Fundamentals of Mobile Web Development

By Paul Kinlan

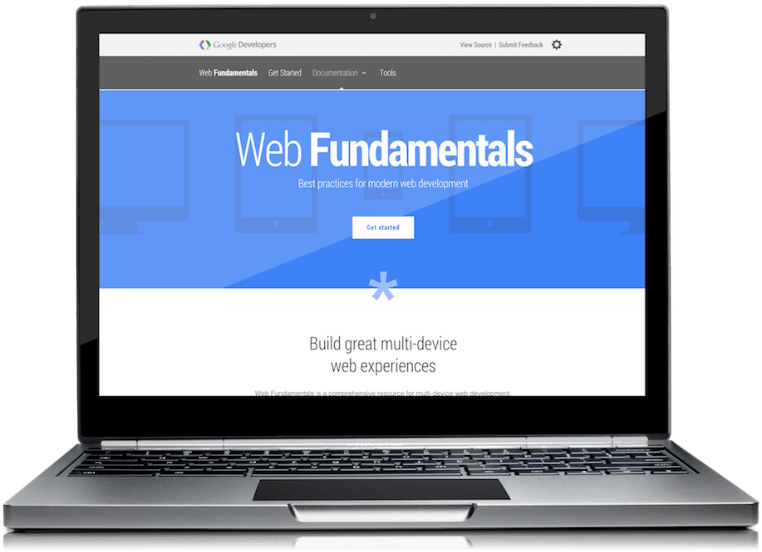
Paul is a Developer Advocate

At Chrome Dev Summit 2014 there was a whole host of topics and brand spanking new API's covered, but its not all about the new and shiny.

If you are a new Web Developer or even an experienced developer about to embark on exploring new APIs, chances are you'll follow these three steps: learn, build and iterate.

Matt Gaunt covers the ongoing efforts to address these problems from the Chrome Developer Platform team.

Learn.

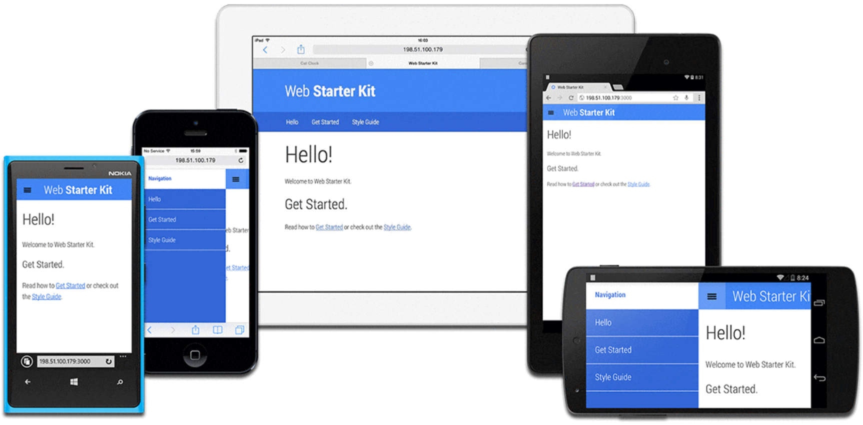

Google Developers
View Source | Submit Feedback
Web Fundamentals
Get Started
Documentation
Tools
Web Fundamentals
Best practices for modern web development
Get started
Build great multi-device web experiences

Web Fundamentals is a set of use case led documentation covering a range of topics. The core goal is get developers from little or no knowledge, to implementing best practices as quickly as possible.

One of the main goals of Web Fundamentals is to ensure that if you are new to a topic, the guidance reduces "choice paralysis" as much as possible. Addy Osmani covers this perfectly over at Pastry Box.

If you do spot any issues with the site or it's content or you'd like Web Fundamentals to cover a particular topic, then please do let us know by submitting feedback on Github.

Build.



