Welcome to Chrome Developer Tools: Raiding the Armory

Slides and project files:

http://bit.ly/wickedweapons

gregmalcolm

I’m Greg Malcolm, I work as a consultant at ICC, a Columbus based IT consulting firm.

Twitter: @gregmalcolm
Speakerdeck: https://speakerdeck.com/gregmalcolm/chrome-dev-tools
Code:
https://github.com/gregmalcolm/wacky-wandas-wicked-weapons-frontend
https://github.com/gregmalcolm/wacky-wandas-wicked-weapons-api

Caveat: The demo code was written in a hurry. It is indeed crude, buggy and not a good example of how to write a frontend. Which suits the purposes of this presentation just fine. Don’t judge me! :)

Wacky Wanda's Wicked Weapons

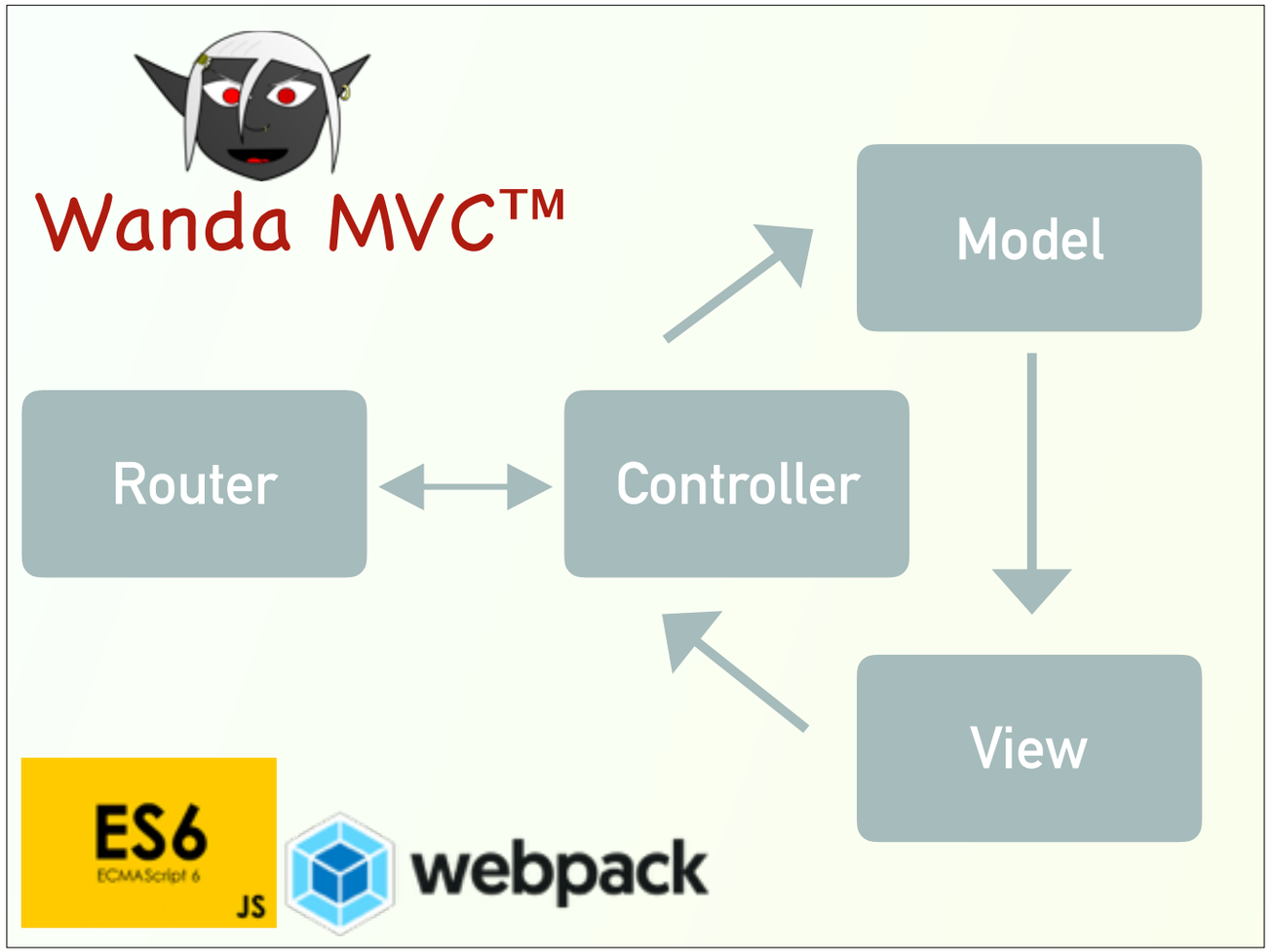
Putting The Whack Back In Wacky!

