Workshop order

Splitting this out to make it a little clearer. It is probably easier to adjust and agree on the order here and then amend slides/document to match.

- Overview of the day and introductions
- Overview of the apps under test - sweetshop, playground, the pulper, todoMVC
- Device Toolbar
- Docking Dev Tools
- Run Command (ctrl+shift+p)
    - Screenshot (of inspected elements etc)
    - Disable/Enable javascript
    - Reload / Hard Reload
- Elements Panel
- Console Panel Basics - basic errors and simple JavaScript
- Sources
- Snippets
- Audits - a basic overview of Lighthouse
- Network
- Application - cookies, storage, etc.
- Security - look its security
- Chrome Extensions
- Others: Incognito Mode, People, Task Manager
- Console Panel Advanced

Logistics:

https://ministryoftesting.com/events/london-tester-gathering-workshops-2019

- https://ti.to/mot/london-tester-gathering-workshops-2019?source=motsite#schedule
- Friday 28th June 2019
- 09:30 - 13:00
    - 09:30 - 10:00 Welcome, Introductions, What do you want to learn? Overviews
    - 10:00 - 10:30 Device Toolbar, Docking, Run Command, Experiment Time
    - 10:30 - 11:00 Elements Panel, Console, Experiment Time
    - 11:00 - 11:15 Formal Break
    - 11:15 - 11:45 Sources, Snippets, Audits, Experiment Time
    - 11:45 - 12:15 Network, Application, Experiment Time
    - 12:15 - 12:45 Security, Console, Plugins, Experiment Time
    - 12:45 - 13:00 Discussion of lessons learned, Q&A, final thoughts, wind down

Course material:

Below is my original brain dump of cool things in DevTools - I've also put in brackets the page names within the Sweet Shop project site where you can find the examples.

- To clear the basket on the sweet shop at anytime, users can go into the console and use

localStorage.clear()

Also show that we can right click on the local storage in Application and 'clear' from there.

Sweet shop site:

- https://sweetshop.netlify.com
- https://sweetshop.vivrichards.co.uk/

DevTools (F12) Windows,

DEVICE TOOLBAR

Cover:

- Difference between simulator and emulator (this is a rendering simulator but it also emulates http header sending)
- Responsive layout testing
    - show media queries
    - Resizing with rulers
    - Editing sizes directly
- Screenshots from this page in different sizes
- Rate Toggling/Throttling

(ABOUT PAGE)

MOBILE PANEL

* Simulate mobile viewport (Use Device Mode to approximate how your page looks and performs on a mobile device.)
Click on the 'Toggle device toolbar' button, here you can select through a number of different presets for devices as well as orientation of the device. You can also define your own presets for devices if you wish.

Show example of somebody using a specific mobile device style to hide or show content. You may want to test that something only shows/doesn't show for a smaller device for example.

(ABOUT PAGE)

* Device frames

Select iPhone 6/7/8 , Click on ... and 'Show device frame'/'hide device frame'

(ABOUT PAGE)

* Throttling

Mid-tier mobile simulates fast 3G and throttles your CPU so that it is 4 times slower than normal. Low-end mobile simulates slow 3G and throttles your CPU 6 times slower than normal. Keep in mind that the throttling is relative to the normal capability of your laptop or desktop.

The Throttle list will be hidden if your Device Toolbar is narrow (browser window is narrow).

ELEMENTS PANEL

Cover:

- What is it?
- What do we see?
- What if we don’t understand HTML?
    - This is a good place to start, then look up the tags on mdn
- This is different from the “View Source” option
    - view source is loaded from server, Element view is the DOM and may be affected by JavaScript
- Can manipulate HTML in this view

- Can see applied CSS, can Amend CSS here as well
- Can see javascript events which have been assigned to elements.
- Element states - e.g. hover

(LOGIN PAGE)

* Field required- client side validation (html5)

(LOGIN PAGE)

* Field type - client side validation (html5)

(LOGIN PAGE)

* Inspect markup (to much info exposed)

Sometimes a page may contain information which you wouldn't want rendered for a user to see. In this example we're going to find a hidden field which is exposing an Id which could be used to retrieve information from a database for example if you had access.