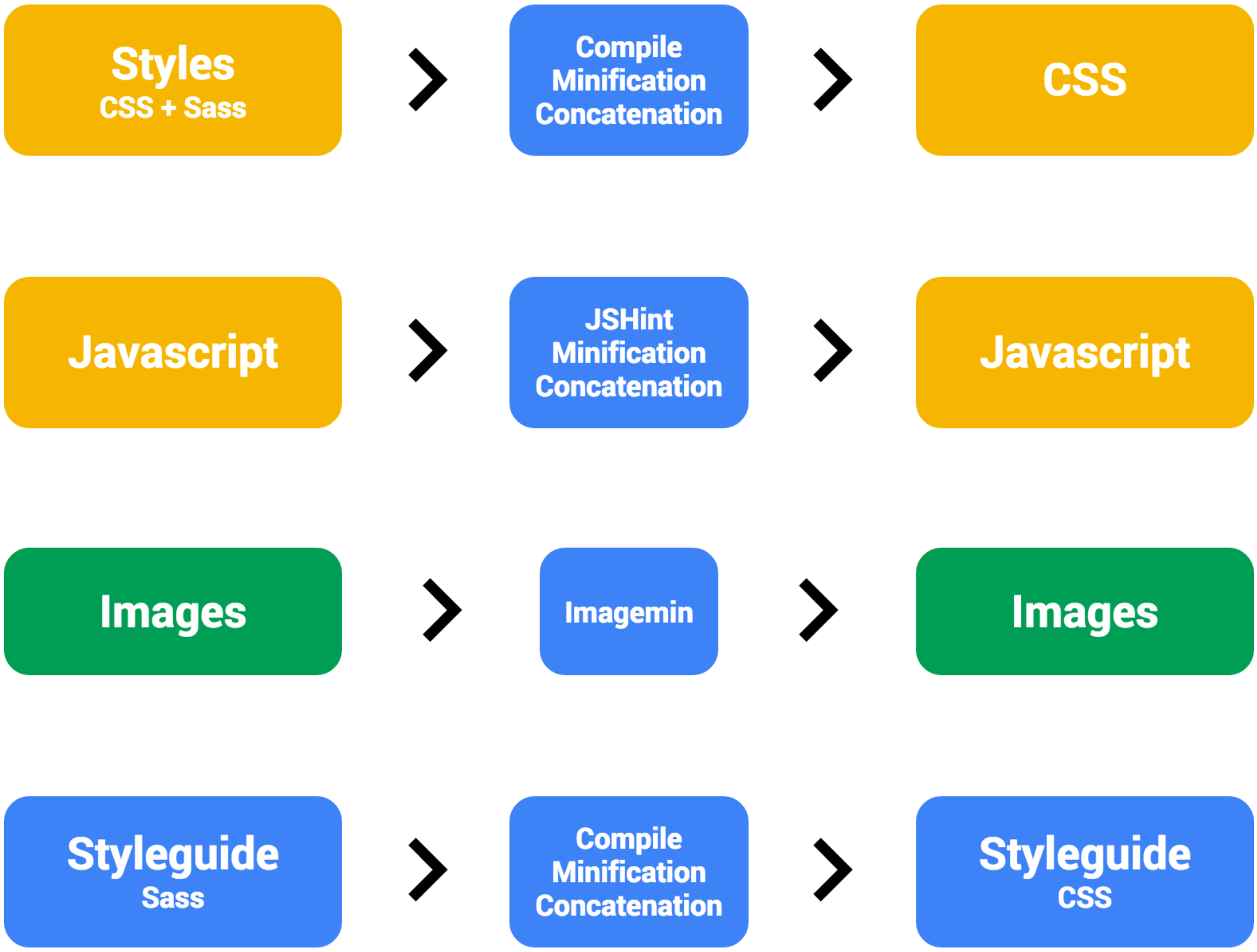
To help you kick off a new web project we created Web Starter Kit. It has everything you need:

- A solid build process
- Boilerplate HTML
- Styleguide

The Build Process

For those of you who are new to build processes, the easiest way to think of a build process is to view it as a program which takes a set of files and performs certain tasks on them and outputs new versions in a different location. The tasks optimise the files to improve load times, check for possible errors or handle tasks that can be automated.

In Web Starter Kit we have the following processes:



