Course finished

===

To learn more, visit developer.chrome.com/devtools

Settings -> Shortcuts

===

Ctrl + Shift + I shortcut for developer tools

console.warn, console.error

console.error is very useful because it gives a stack trace

When you see in the console a link on the right hand side such as script.js:3, clicking it will actually show the line in the code responsible for that console log.

On the bottom of your console, you can filter what you see in the console (all, errors, warnings, logs, debug). Have to click the funnel icon to see this.

clear() clears the console

console.assert. If true, returns nothing, otherwise it returns “Assertion failed:” while also giving a stack trace. Use for sanity checks.

console.group(“...”), console.groupEnd() really useful for organizing your logs

Can even apply CSS to console logs

Ex/ console.group(“%cFinishing part”, “font-size: x-large”);

Can console log DOM elements

console.log(document)

If you do console.dir(document), you get a javascript view of the document

To run a timer:

console.time(“Steps time”); // Or whatever label you want

…

console.timeEnd(“Steps time”); // Matching the label

===

In your JavaScript source code, you can simply type **debugger;** and it will act as a breakpoint when you refresh the page

Typing **inspect($(‘h1’))** in the console will send you to h1 in the elements tab.

Typing **monitorEvents($(‘h1’), “mouse”)** in the console will track mouse events in your h1. The MouseEvent object itself gives a lot of useful information such as X and Y coordinates, what keys are being pressed, etc.

Elements Panel

At the bottom of the elements panel, you’ll see a breadcrumb. USE THIS. Incredibly useful for going up to the parent of the selected DOM element.

If you hover over the selected DOM element, it will be highlighted on the screen, including its width and height in pixels.

To add elements to the DOM, inspect some element, then right click -> edit as HTML. Once you’ve added your new HTML, have to click away for it to take effect.

When clicking on an element, right click -> add attribute

You can drag a clicked element in the elements pane and move it around the page!

You can also delete a clicked element with the delete key

Typing $0 in the console will return the last element that you clicked. Same idea for $1 through $4.

CSS

Computed styles tell you ALL the styles being applied to your element. You can even click on the individual computed styles to see where they are coming from.

If you click on the preview square next to a color, you can change it with the color wheel.

When viewing computed styled, you have the option to show inherited. This includes styles that were not explicitly set.

Toggle element state (little pin icon)

In the elements tab, right click -> break on -> attributes modification. Useful for tracking when Javascript adds attributes to an element.

<h1 onclick=“console.log(‘hi’)”>

Then you can track this event listener in the right sub-panel

Ctrl + f works with CSS selectors (ex/ #chromeImage)

Sources Panel

At the bottom of the panel, you will see a { } icon. This does pretty print, which reformats your JavaScript to look nicer. Especially useful for minified files.

On the left side of the panel, there’s a tab for content scripts. This shows scripts from browser add-ons running on your page.

You can edit JavaScript directly in the sources panel and test the changes in the browser (could be incredibly useful for temporary console logs on say, an ng-click)

When setting a breakpoint, when the breakpoint is active, you can hover over variables to check their values.

Author recommends combining the use of breakpoints with the console

Right click on a variable or function -> add to watch. Can also manually add a watch expression.

The call stack has the most recent call on top. You can follow the call stack by clicking the layers.

Conditional breakpoint -> right click on a breakpoint -> edit breakpoint

Can set a breakpoint on AJAX calls

Breaks when url contains whatever you specify

Workflow:

-Go into the network tab

-Filter by XHR calls

-Figure out which AJAX call you want to break on

-Return to the sources tab

-Under XHR Breakpoints, click +

-Where it says “Break when URL contains”, type in something

-Refresh the page

You will then break on the function call that makes the AJAX request!

Not sure how useful this is in practice because of the huge stack trace you have to go through to find your actual function call (couldn’t even find it for an example using Angular’s $http method).

Event listener breakpoints are really useful. Can break on click events for instance. Also break on load events.

The audit panel gives suggestions on how to improve your webpage

Resources Panel

You can see all the scripts, stylesheets, images, etc of your page neatly categorized in the resources panel under Frames. Much more efficient than looking through the elements panel.

WebSQL obsoleted by IndexedDB

IndexedDB seems pretty lame for local storage (at the very least, hard to use)

Local and session storage very easy to use.

localStorage.setItem(‘test key’, ‘test value’);

sessionStorage.setItem(‘test session key’, ‘test session value’);

The only difference is that local storage will persist when you leave the browser, whereas session storage only lasts for the session.

Local storage used to replace cookies in a lot of web applications.

Can edit local & session storage key & value directly, as well as delete

Creating a cookie is also stupidly simple:

document.cookie = “cookiekey = cookievalue”;

Cannot modify the cookie

Network Panel

The Initiator column looks really useful, tells you which line of code is responsible for that network call

The size column tells you if the resource was loaded from the cache

In settings, you can disable cache

Can filter the documents you’re looking at (ex/ only show stylesheets)

Can sort by any of the columns (name, status, etc)

Timeline sorting is more detailed than the other sort options

Timeline Panel

You need to press the record button

Shift + refresh button does the same thing as Ctrl + F5

Can filter on requests that are taking more than a certain amount of milliseconds

Seems very difficult to use, would only care if you’re obsessed with website performance

Profiles Panel

CSS Selector Profiles removed?

Not terribly useful unless you care about garbage collection and memory management