSS critical with online tools or using a plugin like the one that Addy Osmani developed.

COMPONENTS OF CHECKLIST - CSS

CSS Critical

➢Tool:

- Critical by Addy Osmani on GitHub (https://github.com/addyosmani/critical)
- Critical Path CSS Generator - Prioritize above the fold content :: SiteLocity (https://www.sitelocity.com/critical-path-css-generator)

➢Article:

- Understanding Critical CSS (https://www.smashingmagazine.com/2015/08/understanding-critical-css/)
- Inlining critical CSS for better web performance | Go Make Things (https://gomakethings.com/inlining-critical-css-for-better-web-performance/)
- Reduce the size of the above-the-fold content (https://developers.google.com/speed/docs/insights/PrioritizeVisibleContent)

COMPONENTS OF CHECKLIST - CSS

Embedded or inline CSS (CSS nội dòng)

➢Avoid using embed or inline CSS inside your <body> (Not valid for HTTP/2)

Why:

One of the first reason it's because it's a good practice to separate content from design. It also helps you have a more maintainable code and keep your site accessible. But regarding performance, it's simply because it decreases the file-size of your HTML pages and the load time.

How:

Always use external stylesheets or embed CSS in your <head> (and follow the others CSS performance rules).

COMPONENTS OF CHECKLIST - CSS

Embedded or inline CSS

➢Article:

- Observe CSS Best Practices: Avoid CSS Inline Styles (https://www.lifewire.com/avoid-inline-styles-for-css-3466846)

COMPONENTS OF CHECKLIST - CSS

Analyse stylesheets complexity (phân tách stylesheet phức tạp)

➢Analyzing your stylesheets can help you to flag issues, redundancies and duplicate CSS selectors.

Why:

Sometimes you may have redundancies or validation errors in your CSS, analysing your CSS files and removed these complexities can help you to speed up your CSS files (because your browser will read them faster)

How:

Your CSS should be organized, using a CSS preprocessor can help you with that. Some online tools listed below can also help you analysing and correct your code.

COMPONENTS OF CHECKLIST - CSS

Analyse stylesheets complexity

➢Tool:

- TestMyCSS | Optimize and Check CSS Performance (http://www.testmycss.com/)

- CSS Stats (https://cssstats.com/)
- macbre/analyze-css: CSS selectors complexity and performance analyzer (https://github.com/macbre/analyze-css)
- Project Wallace (https://www.projectwallace.com/))

COMPONENTS OF CHECKLIST

Fonts

COMPONENTS OF CHECKLIST - Fonts

Webfont formats (định dạng font)

You are using WOFF2 on your web project or application.

Why:

According to Google, the WOFF 2.0 Web Font compression format offers 30% average gain over WOFF 1.0. It's then good to use WOFF 2.0, WOFF 1.0 as a fallback and TTF.

How:

Check before buying your new font that the provider gives you the WOFF2 format. If you are using a free font, you can always use Font Squirrel to generate all the formats you need.

COMPONENTS OF CHECKLIST - Fonts

Webfont formats

➢ Tool:

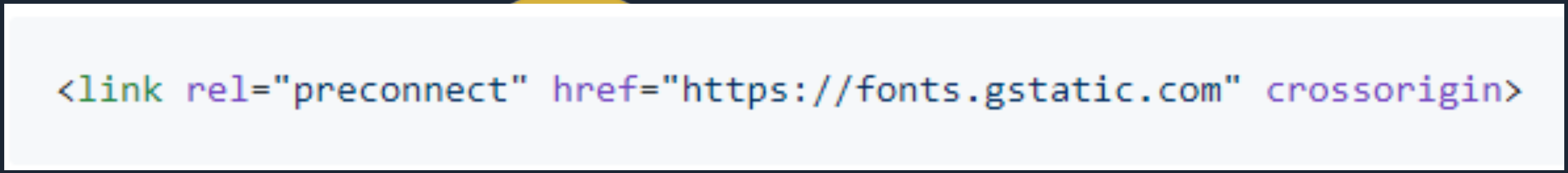
- Create Your Own @font-face Kits » Font Squirrel (https://gist.github.com/sergejmueller/cf6b4f2133bcb3e2f64a)
- IcoMoon App - Icon Font, SVG, PDF & PNG Generator (https://www.fontsquirrel.com/tools/webfont-generator)