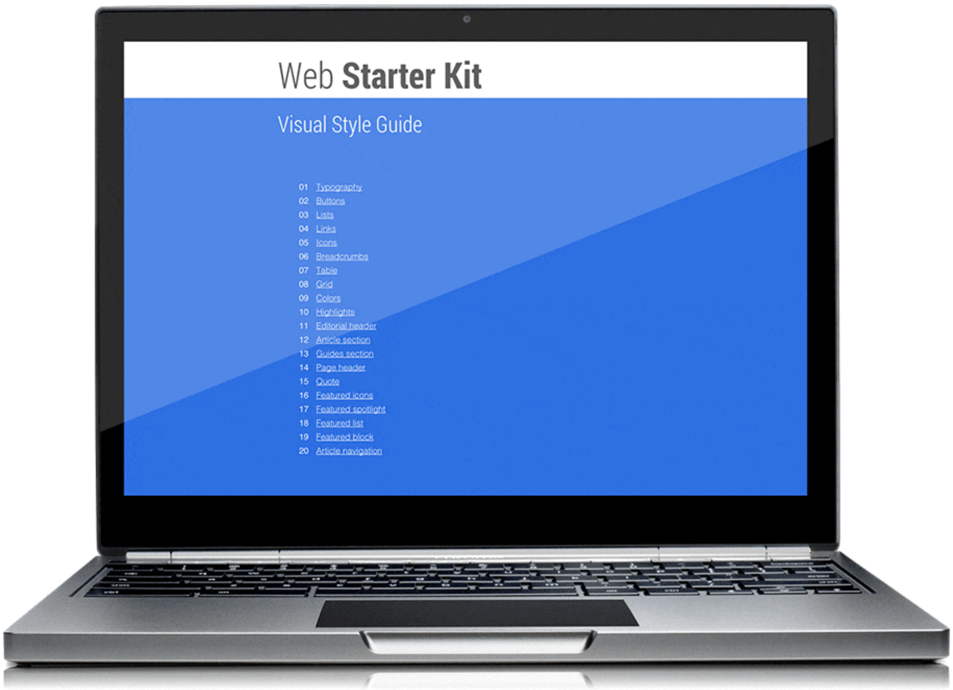
We minify and concatenate CSS and JavaScript so that the browser can fetch the file quickly, the JavaScript is also run through JSHint to check for JavaScript best practices and common coding mistakes. Images are minified with imagemin and you can get **huge** reductions in file size by using this. We also have a process to create the styleguides CSS.

Boilerplate for Multi-Device HTML

The first set of HTML you write for a new page is pretty bog standard and chances are you'll have some way of quickly getting hold a stock HTML file that works well across multiple devices and screen sizes.

In Web Starter Kit we wanted to add in support for any features which blurred the lines between the platform and your site, so we've added support for add to home screen and splash screens for Android, Windows Phone, iOS and Opera Coast.

Styleguide



Web **Starter Kit**

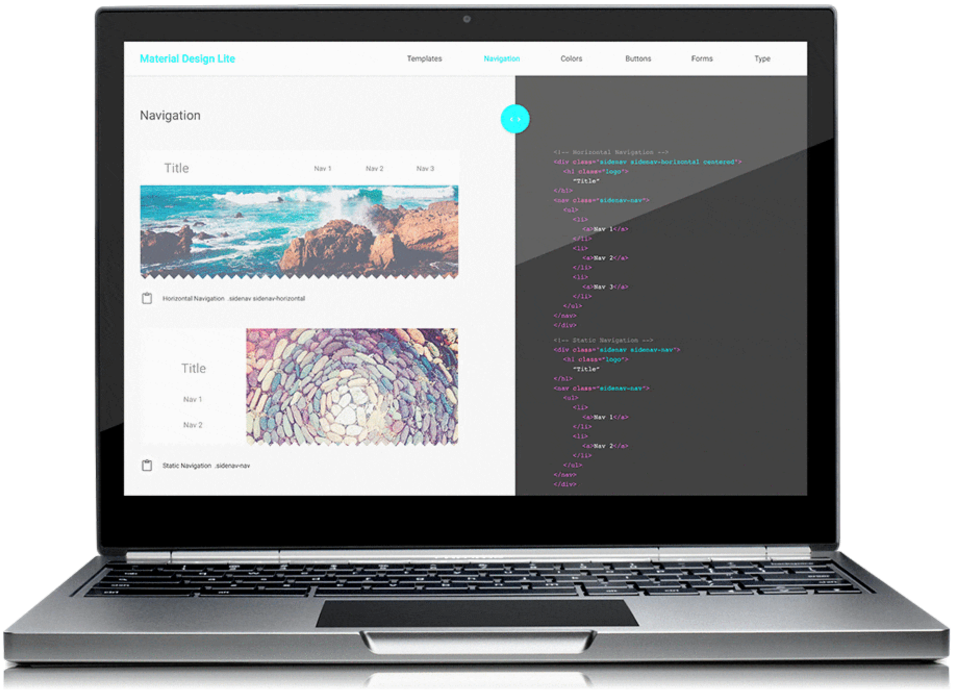
Visual Style Guide

01 Typography
02 Buttons
03 Lists
04 Links
05 Icons
06 Breadcrumbs
07 Table
08 Grid
09 Colors
10 Highlights
11 Editorial header
12 Article section
13 Guides section
14 Page header
15 Quote
16 Featured icons
17 Featured spotlight
18 Featured list
19 Featured block
20 Article navigation

The final piece of Web Starter Kit is it's Styleguide.

This gives any new project a great set of default styles and components that encourages style driven development. You can alter existing styles to elements and add your own.

