Performance Budgets

The idea of a performance budget is to set the performance goals for the site.

What is a performance budget?

There are multiple definitions of what's a performance budget. According to Performance budgets 101:

> A performance budget is a set of limits imposed on metrics that affect site performance. This could be the total size of a page, the time it takes to load on a mobile network, or even the number of HTTP requests that are sent.

From Performance Budgets That Stick:

> A performance budget is a clearly defined limit on one or more performance metrics that the team agrees not to exceed, and that is used to guide design and development.

Defining a performance budget

There are different types of performance budgets (From Start Performance Budgeting):

Milestone timings: : Timings based on the user-experience loading a page (e.g Time-to-Interactive, First Contentful Paint) : You'll often want to pair several milestone timings to accurately represent the complete story during page load : Some teams also maintain custom metrics, like "Time to first tweet" for Twitter.

Quantity-based metrics: : based on raw values (e.g. weight of JavaScript (KB/MB), number of HTTP requests). : These are focused on the browser experience.

Rule-based metrics: : Scores generated by tools such as Lighthouse or WebPageTest : Often, a single number or series to grade your site.

Some examples of a performance budget:

- The home page load and get interactive in under 5 seconds in a 3G mobile connection, and under 2 seconds for subsequent loads
- Our product page must ship less than 170KB of JavaScript on mobile
- Our search page must include less than 1MB of images on desktop
- Our blog must score > 80 on Lighthouse performance audits in mobile and > 90 on desktop
- An individual blog post must make less than 25 HTTP requests

implementing a performance budget

The best way to implement a performance budget is to use a tool like WebPageTest or Lighthouse to generate a base score for your site.

The team can the decide on what makes the most sense based on your clients' location, the devices they use, and the type of content that you're serving.

We'll use some of the example budget goalds to get started in a performance budget

- The site must score > 80 on Lighthouse performance audits in mobile and > 90 on desktop
- An individual page's Largest Contentful Paint (LCP) loads in under 2.5 seconds on desktop and 3 seconds in mobile
- The home page time to interactive (TTI) is under 5 seconds in a 3G mobile connection, and under 2 seconds for subsequent loads

Keeping the development team honest

Knowing what our targets are is one thing, to actually hit it and stay there is a different task altogether.

During development, there are several tools we can use to run checks against new or modified assets:

- Webpack has features that will notify you or cause build errors when assets exceed specified limits at build/bundle time

- Bundlesize, allows you to define and run file size checks in your Continous Integration (CI) pipeline
- Size Limit Github Action will add a comment to all new pull requests listing the size changes of the files involved in the PR
- If you have a paid WebPageTest account, you can use their Github Action to run WebPageTest against all new commits and pull requests in your repository
- Lighthouse also provides a Lighthouse CI Github Action to run Lighthouse against all new Pull Requests in the project's repository

We did all this, now what?

First, we agreed to what our perfomance budget would look like. Then we measure to make sure we stay within the budget that we set up.

Why go through all the trouble?

The idea behind performance budgets is to improve application's performance for all users. We constrain what we do as developers in order to improve the user experience overall.

Performance Budgets also allow the team to make informed decisions on changes to the codebase. If we add something that makes us go over budget then what do we take out? If it's a new feature how much does the feature adds to the budget? But, to me, the biggest question is: Can we optimize our code and assets so that new features don't go over budget?

Links and resources

- Performance budgets — MDN
- Performance Budgets 101 — web.dev
- Performance Budgets — Addy Osmani
- Web Performance Budget Tracking Using Lighthouse — Halodoc
- Webpack Performance Configuration