➢ Article:

- WOFF 2.0 – Learn more about the next generation Web Font Format and convert TTF to WOFF2 (https://abookapart.com/products/webfont-handbook)
- Using @font-face | CSS-Tricks (https://icomoon.io/app/)
- Can I use... WOFF2 (https://caniuse.com/#feat=woff2)

COMPONENTS OF CHECKLIST - Fonts

Use preconnect (giá trị preconnect) to load your fonts faster



```
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
```

COMPONENTS OF CHECKLIST - Fonts

Use preconnect to load your fonts faster

Why:

When you arrived on a website, your device needs to find out where your site lives and which server it needs to connect with. Your browser had to contact a DNS server and wait for the lookup complete before fetching the resource (fonts, CSS files...).

Prefetches and preconnects allow the browser to lookup the DNS information and start establishing a TCP connection to the server hosting the font file. This provides a performance boost because by the time the browser gets around to parsing the css file with the font information and discovering it needs to request a font file from the server, it will already have pre-resolved the DNS information and have an open connection to the server ready in its connection pool.

COMPONENTS OF CHECKLIST - Fonts

Use preconnect to load your fonts faster

How:

Check before buying your new font that the provider gives you the WOFF2 format. If you are using a free font, you can always use Font Squirrel to generate all the formats you need.

COMPONENTS OF CHECKLIST - Fonts

Webfont formats

➢ Tool:

- typekit/webfontloader: Web Font Loader gives you added control when using linked fonts via @font-face. (https://github.com/typekit/webfontloader)

➢ Article:

- Faster Google Fonts with Preconnect - CDN Planet (https://www.cdnplanet.com/blog/faster-google-webfonts-preconnect/)

- Make Your Site Faster with Preconnect Hints | Viget (https://www.viget.com/articles/make-your-site-faster-with-preconnect-hints/)

- Ultimate Guide to Browser Hints: Preload, Prefetch, and Preconnect - MachMetrics Speed Blog (https://www.machmetrics.com/speed-blog/guide-to-browser-hints-preload-preconnect-prefetch/)

- A Comprehensive Guide to Font Loading Strategies—zachleat.com (https://www.zachleat.com/web/comprehensive-webfonts/#font-face)

COMPONENTS OF CHECKLIST - Fonts

Webfont size (kích cỡ font)

Webfont sizes don't exceed 300kb (all variants included)

➢Article:

- Font Bytes - Page Weight (https://httparchive.org/reports/page-weight#bytesFont)

COMPONENTS OF CHECKLIST - Fonts

Prevent Flash or Invisible Text
(tránh sử dụng text ẩn)

Avoid transparent text until the Webfont is loaded.

➢Article:

- font-display for the Masses (https://css-tricks.com/font-display-masses/)
- CSS font-display: The Future of Font Rendering on the Web (https://www.sitepoint.com/css-font-display-future-font-rendering-web/)

COMPONENTS OF CHECKLIST

Images

COMPONENTS OF CHECKLIST - Images

Images optimization (tối ưu hình ảnh)

Your images are optimized, compressed without direct impact to the end user..

Why:

Optimized images load faster in your browser and consume less data.

How:

- Try using CSS3 effects when it's possible (instead of a small image)
- When it's possible, use fonts instead of text encoded in your images
- Use SVG
- Use a tool and specify a level compression under 85.

COMPONENTS OF CHECKLIST - Images

Images optimization

➢Article:

- Image Optimization | Web Fundamentals | Google Developers (https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/image-optimization)
- Essential Image Optimization - An eBook by Addy Osmani (https://images.guide/)
- TinyJPG – Compress JPEG images intelligently (https://tinyjpg.com/)
- Kraken.io - Online Image Optimizer (https://kraken.io/web-interface)
- Compressor.io - optimize and compress JPEG photos and PNG images (https://compressor.io/compress)
- Cloudinary - Image Analysis Tool (https://webspeedtest.cloudinary.com/)
- SVGOMG - Optimize SVG vector graphics files(https://jakearchibald.github.io/svgomg/)

COMPONENTS OF CHECKLIST - Images

Images format (định dạng ảnh)

Choose your image format appropriately.