In the next version of WSK, due for release early next year, we are working hard to simplify how the styleguide fits together and switching to a Material Design look and feel. Matt showed an early mock of what this may look like at Chrome Dev Summit and you can see an example below.



```
<!-- Horizontal Navigation -->
<div class="sidenav sidenav-horizontal centered">
  <h1 class="logo">
    "Title"
</h1>
<nav class="sidenav-nav">
  <ul>
    <li>
      <a>Nav 1</a>
    </li>
    <li>
      <a>Nav 2</a>
    </li>
    <li>
      <a>Nav 3</a>
    </li>
  </ul>
</nav>
</div>
<!-- Static Navigation -->
<div class="sidenav sidenav-nav">
  <h1 class="logo">
    "Title"
</h1>
<nav class="sidenav-nav">
  <ul>
    <li>
      <a>Nav 1</a>
    </li>
    <li>
      <a>Nav 2</a>
    </li>
  </ul>
</nav>
</div>
```

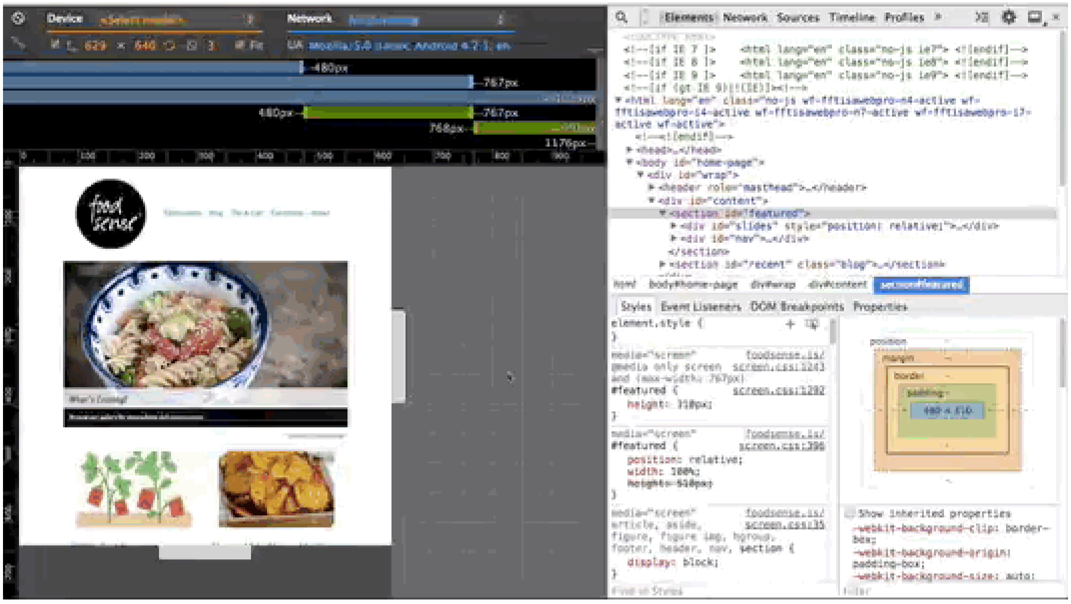
Iterate.

Once you've started to put your new knowledge into practice, you'll want to use DevTools to debug, improve and maintain your work.

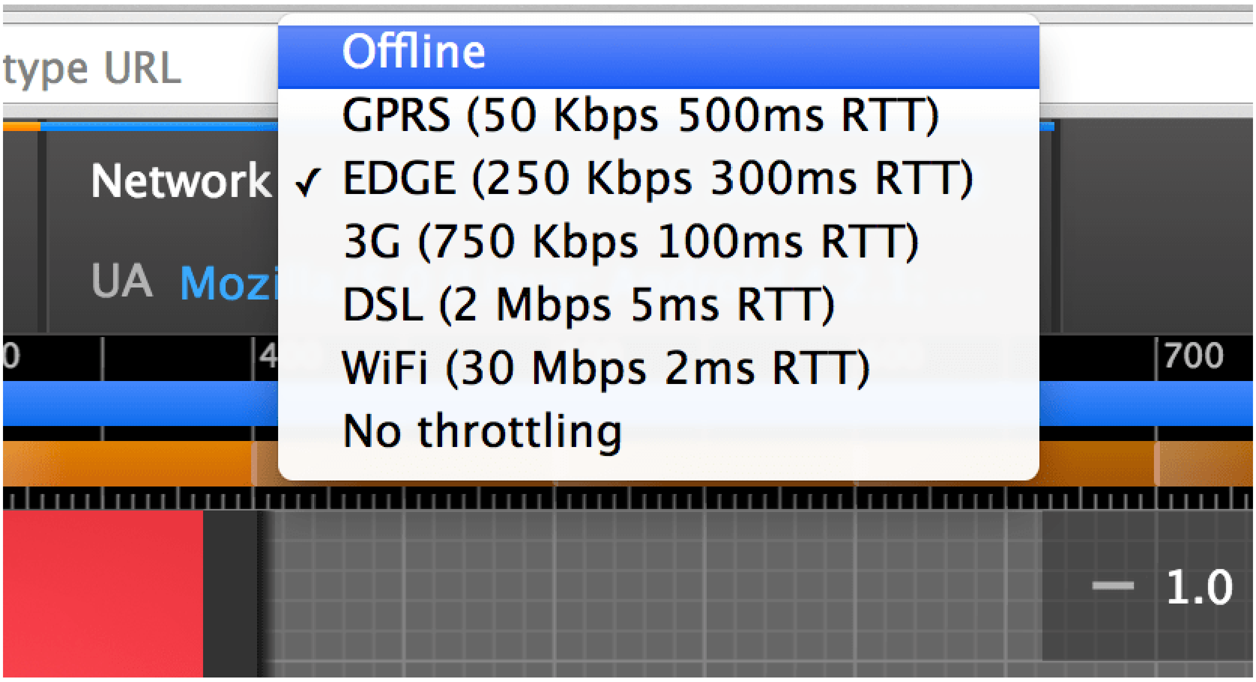
There are some huge new features landing in DevTools and Matt takes a look at the following new features.

