**Preload & images and how they affect the CRP**

**How do images affect the CRP and where does preload fit into all of this?**

This is a very advanced question that one of my students asked me (thanks Marc).

Its also very interesting, so I’ll try take time to explain how I understand the process to work.

**The painting process happens bit by bit**

Firstly, I want to reiterate that the paint process is a gradual one. It happens bit by bit. In fact, Google uses progressive loading as its better for a user to see large chunks of painted content than stare at a white blank screen for ages.

Many developers think of the CRP, and more specifically, the paint process, as one neat-and-tidy sequence of events.

But this isn’t how it works. Each process going on is basically an async thing, firing off whenever the browser thinks it has constructed enough of the DOM + CSSOM. This means there isn't a strict and ordered sequence. **Bottom line: the browser is not precluded from rendering an incomplete DOM/CSSOM tree.**

This means that the painting is decoupled from the entire HTML document, and for good reason. Consider a really large HTML document. If you've ever loaded one on a slower network, you'll know that it'll start showing elements as they are ready, not necessarily waiting for the full HTML file to load (this is progressive loading as I mentioned above).

Once the full HTML has been downloaded and parsed, that's when DOMContentLoaded is fired. A slow server could have actually taken quite a while to get to this point. For many documents, especially small ones or ones that load really quickly, it may be that DOMContentLoaded fires before the first paint. On a larger one, it'll probably have a few paints before the DOMContentLoaded event is fired.

If this is confusing you don’t stress. It is very advanced and in fact, exactly how a browser completes the entire CRP process is kept a secret and is what gives each browser an edge. The faster the paint time, the better service they offer. However, as we’ve seen in the performance and network tabs, browsers do give us clues as to what’s going on. They do this so that we can create our sites in the most optimal way. And in turn, this of course helps them look good. So, you could say it’s a win-win.

**How do images effect the time to first paint?**

The painting process involves … yep, painting pixels to the screen. This is specified by the what (Render Tree) and the how (Layout).

The most important thing that effects the CRP is whats known as above-the-fold content. This is good, because you can call a million things for your webpage (it can have 20,000 pictures and 500 JS files) and the page can still load in a second or so.

**Whaaat?**

That’s right. You can have 20,000 images on your page, but you probably only need one or two to be visible above the fold (critical). That means if we concentrate on those two images, we can make our initial page load time much better. We don't have to show 20,000 images on our first paint, we only need to show those one or two which are in the initial view.

You need to remember that images are not render blocking. We know this because often you’ll see the DOMContentLoaded  event fire before the image has loaded. This event is fired as soon as the DOM is ready. The image is downloaded later, which proves that images don’t block parsing – or even rendering, for that matter. They do, however, block the Load event (the red vertical line). This event signals that all resources the requested page needs have been downloaded and processed. That being said, image loading should of course be optimized. But images are not critical in terms of the rendering path.

**Where does preload come into the picture when we’re dealing with images and other resources?**

Let me give you a quick example. Lets say you have 2 images, ImageOne.png and ImageTwo.png. ImageOne.png is below the fold and ImageTwo.png is above the fold. You and I know, as developers, that the browser will need ImageTwo to render a first meaningful paint. Remember, the first word in CRP is “critical”, and this means content above the fold. So in this instance we should preload ImageTwo.png so that the browser has the image available when it constructs the first render tree. If the image is not available soon then the first conceptual paint may result in text on the screen and no image (we’ve all been there and seen this happen on slow internet connections).

And this is the purpose of preload. To tell the browser to fetch the resource early. Bottom line: it increases your chances of having a meaningful first paint. In other words, before the browser starts rendering the page and parsing the html it has already fetched the resource, so the resource is not blocking a meaningful first paint.

Makes sense?

**Other than images, where is preload used?**

Preload is used a lot with CSS. It works because it removes the stylesheet tag temporarily. We know that the stylesheet blocks rendering as the browser assigns it high priority and stops parsing the to fetch and parse it. So using preload with CSS allows you to asynchronously load CSS, not blocking the rendering of the rest of the page and most critically the above the fold, critical rendering part of the page.

Another common use for preload is with fonts. Fonts are probably the most important thing that gets hidden in many CSS files. They are critical for rendering the text on the page, but they don’t get loaded until browser is sure that they are going to be used. That check happens only after CSS has been parsed, and applied, and the browser has matched CSS rules to the DOM nodes. This happens fairly late in the CRP process and it often results in an unnecessary delay in text rendering.

The final point I want to make is that preload does not does not block the window’s onload event. This makes sense right, as the onload event or "Page Load" fires when the whole page and all of its dependent resources have finished loading. Page load often occurs later, after the point in time when the page is rendered and interactive for a user.

**Closing comments**

All modern browsers implement speculative parsing now. In a way this is like the browser automatically preloading resources. The browser does this for scripts, external CSS and images from the <img> tag.

The way to improve the render path is to remove or defer calls made that are not needed for the initial page view.

Most of the things you are using JavaScript for can probably be defered to load after the page view (things like social buttons, analytics, etc.)

Preload only tells your browser to fetch certain files early – files that you know are critical for a first meaningful paint to the screen. It does not affect the DOMContentLoaded or Load events.

Hope this short article has been useful.

Hang in there, and see you in the next lectures 😉