I love Chrome developer tools. They make it easy to debug issues with your application and they provide some awesome tools and functionality without having to leave the browser UI. These are some of the tools I use the most.

The following recipes assume that you’re familiar with Dev Tool and how to open them.

DevTools Device simulation and Network throttling

Figure 1: Device simulation in DevTools, also allows to create custom presets and modify existing ones

The network throttling options inside DevTools give you a good starting point for testing what your users' experiences will be like in the devices you're targeting.

Figure 2: Available network condition presets and the option to add custom ones

At the top of the list you have the option to add new presets and customize existing ones. I've created one customization to the 3G preset by adding latency to, in my opinion, provide a better representation of network conditions outside urban areas.

Figure 3: Configuring a custom network condition preset using DevTools

Be aware that, while the Network Condition emulator is a good starting point for testing your design under different network conditions, it is not a replacement for testing using actual devices. There are things that a desktop machine cannot simulate including:

- Mobile device startup time
- How many low power cores are started versus how many high powered cores are started in a multi-core devices (not all cores in a multi-core mobile phone are made equally)
- How long it takes for a device to start the wifi receiver

More information about network conditions available at the Google Developers site

Related video about mobile performance from Alex Russell at the Chrome Developer Summit 2016.

DevTools Device Mode

With DevTools Device Mode you can get an approximation of what your site will look like in different devices without having the actual devices available.

Figure 4: Default Device Mode View

To activate Device Mode, click the button to the left of the DevTools menu (in the image below it’s inside a red circle).

Figure 5: Where is the button to activate Device Mode

When Device Mode is active, the button is blue (device mode active) and when it’s not then the button is black (device mode disabled)

DevTools provides a basic set of devices it will emulate and it gives you the option of creating additional devices that are specific to your development workflow. To use the preset devices pull the responsive menu at the top of the DevTools window. It will provide you a list of available devices to emulate as well as a responsive mode (default) that you can play with by resizing the DevTools window.

Figure 6: Default Device Mode View

Furthermore you can create custom devices to test with using the edit function at the bottom of the responsive and device menu. This will take you to the device

editor.

Figure 7: Device Editor

Clicking the add device will show you a dialogue where you can enter information for your custom device.

Figure 8: Add a custom device

OK, we've create new devices but how do we use them? For t` szhe most part we can scroll up and down the page using our mouse and test special things we've coded for touch devices.

At the top of the screen you'll see a button to change the device's orientation (if supported).

Figure 9: The top bar on Device Mode. From here you can change the orientation of your simulated device

I know, there's a lot of information for just one feature. This is the last one, promise.

Figure 10: additional options

If you click the menu on the far right of the Device Mode bar (show in the figure

10) you will get access to additional options. We will not go into details about these options but will mention them as resources for further study:

- Toggle display of the device’s frame
- Toggle display of media queries
- Show / Hide rulers
- Change DPR value
- Change the device type and whether it’s a touch device
- Hide the device type
- Throttle Network Connection
- Capture a screenshot of the content in the current viewport
- Reset to defaults

Work with your content directly on DevTools

There are times when I'm working on a design and start tweaking the design in the browser by adding attributes or making changes to the content directly in the browser and wish I could make those changes permanents. Now you can :)

To make a local folder's source files editable in the Sources panel:

1. Right-click in the left-side panel
2. Select `Add Folder to Workspace`
3. Choose location of local folder that you want to map
4. Click Allow to give Chrome access to the folder

To get this working I open a page on the site I want to work with:

Figure 11: DevTools detached window

When I click on the left-side panel I'm show the prompt to add the folder to the workspace.

Figure 12: Add folder to workspace
Figure 12:

The browser will then ask you for full permissions for the workspace.

> Make sure you don't share any sensitive information. THis may not be a big problem but we better be sure we're not sharing anything we wouldn't want to share in public.

Figure 13:

Full permission request

Furthermore you can inspect and edit the DOM and HTML of your page directly. Be careful as this assumes that you are at least familiar with HTML and how CSS classes and IDs affect the document's styling

To inspect a specific element on your page highlight the element, right click and select `inspect`.

Figure 14: How to inspect an element on yoru page

You can also use keyboard shortcuts to open DevTools in Inspect Element mode: `Ctrl + Shift + C` (Windows) or `Cmd + Shift + C (Mac)`, then hover over an element. DevTools automatically highlights the element that you are hovering over in the Elements panel.

You can edit the elements by double clicking on the element or right clicking on the element and choose an option from the list it presents (shown below).