Device Mode

Device mode is a new section in DevTools which allows you to quickly see how your site works across different mobile devices, while viewing the media queries in your CSS.



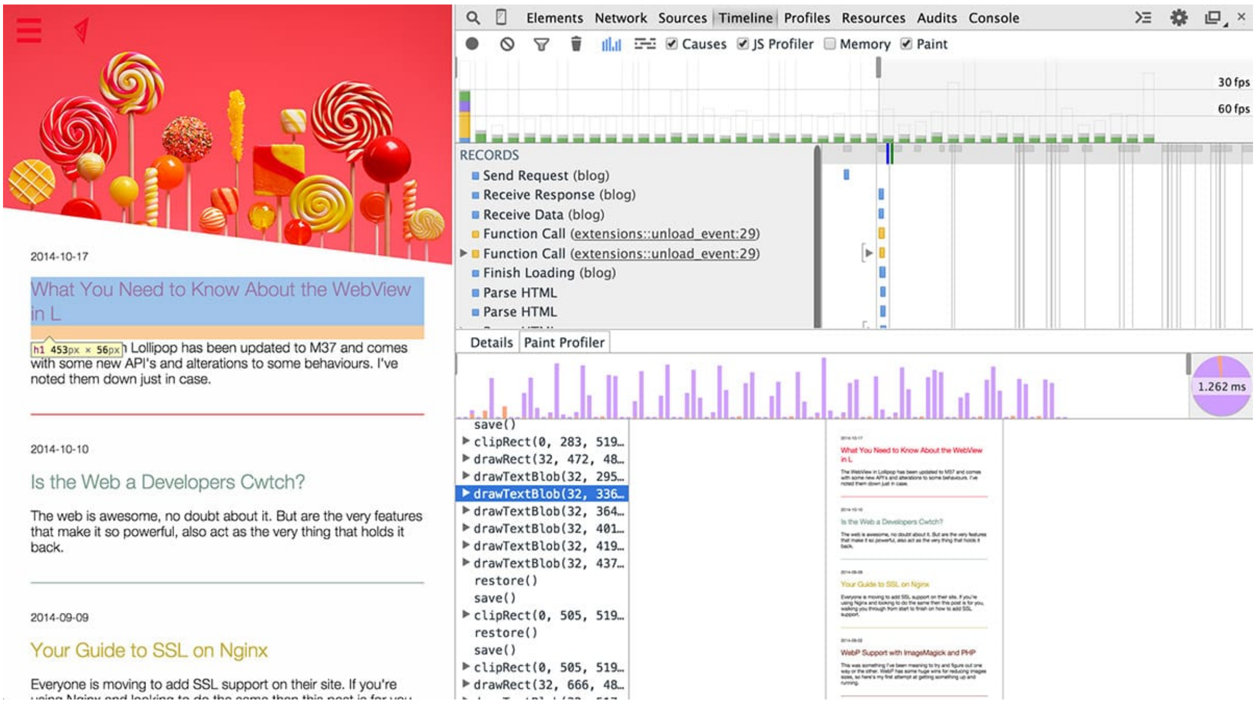
One of the great features of Device Mode is the ability to throttle the network speeds, allowing you simulate the experience of a user on a GPRS, EDGE, 3G, DSL or Wifi connection.



