Optimization and Event loop

1. **Firefox display list**

   — Firefox goes over the render tree and builds a display list for the painted rectangular. It contains the renderers relevant for the rectangular, in the right painting order (backgrounds of the renderers, then borders etc). That way the tree needs to be traversed only once for a repaint instead of several times–painting all backgrounds, then all images, then all borders etc.

   — Firefox optimizes the process by not adding elements that will be hidden, like elements completely beneath other opaque elements.

2. **WebKit rectangle storage**

   Before repainting, WebKit saves the old rectangle as a bitmap. It then paints only the delta between the new and old rectangles.

3. **Dynamic changes**

   The browsers try to do the minimal possible actions in response to a change. So changes to an element's color will cause only repaint of the element. Changes to the element position will cause layout and repaint of the element, its children and possibly siblings. Adding a DOM node will cause layout and repaint of the node. Major changes, like increasing font size of the "html" element, will cause invalidation of caches, relayout and repaint of the entire tree.

4. **The rendering engine's threads**

   — The rendering engine is single threaded. Almost everything, except network operations, happens in a single thread. In Firefox and Safari this is the main thread of the browser. In Chrome it's the tab process main thread.

   — Network operations can be performed by several parallel threads. The number of parallel connections is limited (usually 2–6 connections).

5. **Event loop**

   The browser main thread is an event loop. It's an infinite loop that keeps the process alive. It waits for events (like layout and paint events) and processes them.

Reference

- How Browsers Work: Behind the scenes of modern web browsers
- MDN：Populating the page: how browsers work
- 深入浅出浏览器渲染原理
- 浏览器渲染原理