Why:

To ensure that your images don't slow your website, choose the format that will correspond to your image. If it's a photo, JPEG is most of the time more appropriate than PNG or GIF. But don't forget to look a the nex-gen formats which can reduce the size of your files. Each image format has pros and cons, it's important to know these to make the best choice possible.

How:

- Use Lighthouse to identify which images can eventually use next-gen formats (like JPEG 2000m JPEG XR or WebP)

- Compare different formats, sometimes using PNG8 is better than PNG16, sometimes it's not.

COMPONENTS OF CHECKLIST - Images

Images format

Article:

- Serve Images in Next-Gen Formats | Tools for Web Developers | Google Developers (https://developers.google.com/web/tools/lighthouse/audits/webp)
- What Is the Right Image Format for Your Website? — SitePoint (https://www.sitepoint.com/what-is-the-right-image-format-for-your-website/)
- PNG8 - The Clear Winner — SitePoint (https://www.sitepoint.com/png8-the-clear-winner/)
- 8-bit vs 16-bit - What Color Depth You Should Use And Why It Matters - DIY Photography (https://www.diyphotography.net/8-bit-vs-16-bit-color-depth-use-matters/)

COMPONENTS OF CHECKLIST - Images

Use vector image vs raster/bitmap
(sử dụng ảnh vector so với raster/bitmap)

Prefer using vector image rather than bitmap images (when possible).

Why:

Vector images (SVG) tend to be smaller than images and SVG's are responsive and scale perfectly. These images can be animated and modified by CSS.

COMPONENTS OF CHECKLIST - Images

Images dimensions
(kích thước hình ảnh)

Set width and height attributes on <img> if the final rendered image size is known.

Why:

If height and width are set, the space required for the image is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the image, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the images load).

COMPONENTS OF CHECKLIST - Images

Avoid using Base64 images
(tránh sử dụng ảnh Base64)

Base64 encoded images can be embedded using img tags or CSS, speeding up load times for smaller images by preventing additional HTTP requests.

This can be done to build single-file mockups / demo pages for your clients, HTML email signatures that will not trigger the nasty "show images" warning in email clients, etc.

You could eventually convert tiny images to base64 but it's actually not the best practice.

COMPONENTS OF CHECKLIST - Images

Avoid using Base64 images (tránh sử dụng ảnh Base64)

Article:

- Base64 Encoding & Performance, Part 1 and 2 by Harry Roberts (https://csswizardry.com/2017/02/base64-encoding-and-performance/)
- A closer look at Base64 image performance – The Page Not Found Blog (http://www.andygup.net/a-closer-look-at-base64-image-performance/)
- When to base64 encode images (and when not to) | David Calhoun (https://www.davidbcalhoun.com/2011/when-to-base64-encode-images-and-when-not-to/)
- Base64 encoding images for faster pages | Performance and seo factors (https://varvy.com/pagespeed/base64-images.html)

COMPONENTS OF CHECKLIST - Images

Lazy loading (tải lười biếng)

Lazy loading (also called on-demand loading) is an optimization technique for the online content, be it a website or a web app.

Instead of loading the entire web page and rendering it to the user in one go as in bulk loading, the concept of lazy loading assists in loading only the required section and delays the remaining, until it is needed by the user.

Offscreen images are loaded lazily (A noscript fallback is always provided).

COMPONENTS OF CHECKLIST - Images

Lazy loading

Why:

It will improve the response time of the current page and then avoid loading unnecessary images that the user may not need.

How:

- Use Lighthouse to identify how many images are offscreen.
- Use a JavaScript plugin like the following to lazyload your images. Make sure you target offscreen images only.
- Also make sure to lazyload alternative images shown at mouseover or upon other user actions.

