Render Blocking CSS

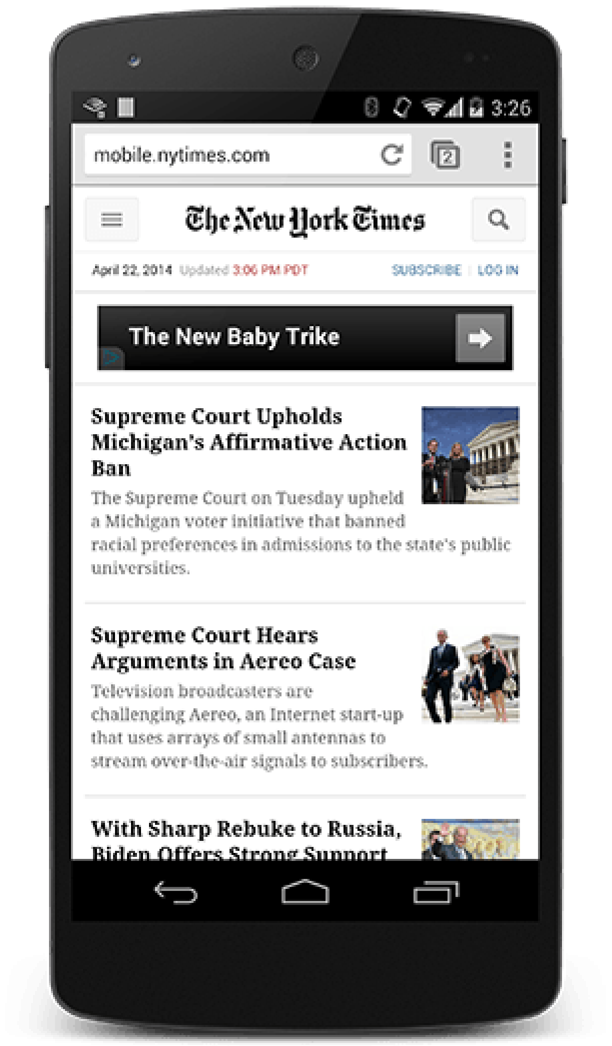
By Ilya Grigorik
Ilya is a Developer Advocate and Web Perf Guru

By default, CSS is treated as a render blocking resource, which means that the browser won't render any processed content until the CSSOM is constructed. Make sure to keep your CSS lean, deliver it as quickly as possible, and use media types and queries to unblock rendering.

In the render tree construction we saw that the critical rendering path requires both the DOM and the CSSOM to construct the render tree. This creates an important performance implication: **both HTML and CSS are render blocking resources.** The HTML is obvious, since without the DOM we would not have anything to render, but the CSS requirement may be less obvious. What would happen if we try to render a typical page without blocking rendering on CSS?

TL;DR

- By default, CSS is treated as a render blocking resource.
- Media types and media queries allow us to mark some CSS resources as non-render blocking.
- The browser downloads all CSS resources, regardless of blocking or non-blocking behavior.



April 22, 2014 Updated 3:06 PM PDT SUBSCRIBE | LOG IN

The New Baby Trike

Supreme Court Upholds Michigan's Affirmative Action Ban

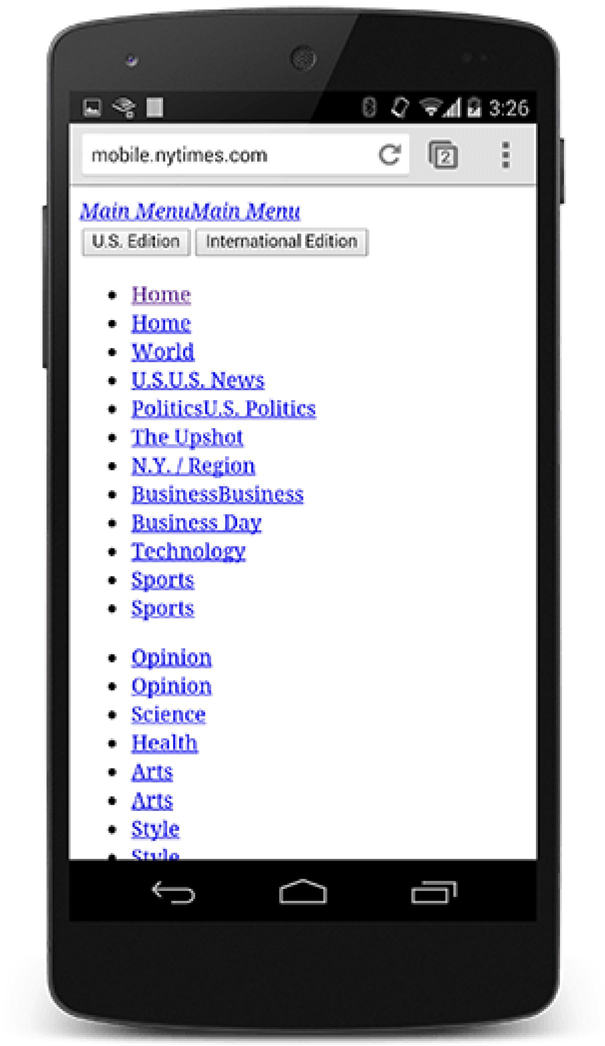
The Supreme Court on Tuesday upheld a Michigan voter initiative that banned racial preferences in admissions to the state's public universities.