Paint Profiler

If you've ever opened up the timeline tab and hit the record button, you've probably seen some paint events happen in the waterfall. Normally this would be a black box with no way for you to know why the browser had done, or what it was doing.

Paint profiler no gives you more information on what exactly the browser is doing during that paint.



2014-10-17

What You Need to Know About the WebView in L

Lollipop has been updated to M37 and comes with some new API's and alterations to some behaviours. I've noted them down just in case.

2014-10-10

Is the Web a Developers Cwtch?

The web is awesome, no doubt about it. But are the very features that make it so powerful, also act as the very thing that holds it back.

2014-09-09

Your Guide to SSL on Nginx

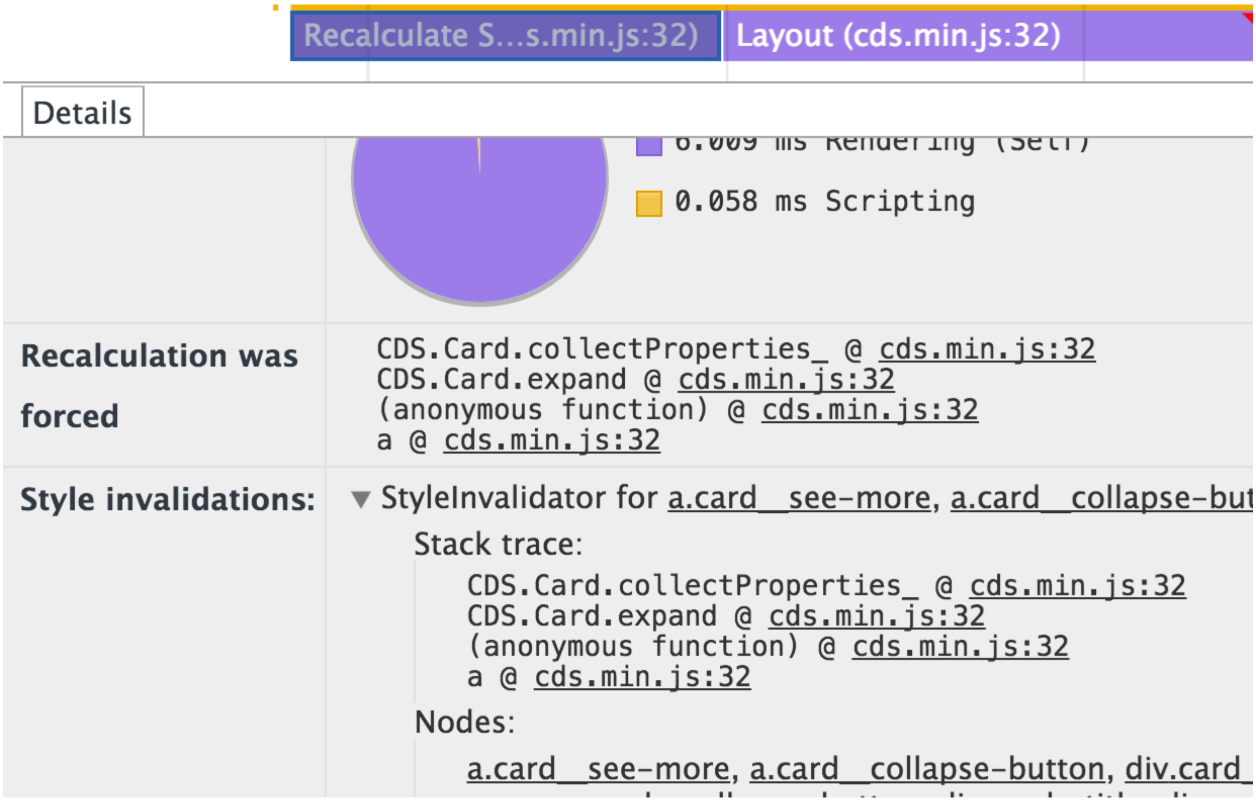
Everyone is moving to add SSL support on their site. If you're

```
Send Request (blog)
Receive Response (blog)
Receive Data (blog)
Function Call (extensions::unload_event:29)
▶ Function Call (extensions::unload_event:29)
Finish Loading (blog)
Parse HTML
Parse HTML
```

```
  save()
▶ clipRect(0, 283, 519…
▶ drawRect(32, 472, 48…
▶ drawTextBlob(32, 295…
▶ drawTextBlob(32, 336…
▶ drawTextBlob(32, 364…
▶ drawTextBlob(32, 401…
▶ drawTextBlob(32, 419…
▶ drawTextBlob(32, 437…
  restore()
  save()
▶ clipRect(0, 505, 519…
  restore()
  save()
▶ clipRect(0, 505, 519…
▶ drawRect(32, 666, 48…
```