COMPONENTS OF CHECKLIST - Images

Lazy loading

Tool:

- verlok/lazyload: GitHub (https://github.com/verlok/lazyload)
- aFarkas/lazysizes: GitHub (https://github.com/aFarkas/lazysizes/)

Article:

- Lazy Loading Images and Video | Web Fundamentals | Google Developers (https://developers.google.com/web/fundamentals/performance/lazy-loading-guidance/images-and-video/)
- 5 Brilliant Ways to Lazy Load Images For Faster Page Loads - Dynamic Drive Blog (http://blog.dynamicdrive.com/5-brilliant-ways-to-lazy-load-images-for-faster-page-loads/)

COMPONENTS OF CHECKLIST - Images

Responsive images

Ensure to serve images that are close to your display size.

Why:

Small devices don't need images bigger than their viewport. It's recommended to have multiple versions of one image on different sizes.

How:

- Create different image sizes for the devices you want to target.
- Use srcset and picture to deliver multiple variants of each image.

COMPONENTS OF CHECKLIST - Images

Responsive images

Article:

- Responsive images - Learn web development | MDN (https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images)

COMPONENTS OF CHECKLIST

JavaScript (JS)

COMPONENTS OF CHECKLIST - JS

JS Minification (tinh giảm JS)

All JavaScript files are minified, comments, white spaces and new lines are removed from production files.

Why:

Removing all unnecessary spaces, comments and break will reduce the size of your JavaScript files and speed up your site's page load times and obviously lighten the download for your user

How:

- Use the tools suggested below to minify your files automatically before or during your build or your deployment.

COMPONENTS OF CHECKLIST - JS

JS Minification

Tools:

- uglify-js – npm (https://www.npmjs.com/package/uglify-js)
- Online JavaScript Compressor (http://refresh-sf.com/)

Article:

- Short read: How is HTTP/2 different? Should we still minify and concatenate? (https://scaleyourcode.com/blog/article/28)

COMPONENTS OF CHECKLIST - JS

No JavaScript inside
(không có JS bên trong nội dung)

(Only valid for website) Avoid having multiple JavaScript codes embedded in the middle of your body. Regroup your JavaScript code inside external files or eventually in the <head> or at the end of your page (before </body>).

Why:

Placing JavaScript embedded code directly in your <body> can slow down your page because it loads while the DOM is being built. The best option is to use external files with async or defer to avoid blocking the DOM. Another option is to place some scripts inside your <head>. Most of the time analytics code or small script that need to load before the DOM gets to main processing.

COMPONENTS OF CHECKLIST - JS

No JavaScript inside

How:

Ensure that all your files are loaded using async or defer and decide wisely the code that you will need to inject in your <head>.

Article:

- 11 Tips to Optimize JavaScript and Improve Website Loading Speeds (https://www.upwork.com/hiring/development/11-tips-to-optimize-javascript-and-improve-website-loading-speeds/)

COMPONENTS OF CHECKLIST - JS

No JavaScript inside

Tools:

- uglify-js – npm (https://www.npmjs.com/package/uglify-js)
- Online JavaScript Compressor (http://refresh-sf.com/)