Supreme Court Hears Arguments in Aereo Case

Television broadcasters are challenging Aereo, an Internet start-up that uses arrays of small antennas to stream over-the-air signals to subscribers.

With Sharp Rebuke to Russia, Biden Offers Strong Support

The New York Times with CSS



Main MenuMain Menu

U.S. Edition International Edition

- Home
- Home
- World
- U.S.U.S. News
- PoliticsU.S. Politics
- The Upshot
- N.Y. / Region
- BusinessBusiness
- Business Day
- Technology
- Sports
- Sports

- Opinion
- Opinion
- Science
- Health
- Arts
- Arts
- Style
- Style

The New York Times without CSS (FOUC)

The above example, showing the NYTimes website with and without CSS, demonstrates why rendering is blocked until CSS is available—without CSS the page is relatively unusable. The experience on the right is often referred to as a "Flash of Unstyled Content" (FOUC). The browser blocks rendering until it has both the DOM and the CSSOM.

> ***CSS is a render blocking resource. Get it to the client as soon and as quickly as possible to optimize the time to first render.***

However, what if we have some CSS styles that are only used under certain conditions, for example, when the page is being printed or being projected onto a large monitor? It would be nice if we didn't have to block rendering on these resources.

CSS "media types" and "media queries" allow us to address these use cases:

```
<link href="style.css" rel="stylesheet">
<link href="print.css" rel="stylesheet" media="print">
```

```
<link href="other.css" rel="stylesheet" media="(min-width: 40em)">
```

A media query consists of a media type and zero or more expressions that check for the conditions of particular media features. For example, our first stylesheet declaration doesn't provide a media type or query, so it applies in all cases; that is to say, it is always render blocking. On the other hand, the second stylesheet declaration applies only when the content is being printed—perhaps you want to rearrange the layout, change the fonts, and so on, and hence this stylesheet declaration doesn't need to block the rendering of the page when it is first loaded. Finally, the last stylesheet declaration provides a "media query," which is executed by the browser: if the conditions match, the browser blocks rendering until the style sheet is downloaded and processed.

By using media queries, we can tailor our presentation to specific use cases, such as display versus print, and also to dynamic conditions such as changes in screen orientation, resize events, and more. **When declaring your style sheet assets, pay close attention to the media type and queries; they greatly impact critical rendering path performance.**

Let's consider some hands-on examples:

```
<link href="style.css"    rel="stylesheet">
<link href="style.css"    rel="stylesheet" media="all">
<link href="portrait.css" rel="stylesheet" media="orientation:portrait">
<link href="print.css"    rel="stylesheet" media="print">
```

- The first declaration is render blocking and matches in all conditions.
- The second declaration is also render blocking: "all" is the default type so if you don't specify any type, it's implicitly set to "all". Hence, the first and second declarations are actually equivalent.
- The third declaration has a dynamic media query, which is evaluated when the page is loaded. Depending on the orientation of the device while the page is loading, portrait.css may or may not be render blocking.
- The last declaration is only applied when the page is being printed so it is not render blocking when the page is first loaded in the browser.

Finally, note that "render blocking" only refers to whether the browser has to hold the initial rendering of the page on that resource. In either case, the browser still downloads the CSS asset, albeit with a lower priority for non-blocking resources.

Previous
← Render-Tree Construction, Layout, and Paint

Next
Adding Interactivity with JavaScript
→