Figure 15: List of options presented when you right click an element in inspect mode

More in depth information about what you can do and how will it help your workflow is located in the DevTools Documentation Pages

Working with Service Workers

Service Workers are awesome and they are very powerful. They are also very hard to debug. DevTools has supported service workers for a while and has reorganized things around to produce the application panel.

Figure 16: Devtools application panel

From here you can work with several different technologies that make (Progressive) Web Applications:

- Service Workers
- Local and Session Storage
- IndexedDB
- WebSQL
- Cookies

In this section we'll work with Service Workers. The other sections of the panel are left as an exercise for the reader.

In the application panel the service worker section is the second one from the top on the left-side of the application panel (and highlighted in figure 17)

Figure 17: Service workers option in application panel

The panel will show the service workers active for the site or, if you select the show `all` checkbox at the top of the screen it will show all service workers active in the browser.

The other options include:

- **Offline** takes the browser offline and allows you to test if the offline caching

functionality works. For this to work you must have a service worker for your site

- **Update on reload** forces the service worker to update when the page is reloaded. This saves you from having to unregister the service worker every time you make changes to it
- **Bypass for network** ignores the service worker and fetches resources from the network

Each service worker will the source file and when it was received, show its status (active or stopped) and how many clients (windows or browser tabs) are using that particular service worker.

This is important because, unless you've configured the worker to do automatically claim all clients, you must close all the clients before a new version of the service worker will take over.

Each service worker also gives you the following options

- **update** updates the service worker
- **push** simulates a push event
- **sync** simulates a background sync event
- **unregister** removes the service worker from the list

This panel gives you a good starting point for debugging your service workers.

Code coverage

This feature is only available in Chrome 59 and later

I use Critical to create and inline the CSS for above the fold content of a page and UNCSS to remove any unnecessary CSS for these web pages.

This is important because in large or long-lasting projects there may be CSS that is no longer needed in a page or the whole site and it hasn't been removed out of laziness or because the people who oringally created the CSS are no longer available and sometims developers, myself included, think that if we don't use the content it won't be downloaded.

Since Chrome 59 DevTools offers a coverage tool that will tell you how much of your CSS is used in a given page. This may help you decide if you should use Critical to inline that CSS on the page or UnCSS to remove the unused CSS rules and selectors from your CSS stylesheets.

Figure 18: Where is the coverage panel item

To use the coverage tool, open DevTools open the tools menu in the far right corner of the DevTools GUI, select `More Tools` and from the submenu select `Coverage`.

The one decision that the coverage panel doesn't help with is how to work on the coverage of external scripts. We can still tell how much of the script is unused you can't really integrate it to your own bundled script without risk of loosing any updates the vendor makes.

Audits: powered by Lighthouse

This feature is only available in Chrome 60 and later

Lighthouse is a tool to test if your web content (app or site) meets the criteria for Progressive Web Applications and other tests to make sure your application performs well and is accessible.

As of Chrome 60 you can run the lighthouse tests within DevTools Audits menu. This menu is located on the far right of the DevTools menu bar.

Figure 19: DevTools Audits Menu

When you access the menu you will see the Lighthouse logo and a button to begin the tests... When you click on that button you will be presented with a menu of options representing the tests you can run.

Figure 20: The 4 types of test you can run from DevTools

The available tests are:

- **Progressive Web Application** Does the oage meet the requirements of a PWA?
- **Performance** How long does it take for the app to load and become responsive?
- **Best Practices** Does the app follow best practices in application development?

- **Accessibility** How accessible is your application?

Once you've chosen what test to run and have clicked the `perform an audit` button you will be see results similar to the ones shown in the image below.

Figure 21: Audit results

The circles at the top of the report show passing percentages for each of the tests you ran. Individual results and suggestions appear as you scroll down the report. The left window will have a list of all the reports you've run on this browser. You can run additional tests and remove tests when you need to run them again.

As with many things in DevTools, these reports are advisory in nature and will not implement any changes on your code. It is up to you as developer to make the changes you need to make.

How to find stuff in DevTools?

This feature is only available in Chrome 59 and later

DevTools is awesome but sometimes can be hard to manage. There's so many tools and functions and shortcuts that it gets hard to remember everything you use unless you use it regularly. In Chrom 59, DevTools introduced a command menu similar to the one in Visual Studio Code

`Cmd + Shift + P` (in Mac) or `Ctrl + Shift + P` (in Windows) bring up the DevTools Command Menu, then type to filter and hit Enter to trigger the action. Typing ? will give you a list of commands you can use in addition to just searching for the task you want to perform.

Figure 22: DevTools Command Menu

A few sample actions you could try:

- Appearance: Switch to Dark Theme
- DevTools: Dock to bottom
- Mobile: Inspect Devices...
- Network: Go offline