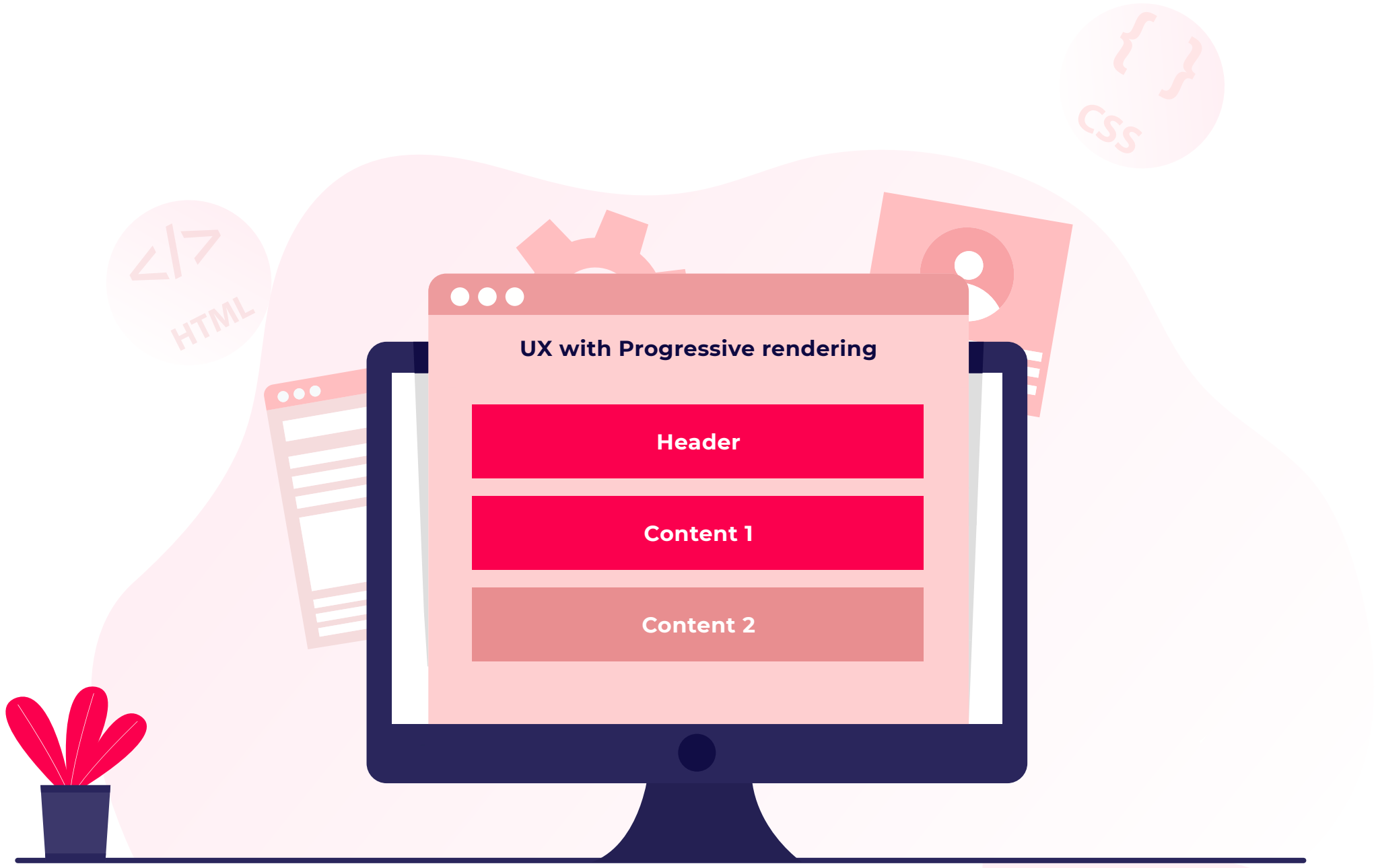
QBurst

# Frontend Optimization

## Tips for Improving the Performance of Single Page Applications



## 01. Module-Wise Page Loading

Lazy loading helps keep the initial bundle size smaller and improve load time. Split the CSS, JS files into module-specific chunk files. Prioritize above-the-fold content and load the minimum HTML, CSS, and JavaScript to quickly display the visible portion of the page to users while deferring the render of remaining resources.

## 02. Mobile-First CSS

Apply CSS styles first to mobile devices. You can add advanced styles and overrides for larger screens into the stylesheet using media queries. As mobile styles are less complicated than those for desktop, a mobile-first approach helps simplify your CSS code.

## 03. JS Optimization

Splitting the JavaScript code and bundling it into different modules is a sure step toward reducing JS payload. Analyze these bundles to detect and remove unused libraries. By minifying JS and loading only the most important resources first, initial page load time can be reduced. Third-party scripts such as for analytics can be delayed and loaded later.

## 04. Optimizing Images

When using images, choose the optimal format. SVG, PNG, JPEG, font icons—all have their own use case. Upload compressed images and set limits for image size without compromising quality. Prioritized and page-specific image loading is another way to reduce loading delays. By loading low-quality images first and then replacing them with better versions as the page continues to load, you can progressively load images to speed up the page.

## 05. Caching

The browser's HTTP cache is an effective approach to loading resources faster avoiding unnecessary network requests. However, to avoid stale content from being served to users from the cache, you could implement filename or file path versioning.

## 06. Reusable CSS

As per the latest SPA component structure, you can define CSS for global purposes or a particular section. If a style is to be applied throughout the app, you can write it as reusable CSS and avoid multiple declarations. This will help reduce the overall CSS bundle size.

## 07. HTTP/2 Connection

Make sure all the assets are served over HTTP/2. If you have to, migrate to HTTPS first and then switch to HTTP/2 protocol that allows for multiplexing and prioritization.

## 08. API Cache

API caching is a mechanism to store and retrieve data from the cache rather than hit the backend server on each network request. This can avoid the delay in request-response and improve performance.

## 09. Pagination of Data

Pagination is a process to divide the response data based on the application required to process it. You need not fetch all the data at a time. This will help reduce the entire API size, rendering and parsing data in smaller chunks.

## 10. Compressing Files

Reducing file size using compression algorithms can cause a significant improvement in page performance. Tools such as gzip are supported by most of the popular browsers and can be used without worry.

QBurst

USA | UK | UAE | INDIA | SINGAPORE | AUSTRALIA | JAPAN

www.qburst.com | info@qburst.com