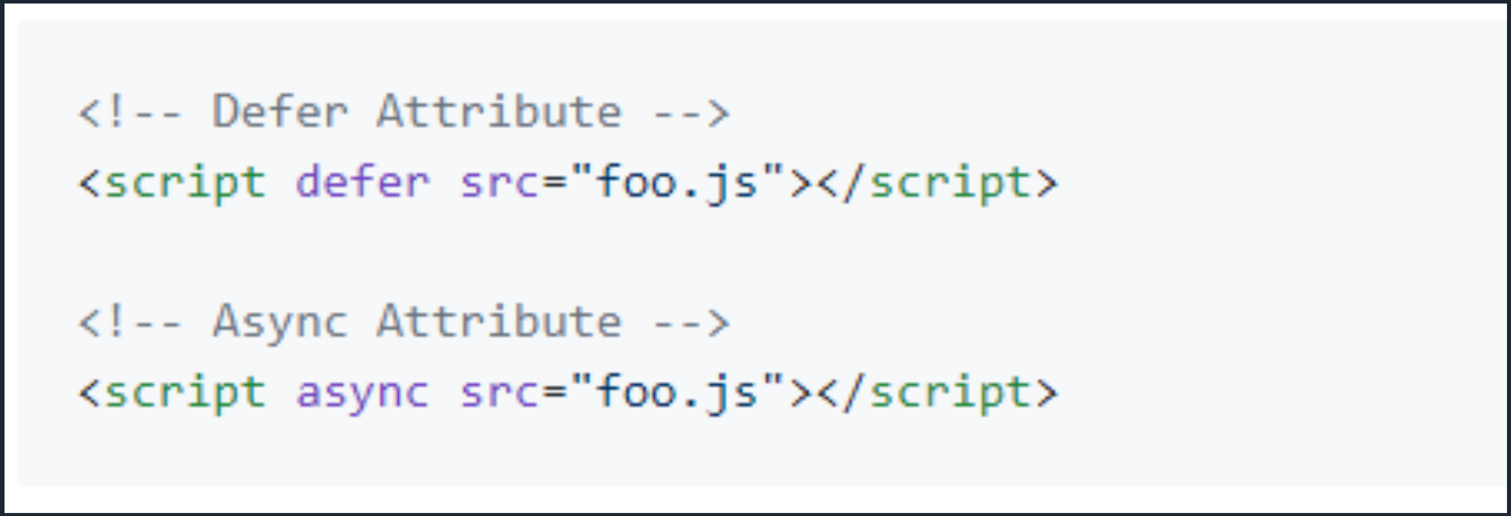
Article:

- Short read: How is HTTP/2 different? Should we still minify and concatenate? (https://scaleyourcode.com/blog/article/28)

COMPONENTS OF CHECKLIST - JS

Non-blocking JavaScript (không chặn JS)

JavaScript files are loaded asynchronously using async or deferred using defer attribute.



```
<!-- Defer Attribute -->
<script defer src="foo.js"></script>

<!-- Async Attribute -->
<script async src="foo.js"></script>
```

COMPONENTS OF CHECKLIST - JS

Non-blocking JavaScript

Why:

JavaScript blocks the normal parsing of the HTML document, so when the parser reaches a <script> tag (particularly is inside the <head>), it stops to fetch and run it. Adding async or defer are highly recommended if your scripts are placed in the top of your page but less valuable if just before your </body> tag. But it's a good practice to always use these attributes to avoid any performance issue.

How:

- Add async (if the script don't rely on other scripts) or defer (if the script relies upon or relied upon by an async script) as an attribute to your script tag.

- If you have small scripts, maybe use inline script place above async scripts.

COMPONENTS OF CHECKLIST - JS

Non-blocking JavaScript

Article:

- Remove Render-Blocking JavaScript (https://developers.google.com/speed/docs/insights/BlockingJS)
- Defer loading JavaScript (https://varvy.com/pagespeed/defer-loading-javascript.html)

COMPONENTS OF CHECKLIST - JS

Optimized and updated JS libraries (tối ưu và cập nhật các thư viện JS)

All JavaScript libraries used in your project are necessary (prefer Vanilla JavaScript for simple functionalities), updated to their latest version and don't overwhelm your JavaScript with unnecessary methods.

Why:

Most of the time, new versions come with optimization and security fix. You should use the most optimized code to speed up your project and ensure that you'll not slow down your website or app without outdated plugin.

How:

If your project use NPM packages, npm-check is a pretty interesting library to upgrade / update your libraries. Greenkeeper can automatically look for your dependencies and suggest an update evey time a new version is out.

COMPONENTS OF CHECKLIST - JS

Optimized and updated JS libraries

Article:

- You may not need jQuery (http://youmightnotneedjquery.com/)
- Vanilla JavaScript for building powerful web applications (https://plainjs.com/)

COMPONENTS OF CHECKLIST - JS

Check dependencies size limit
(kiểm tra giới hạn kích thước các thư viện)

Ensure to use wisely external libraries, most of the time, you can use a lighter library for a same functionality.