(LOGIN PAGE)

* embeded styles in pagewithin <style> attributes

(LOGIN PAGE)

* Inspect markup (finding style issue and resolving)

It's nice to be able to not just raise a bug but where possible find out what the problem is and either make the fix and get it reviewed or pass the fix on to a developer. In this example we are going to find an issue with some styles, and fix it.

Right click inspect the page (or click 'Inspect' icon) and start to hover over elements. In this example we can see there is a line or something rendered, click on it. We can see there's text which is to small to be read.

(LOGIN PAGE)

- Show both altering the inline style and adding a new style!

(LOGIN PAGE)

* Colour Contrasts (A11y)

Right click, inspect element, select element colour in elements panel, click on contrast ratio drop down to see best contrast ratio AA/AAA

**** CHROME CANARY - You get two lines so can see where AA or AAA accessible

CONSOLE PANEL - BASICS

Provide a quick overview of console, but leave deeper stuff until later.

- This is a javascript console, we can see debug messages, errors and write simple javascript
- Examples of simple javascript
    - I often use it as a calculator e.g. 2+2
- Examples of console commands
    - `variable_name`
        - See contents of variables found in source
    - console.log(“hello”)
        - Write messages etc. to console
    - Console.table
        - View variables and structures in console

(YOUR ACCOUNT PAGE)

- how to get a list of element properties in tabular format

get list of inputs properties

console.table($$('input'));

- expand array and show the fields, hovering over fields displays the element to the user in the browser

Likewise we could get list of links and their properties

console.table($$('a')));

get specific properties of inputs

console.table($$('input'), 'maxLength');

Fun, but not just for fun examples:

- https://developer.mozilla.org/en-US/docs/Web/API/Document/designMode
- document.designMode="on"
- document.designMode="off"

Just for fun examples:

- speechSynthesis.speak(new SpeechSynthesisUtterance(2+2))

Base 64 conversions

- btoa(“test”)
- atob("aGVsbG8=")

JSON conversion - pretty print

- JSON.stringify(anObject , null, 2)
- JSON.stringify(JSON.parse('{"message" : {"title": "hello"}}') , null, 2)

Example sites with embedded JSON to pull out and use as examples if needed:

- https://www.bbc.co.uk/

CONSOLE PANEL - ADVANCED

Cover:

- JavaScript debugging
- Demo the snippets chrome extension and explain it as a way to learn simple javascript
- Coding
- Coding with snippets view

(SWEETS PAGE)

* Errors - sometimes you may get some errors on your application being thrown

In this example we can see there is a '404 (Not Found) error being thrown. Sometimes for instance you find that an asset i.e. image, style sheet or javascript file name may be incorrect and so cannot be found.

(BASKET PAGE)

* Show how shipping doesn't total correctly..

ADD BREAK POINT ON TO LINE WHICH SAYS: subTotal + shippingCost within custom.js

GO INTO CONSOLE

Input

subTotal + shippingCost

This will show the number appended

Next input:

+subTotal + +shippingCost

This now shows the correct calculation

Later on we will show how to keep a local fix of this using local over rides… (for more advanced learners possibly or to do later in the workshop)

TIP: SHIFT-ENTER IN THE CONSOLE
To write commands that span over multiple lines in the Console, press shift-enter.

Once you're ready, press enter at the end of the script to execute it:

- Remove max-length off all input fields

```
let inputTags = document.querySelectorAll('input');
for(let i=0; i<inputTags.length; i++)
{
   inputTags[i].removeAttribute('maxLength');
}
```

// now calling inputTags shows each field but also we can start to type and break the maxlength

(COULD ALSO SHOW BUGMAGNET FOR INJECTING TEXT HERE??)

- CLEAR CONSOLE

Ctrl + L or click Circle stp icon in console - alternatively type clear(); in the console!

SOURCES PANEL

(ABOUT PAGE)

* Local overrides (CSS)

Sources, >> Enable local overrides > select folder to save css changes

(SWEETS PAGE)

Show home page then show sweet page which is broken...

