First Screen Paint In Advance

Abstract

This specification defines a means for site developers to determine whether to apply a render optimization policy which specifies painting content above the fold(we name it *first screen*) in advance.

Table of Contents

1. Introduction
	1. Render optimization policy with first screen paint in advance

Render optimization policy with first screen content painted in advance is used to help accelerate the displaying speed for the web content above the fold, especially on the mobile devices.

Browser’s processing for web pages mainly includes parse, layout and paint phases. As web pages are usually long and complicated with many external resources, browser will not wait all the web code and external resources downloaded and parsed to start layout and paint phase. Normally browser applies an incremental iterative policy, parses part of the tokens in web page, starts layout and paint, and then continues the processing loop for the remaining web code incrementally, until entire web page is handled completely. For the processing phase transitions from parse to layout or layout to paint, browser usually has some trigger conditions like the parsed token number, parsing time slice etc.

For mobile devices, usually a short code snippet and limited resources can decide the displaying content above the fold, and this part of content is just what end-users actually see and feel first. In current browsers render implementation, the scheduling policy for phase transition doesn’t take full consideration for the size of mobile devices screen, and the importance of rendering first user perceived content with highest priority. In complex mobile networking environment, this kind of policy will have larger performance limitation on browsing speed.

To provide a better user experience on mobile devices, browser can adjust the scheduling policy for phase transition, which helps to make a complete and correct full screen content rendered in advance. Before painting the complete first screen content, browser can reduce the parsing tokens number for one parsing phase, and also the time slice allocated for parsing phase. While entering layout phase, browser tracks the height of layout content to check whether it has reached the full screen height. Before reaching the screen height, browser can suspend paint and continue the iteration of parse and layout. After a full screen layout height is detected or a timeout is reached, browser can trigger the first paint operation immediately. When first screen content is painted, browser can change back to the default phases scheduling policy.

* 1. Situations to enable or disable first screen paint optimization

By defining the specification of the first screen paint optimization, web developers can indicate the browser to apply this optimization to help accelerate painting speed for content above the fold, shorten the time for the non-blank content displaying. For some web pages, first screen content relies heavily on all kinds of dynamic resources loading and execution. Blindly make the first screen content painted in advance will cause painting disorder and performance regression. Web developers can indicate browser to enable or disable render optimization in meta tag.

1. Specification description

First screen paint in advance specification introduces a new name and content attribute of Meta element.

The proposed name attribute is ***render-optimize-policy***, illustrating it is a render phase optimization policy.

The proposed content attribute’s key is ***first-screen-advance***, illustrating this render optimize policy focuses on painting the first screen content in advance.

The proposed content attribute’s value is ***enable*** or ***disable***, which illustrates browser could apply first screen paint optimization or not. The default value is ***enable***.

The specific example is as follow:

* **<meta name=" render-optimize-policy" content=" first-screen-advance [;enable]">**

This description indicates browser can apply first screen paint optimization according to its own first screen judgement and phase transition scheduling policy.

* **<meta name=" render-optimize-policy" content=" first-screen-advance;disable">**

This description indicates browser should not apply the first screen paint optimization.

1. Use cases
	1. Cases should apply first screen paint optimization

For a web page, if its first screen content doesn’t rely heavily on dynamic loading or parsing resources, web developers can add a meta tag with content as “first-screen-advance;enable” to indicate browser applying first screen paint optimization.

An example is as following:

<html>

   <head>

**<meta name="render-optimize-policy" content="first-screen-advance;enable">**

      <style>

          …

      </style>

      <script type=”text/javascript”>

          …

      </script>

      …

   </head>

   <body>

      <div class=”…”>

          <font>..</font>

      </div>

      <div>

          <p>…</p>

      </div>

      <div class=”…”>

          <img width=”256” height=”128” src=”…” />

      </div>

…

   </body>

</html>

For the above example, the web page’s first screen related elements and layout info can be decided at parse phase, it is a good case for applying first screen paint optimization.

* 1. Cases should not apply first screen paint optimization

For some complex web page, its first screen content relies heavily on dynamic resources, especially some external CSS/JS, or latter js event, which cannot be downloaded or decided in time at early parse phase. In this case, web developers can add a meta tag with content value as “first-screen-advance;disable” to indicate browser it is not a proper case to apply the optimization.

An example is as following:

<html>

   <head>

      **<meta name="render-optimize-policy" content="first-screen-advance;disable">**

      <script>

          function bodyOnload() {

             var elem = document.getElementById(“firstElem”);

elem.innerHTML = …;

          }

    …

      </script>

      <link ref=”stylesheet” href=”…” type=”text/css” />

 <link ref=”stylesheet” href=”…” type=”text/css” />

      <script href=”…” type=”text/javascript” />

 <script href=”…” type=”text/javascript” />

      …

   </head>

   <body onload=”bodyOnload”>

      <div id=”firstElem” class=”…”>

          <img width=”100%” id=“firstImage” src=”…” />

      </div>

      <div class=”…”>

          <table> … </table>

      </div>

…

<script>

 (function() {

 var styleSheetList = document.styleSheets;

 if (styleSheetList.length == 0) {

 var styleElem = document.createElement('style');

 styleElem.setAttribute('type', 'text/css');

 (document.getElementsByTagName('head')[0]).appendChild(styleElem);

 }

 var linkElem = document.createElement('link');

 linkElem.setAttribute('type', 'text/css');

 linkElem.setAttribute('ref', 'stylesheet');

 linkElem.setAttribute('href', '...');

 (document.getElementsByTagName('head')[0]).appendChild(linkElem);

 document.styleSheets[document.styleSheets.length-1].insertRule('......', 0);

})();

 </script>

…

   </body>

</html>

For the above example, the web page’s first screen content depends on many external links, css resources specified by body script, and even some latter triggered body.onload event. The layout information of first screen content is hard to predicate, applying the first screen paint optimization blindly will not get expected performance improvement, and it may result in incorrectness and disorder in painting for a long time. It is a bad case for applying such kind of optimization.