Why:

You may be tempted to use one of the 745 000 packages you can find on npm, but you need to choose the best package for your needs. For example, MomentJS is an awesome library but with a lot of methods you may never use, that's why Day.js was created. It's just 2kB vs 16.4kB gz for Moment.

How:

Always compare and choose the best and lighter library for your needs. You can also use tools like npm trends to compare NPM package downloads counts or Bundlephobia to know the size of your dependencies.

COMPONENTS OF CHECKLIST - JS

Check dependencies size limit

Tools:

- ai/size-limit: Prevent JS libraries bloat. If you accidentally add a massive dependency, Size Limit will throw an error. (https://github.com/ai/size-limit)
- webpack-bundle-analyzer – npm (https://www.npmjs.com/package/webpack-bundle-analyzer)

Article:

- Size Limit: Make the Web lighter — Martian Chronicles, Evil Martians' team blog (https://evilmartians.com/chronicles/size-limit-make-the-web-lighter)

COMPONENTS OF CHECKLIST - JS

JavaScript Profiling
(sắp xếp JavaScript)

Check for performance problems in your JavaScript files (and CSS too).

Why:

JavaScript complexity can slow down runtime performance. Identifying these possible issues are essential to offer the smoothest user experience.

How:

Use the Timeline tool in the Chrome Developer Tool to evaluate scripts events and found the one that may take too much time.

COMPONENTS OF CHECKLIST - JS

JavaScript Profiling

Article:

- Speed Up JavaScript Execution | Tools for Web Developers | Google Developers (https://developers.google.com/web/tools/chrome-devtools/rendering-tools/js-execution)
- JavaScript Profiling With The Chrome Developer Tools — Smashing Magazine (https://www.smashingmagazine.com/2012/06/javascript-profiling-chrome-developer-tools/)
- How to Record Heap Snapshots | Tools for Web Developers | Google Developers (https://developers.google.com/web/tools/chrome-devtools/memory-problems/heap-snapshots)
- Chapter 22 - Profiling the Frontend – Blackfire (https://blackfire.io/docs/book/22-frontend-profiling)
- 30 Tips To Improve Javascript Performance (https://www.monitis.com/blog/30-tips-to-improve-javascript-performance/)

COMPONENTS OF CHECKLIST - JS

Use of Service Workers
(sử dụng Service Workers)

A service worker is a script that your browser runs in the background, separate from a web page, opening the door to features that don't need a web page or user interaction.

You are using Service Workers in your PWA to cache data or execute possible heavy tasks without impacting the user experience of your application.

COMPONENTS OF CHECKLIST - JS

Use of Service Workers

Article:

- Service Workers: an Introduction | Web Fundamentals | Google Developers (https://developers.google.com/web/fundamentals/primers/service-workers/)
- Measuring the Real-world Performance Impact of Service Workers | Web | Google Developers (https://developers.google.com/web/showcase/2016/service-worker-perf)
- What Are Service Workers and How They Help Improve Performance (https://www.keycdn.com/blog/service-workers)
- How does a service worker work? – YouTube (https://www.youtube.com/watch?v=__xAtWgfzvc)

COMPONENTS OF CHECKLIST

Server

COMPONENTS OF CHECKLIST – Server

Your website is using HTTPS

Why:

HTTPS is not only for ecommerce websites, but for all websites that are exchanging data. Data shared by a user or data shared to an external entity. Modern browsers today limit functionalities for sites that are not secure. For example: geolocation, push notifications and service workers don't work if your instance is not using HTTPS. And today is much more easy to setup a project with an SSL certificate than it was before (and for free, thanks to Let's Encrypt).