* Local overrides (JS) (KEEP OVERRIDES ENABLED)

Fix shopping cart - currently appends to string, need to debug and fix - it just currently adds on string on to end of other, adding + before each variable allows a calculation

Now alter local override to show it works with the fix!

totalWithShipping = +subTotal + +shippingCost; (REMEMBER TO SAVE!!!)

(YOUR ACCOUNT PAGE)

* Snippets (https://developers.google.com/web/tools/chrome-devtools/snippets)

- here you can store JS snippets and run by right clicking the snippet and executing in the console

// get a list of element properties in tabular format

console.table($$('input'));

NETWORK PANEL

- View traffic
- Filter Traffic
- Throttling
- Preserve Log
- See error status
- View message and response and
- Right click
    - Open in sources
    - Copy as cURL, HAR
    - Block request - exclude css, javascript, etc.

* Throtlling

Enables you to render the application with a preset connection type, create your own connection type or see how your page works offline (you can create offline content for your users)

PERFORMANCE / MEMORY PANEL

We're not going to cover both of these panels in this section.

APPLICATION PANEL

* Cookies

Cookies are simple text files that are stored in a user’s machine. More often than not any sensitive data within cookies will be encrypted but it isn't always the case. Sometimes it's worth modifying cookies, starting to experiment to see if you can find vulnerabilities.

* Web SQL

In November 2010, the W3C announced that Web SQL database is a deprecated specification. It was a recommendation for web developers to no longer use the technology as effectively the spec will receive no new updates and browser vendors aren't encouraged to support this technology.

I've seen a number of web applications still using WebSQL in order to use various features both online and off-line.

The purpose of me showing you this is so that you know where the data is stored if people are using WebSQL.

* Local Storage / Web SQL

Here you can see any local storage your application 'may' be using. Here people may store KVPs

SECURITY PANEL

We're not going to cover this in this session but the security panel makes it a lot easier to see any issues you have with certificates and mixed content.

AUDITS PANEL

* RUN ALL AUDITS on LIGHTHOUSE
* Lighthouse SHOW ACCESSIBILITY
* Best Practices

Show that it picks up the fact WebSQL is being used despite it being deprecated. It also gives help and shows an alternative to use.

MORE TOOLS

CHANGES

Click ... and select 'Changes' to be able to view the changes which have been made - demonstrate a style change

COVERAGE

(ABOUT PAGE)

- COVERAGE (Updated in read-time as you use the page)

Find unused CSS and JS

- Click .cls within elements page and apply one of the class names which was showing red in a .css style sheet and this will now go green within coverage to show it's used using real time updating.

**** CHROME CANARY - You can export the coverage as json (could be useful and used by third part tools)

QUICK JUMP TO MENU

(SWEETS PAGE)

* Screen shots
    * Full page

Cmd + Shft + P - type in capture

Within canary you can take a full size screenshot of the page. This used to require an extension in order to do this.

(SWEETS PAGE)

    * Element

Right click and inspect element, Cmd + Shft + P - 'Capture node screenshot' , this allows you to store a screenshot of the selected element

* Disable JS, always good to check how pages look or function with it disabled - government projects have to be built to function if JS is disabled!

EXTENDING DEV TOOLS

* aXe

Extend Chrome DevTools to allow you to inspect page markup

CHROME CANARY

Chrome Canary is Google's cutting edge web browser aimed at developers, experienced techies, and browser enthusiasts. If you enjoy experimenting with new web browsers, then it might be for you.

Apps for Exercises:

For each section of the session I originally figured we could use this example sweet show app https://sweetshop.netlify.com/ where attendees could follow along to get a feel for how to use a tool/panel within DevTools. After this they would then move on to a challenge from the testers playground http://playground.epizy.com/ . I'm also keen after our discussions to use all the other advanced stuff you showed for the people who want to push on.

- https://sweetshop.netlify.com/
- https://sweetshop.vivrichards.co.uk/
- http://playground.epizy.com/

The code for the sweet shop can be found here:
https://github.com/vivrichards600/sweet-shop

It's built using just HTML/JS

The code for the testers playground can be found here:

https://github.com/vivrichards600/testersplayground/

It's built using PHP

I'm keen for us to come up with new challenges or perhaps things to add to the sweet shop. Feel free to make these changes or if you have ideas I'm happy also to make the changes. I'm also not precious on any of the above of what we show but just want to do something where we can ensure participants feel comfortable using DevTools and understand some of the great tools within it.