Invalidation Tracking

DevTools now gives a reason why a paint or layout occurred whenever it can, this is useful for anyone learning about the timeline, the browser behaviours and allows you to optimise your code to prevent performance issues.



```
0.009 ms Rendering (Self)
0.058 ms Scripting
```

Recalculation was forced

```
CDS.Card.collectProperties_ @ cds.min.js:32
CDS.Card.expand @ cds.min.js:32
(anonymous function) @ cds.min.js:32
a @ cds.min.js:32
```

Style invalidations:

▼ StyleInvalidator for a.card__see-more, a.card__collapse-but

Stack trace:

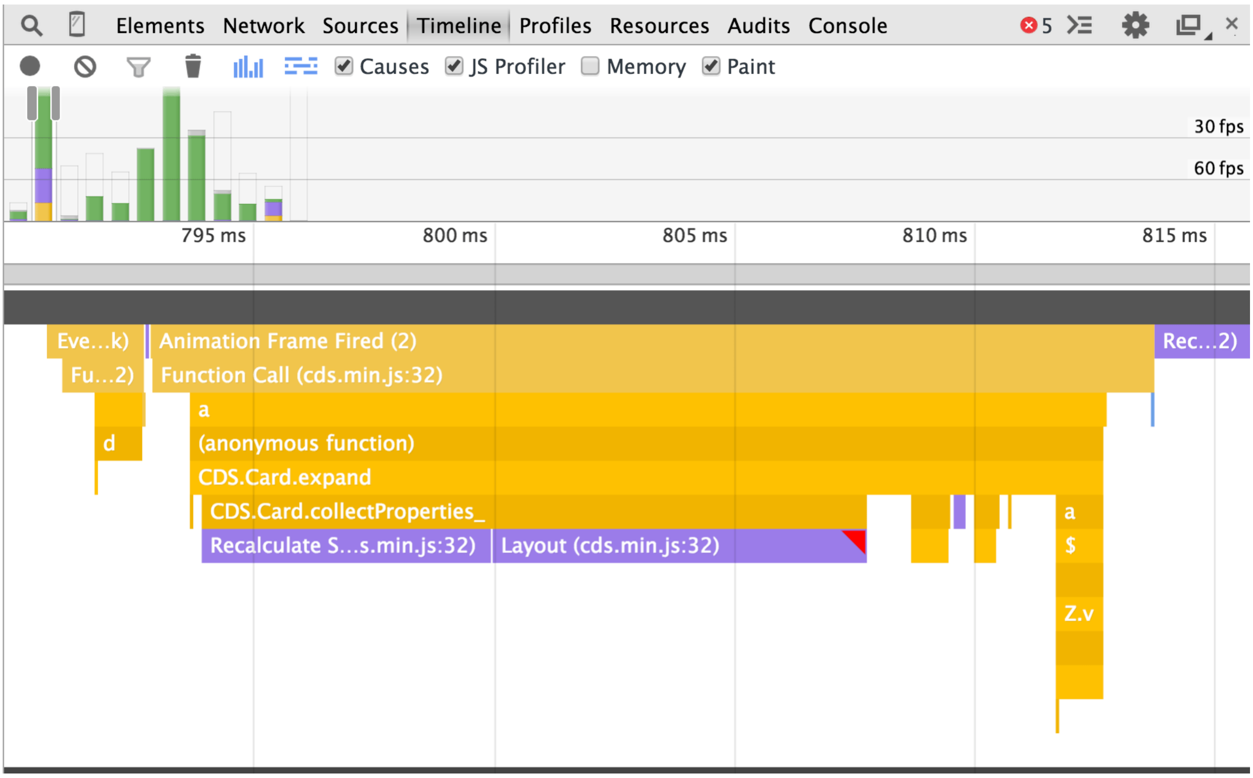
```
CDS.Card.collectProperties_ @ cds.min.js:32
CDS.Card.expand @ cds.min.js:32
(anonymous function) @ cds.min.js:32
a @ cds.min.js:32
```