COMPONENTS OF CHECKLIST – Server

Your website is using HTTPS

Article:

- Why Use HTTPS? | Cloudflare (https://www.cloudflare.com/learning/ssl/why-use-https/)
- Enabling HTTPS Without Sacrificing Your Web Performance – Moz (https://moz.com/blog/enabling-https-without-sacrificing-web-performance)
- How HTTPS Affects Website Performance (https://wp-rocket.me/blog/https-affects-website-performance/)
- HTTP versus HTTPS versus HTTP2 - The real story | Tune The Web (https://www.tunetheweb.com/blog/http-versus-https-versus-http2/)
- HTTP vs HTTPS — Test them both yourself (https://www.httpvshttps.com/)

COMPONENTS OF CHECKLIST – Server

Page weight < 1500 KB (ideally < 500 KB)

Reduce the size of your page + resources as much as you can.

Why:

Ideally you should try to target < 500 KB but the state of web shows that the median of Kilobytes is around 1500 KB (even on mobile). Depending on your target users, network connection, devices, it's important to reduce as much as possible your total Kilobytes to have the best user experience possible.

How:

- All the rules inside the Front-End Performance Checklist will help you to reduce as much as possible your resources and your code.

COMPONENTS OF CHECKLIST – Server

Page weight < 1500 KB (ideally < 500 KB)
(dung lượng page < 1500KB – lý tưởng < 500KB)

Tool:

- What Does My Site Cost? (https://httparchive.org/reports/page-weight#bytesTotal)
- web - Measure full page size in Chrome DevTools - Stack Overflow (https://whatdoesmysitecost.com/)

Article:

- Page Weight (https://stackoverflow.com/questions/38239980/measure-full-page-size-in-chrome-devtools)

COMPONENTS OF CHECKLIST – Server

Page load times < 3 seconds
(thời gian load page < 3 giây)

Reduce as much as possible your page load times to quickly deliver your content to your users.

Why:

Faster your website or app is, less you have probability of bounce increases, in other terms you have less chances to lose your user or future client. Enough researches on the subject prove that point.

How:

Use online tools like Page Speed Insight or WebPageTest to analyze what could be slowing you down and use the Front-End Performance Checklist to improve your load times.

COMPONENTS OF CHECKLIST – Server

Page load times < 3 seconds

Tool:

- Compare your mobile site speed (https://www.thinkwithgoogle.com/feature/testmysite)
- Test Your Mobile Website Speed and Performance - Think With Google (https://www.thinkwithgoogle.com/feature/testmysite)

Article:

- Average Page Load Times for 2018 - How does yours compare? - MachMetrics Speed Blog (https://www.machmetrics.com/speed-blog/average-page-load-times-websites-2018/)

COMPONENTS OF CHECKLIST – Server

Time To First Byte < 1.3 seconds

(Thời gian cho byte đầu tiên < 1.3 giây)

Reduce as much as you can the time your browser waits before receiving data.

Tool:

- Global latency testing tool (https://latency.apex.sh/)

Article:

- What is Waiting (TTFB) in DevTools, and what to do about it (https://scaleyourcode.com/blog/article/27)
- Monitoring your servers with free tools is easy (https://scaleyourcode.com/blog/article/7)
- Time to First Byte (TTFB) (https://varvy.com/pagespeed/ttfb.html)

COMPONENTS OF CHECKLIST – Server

Cookie size

(kích thước Cookie)

If you are using cookies, be sure each cookie doesn't exceed 4096 bytes and your domain name doesn't have more than 20 cookies.

Why:

Cookies are exchanged in the HTTP headers between web servers and browsers. It's important to keep the size of cookies as low as possible to minimize the impact on the user's response time.

How:

Eliminate unnecessary cookies.

COMPONENTS OF CHECKLIST – Server

Cookie size

Tool:

- Browser Cookie Limits (http://browsercookielimits.squawky.net/)

Article:

- Cookie specification: RFC 6265 (https://tools.ietf.org/html/rfc6265)
- Cookies (https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies)
- Website Performance: Cookies Don't Taste So Good - Monitis Blog (https://www.monitis.com/blog/website-performance-cookies-dont-taste-so-good/)
- Google's Web Performance Best Practices #3: Minimize Request Overhead - GlobalDots Blog (https://www.globaldots.com/googles-web-performance-best-practices-3-minimize-request-overhead/)

COMPONENTS OF CHECKLIST – Server

Minimizing HTTP requests (giảm thiểu request HTTP)

Always ensure that every file requested are essential for your website or application.