Additional Apps

http://todomvc.com/

- Useful because different implementations support different types of manipulation e.g. in VanillaJS we can use Element Pane to amend the text on elements but that doesn't amend in localstorage.

https://thepulper.herokuapp.com

- I will make notes on how this can be useful to people using the dev tools, but it basically provides a CRUD app for working with and exploring.

https://www.compendiumdev.co.uk/games/buggygames/

- I need to package up buggy games as a heroku deploy, but if I don't have time prior to the workshop we can use the apps on the page.
- I will create notes for the different games and how they might be useful to people using the dev tools.

https://testpages.eviltester.com

- A growing list of test pages which also links to other sites to test on.

Galaxians

Found this little Galaxians game which is quite nice and hackable:

http://michal.biniek.pl/games/galaxian.html

Only bug I found is the rendering of flags where they get too long for the screen - but it is hard to play the game to that level.

Bug can be triggered from the console:

`oLevel.iLevel=21`

Might be a useful example to show how the JavaScript can help test the untestable.

Here are some simple bots for the game

~~~~~~~~
var infinilivesBot = setInterval(function(){if(oLives.iLives<3){oLives.add();}},1000);
~~~~~~~~

auto fire

~~~~~~~~
var autofireBot = setInterval(function(){MyShip.shoot();},200)
~~~~~~~~

~~~~~~~~
machineGunner = setInterval(function(){new bullet({x: MyShip.x + aParams.width/2, y: MyShip.y - aParamsBullet.height, direction: -1});},200);
~~~~~~~~

And clear it with

~~~~~~~~
clearInterval(machineGunner)
~~~~~~~~

Real world examples using DevTools

We spoke about showing real world examples where DevTools can be handy. Below I've quickly noted down some of the things I've used it for - maybe we can structure all this brain dump out in a better way..

Device Toolbar

Ebay listings respond differently based on the device size when you reload, but not necessarily on resize.

Twitter redirects to mobile.twitter.com if it detects mobile headers

Sites for network tab:

- Twitter - since it makes AJAX calls

Sites for difference between DOM and Source

- https://twitter.com
- View page source on Twitter makes it look like a site powered by magic

Sites with embedded JSON for experimenting with JSON in console:

- https://www.bbc.co.uk/

Sites where view source can be useful:

- Spectator - https://www.spectator.co.uk/
    - Only allows 5? Article views per month
    - Click on 5 articles, don't read them
    - Click on a 6th and the text of the article will be 'hidden'
    - Refresh the page and note that the article is diplayed and then hidden
    - Article text is not present in the DOM - deleted by JavaScript, but possible to read in the source
    - Also - switching off JavaScript makes viewing article possible - demonstrates why client side validation and processing only is 'dangerous'
    -

Jira

- modifying a story if you do not click save the page has a pop up which tries to force the user to click a button to refresh. Using the inspect tool you can find the blanket/popup div and remove it from the DOM, copy the text from the input and then reload - this way you don't lose your work.

JS Booklets

- you mentioned you had a booklet to remove Twitter adverts

I'm going to try and package a lot of these in a Chrome extension to make them easier to share

Start Twitter Bot - call this first

~~~~~~~~
~~~~~~~~

javascript:(function()%7Bvar%20twitterAdCloserBot%20%3D%20setInterval(function()%7B%20var%20advert%20%3D%20document.querySelector(%22div.promoted-tweet%20button.js-action-dismiss%22)%3B%20if(advert)%7Badvert.click()%3B%7D%20var%20didYouMiss%20%3D%20document.querySelector(%22div.recap-header%20button%5Bdata-feedback-type%3D%27SeeFewer%27%5D%22)%3B%20if(didYouMiss)%7BdidYouMiss.click()%3B%7D%20%7D,200)%3B%7D)()
~~~~~~~~

End Twitter Bot - call this to stop the bot running

~~~~~~~~
javascript:(function()%7BclearInterval(twitterAdCloserBot)%7D)()
~~~~~~~~
~~~~~~~~