Nodes:

a.card__see-more, a.card__collapse-button, div.card_

Flame Chart View

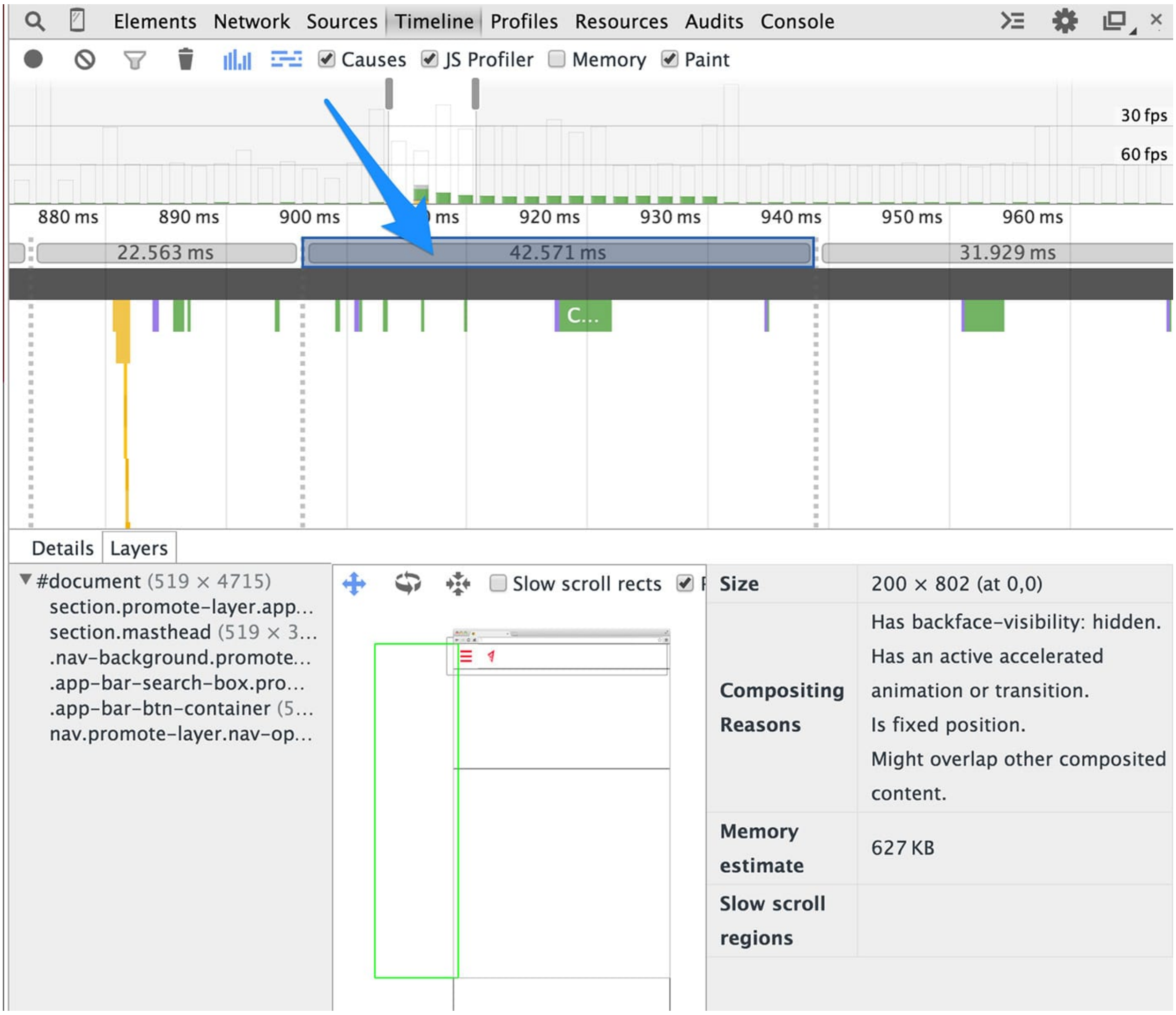
This is a very different way of viewing the information available in the timeline. This makes it much easier to see how tasks overlap and what browser behaviour happened as a result of other tasks.



☑ Causes ☑ JS Profiler □ Memory ☑ Paint

Frame Viewer

While in Flame Chart view, you can select a specific frame and within this, you'll be able to explore which elements in the page had been promoted to a composite layer as well as why they've been promoted.



▼#document (519 × 4715)
section.promote-layer.app...
section.masthead (519 × 3...
.nav-background.promote...
.app-bar-search-box.pro...
.app-bar-btn-container (5...
nav.promote-layer.nav-op...

Size	200 × 802 (at 0,0)
Compositing Reasons	Has backface-visibility: hidden. Has an active accelerated animation or transition. Is fixed position. Might overlap other composited content.
Memory estimate	627 KB
Slow scroll regions	

Learn. Build. Iterate.

These are some of the efforts from the Chrome team to help developers get up to speed with web development, so be sure to check out Web Fundamentals, Web Starter Kit and the new features in Chrome DevTools.

Last updated July 2, 2018.