Article:

- Combine external CSS (https://varvy.com/pagespeed/combine-external-css.html)
- Combine external JavaScript (https://varvy.com/pagespeed/combine-external-javascript.html)

COMPONENTS OF CHECKLIST – Server

Use a CDN to deliver your assets
(Sử dụng CDN để phân phối nội dung)

Use a CDN to deliver faster your content over the world

Article:

- 10 Tips to Optimize CDN Performance - CDN Planet (https://www.cdnplanet.com/blog/10-tips-optimize-cdn-performance/)
- HTTP Caching | Web Fundamentals | Google Developers (https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching)

COMPONENTS OF CHECKLIST – Server

Serve files from the same protocol
(Phân phối tệp từ cùng một giao thức)

Avoid having your website serving files coming from source using HTTP on your website which is using HTTPS for example. If your website is using HTTPS, external files should come from the same protocol.

COMPONENTS OF CHECKLIST – Server

Serve reachable files

Avoid requesting unreachable files (404).

Article:

- How to avoid bad requests (https://varvy.com/pagespeed/avoid-bad-requests.html)

COMPONENTS OF CHECKLIST – Server

Set HTTP cache headers properly
(Đặt tiêu đề bộ nhớ cache HTTP đúng cách)

Set HTTP headers to avoid expensive number of roundtrips between your browser and the server.

Article:

- Using cache-control for browser caching (https://varvy.com/pagespeed/cache-control.html)

COMPONENTS OF CHECKLIST – Server

GZIP / Brotli compression is enabled
(bật chức năng nén Brotli/GZIP)

Brotli is a new compression algorithm optimized for the web, in particular small text documents. Brotli decompression is at least as fast as for gzip while significantly improving the compression ratio.

Use a compression method such as Gzip or Brotli to reduce the size of your JavaScript files. With a smaller sizes file, users will be able to download the asset faster, resulting in improved performance..

Tools:

- Check GZIP compression (https://checkgzipcompression.com/)
- Check Brotli Compression (https://tools.keycdn.com/brotli-test)

Article:

- Can I use... Brotli (https://caniuse.com/#feat=brotli)

PERFORMANCE TOOLS

PERFORMANCE TOOLS

List of the tools you can use to test or monitor your website or application:

https://www.webpagetest.org/
https://www.dareboost.com/en
https://treo.sh/?ref=perfchecklist
https://gtmetrix.com/
https://developers.google.com/speed/pagespeed/insights/
https://web.dev/measure
https://tools.pingdom.com/
https://developers.google.com/speed/
https://www.sitespeed.io/
https://calibreapp.com/
https://www.dotcom-tools.com/website-speed-test.aspx
https://www.pingdom.com/product/uptime-monitoring/
https://uptimerobot.com/
https://speedcurve.com/
https://varvy.com/pagespeed/
https://developers.google.com/web/tools/lighthouse/#devtools
https://www.checkbot.io/
https://yellowlab.tools/
https://speedrank.app/en
https://www.debugbear.com/

REFERRENCES

REFERRENCES

https://www.youtube.com/watch?v=_bzqF05xsC4
https://addyosmani.com/blog/performance-budgets/
https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/
https://httparchive.org/reports/state-of-the-web?start=2018_01_01
https://www.speedshop.co/2015/11/05/page-weight-doesnt-matter.html
https://www.smashingmagazine.com/2019/01/front-end-performance-checklist-2019-pdf-pages/
http://designingforperformance.com/index.html
https://varvy.com/performance/
https://github.com/fabkrum/web-performance-resources
https://www.checkbot.io/guide/speed/
https://progressivetooling.com/

CYBERSOFT ACADEMY

http://cybersoft.edu.vn

Thank You !

CYBERSOFT

ĐÀO TẠO CHUYÊN GIA LẬP TRÌNH