Today's presentation is sponsored by Wacky Wanda's Wicked Weapons.

“Putting the Whack back in Wacky!”

I’ve been contracted to work on bugs on the new store front for Wicked Wanda’s Wicked Weapons. To save time I’m going to work on this while giving the demo, hope you don’t mind!

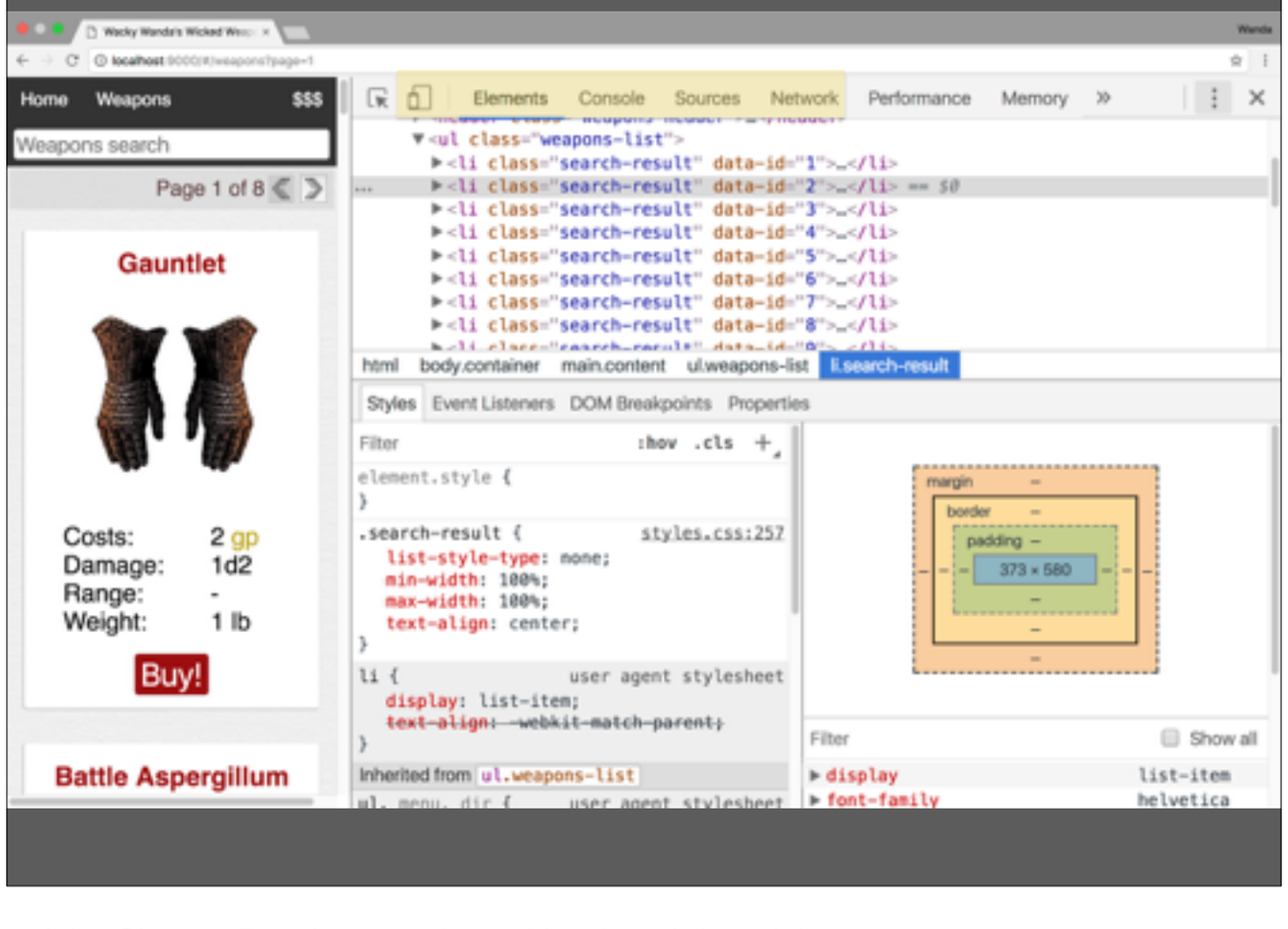
<I give a quick demo of the app at this point and show that it needs work>



Wacky Wanda's Wicked Weapons is built using plain old ES6 Javascript and CSS. We do however use Webpack to compile the resources and assets for redistribution and squashing the javascript and css down to one file each.

We're not using any kind of Javascript framework, but we are still structuring the code in a variation of the Model View Controller pattern. There is also a Router bolted to this concoction to support single page application behavior, allowing the app to respond to url changes.

I nickname this thing Wanda MVC.

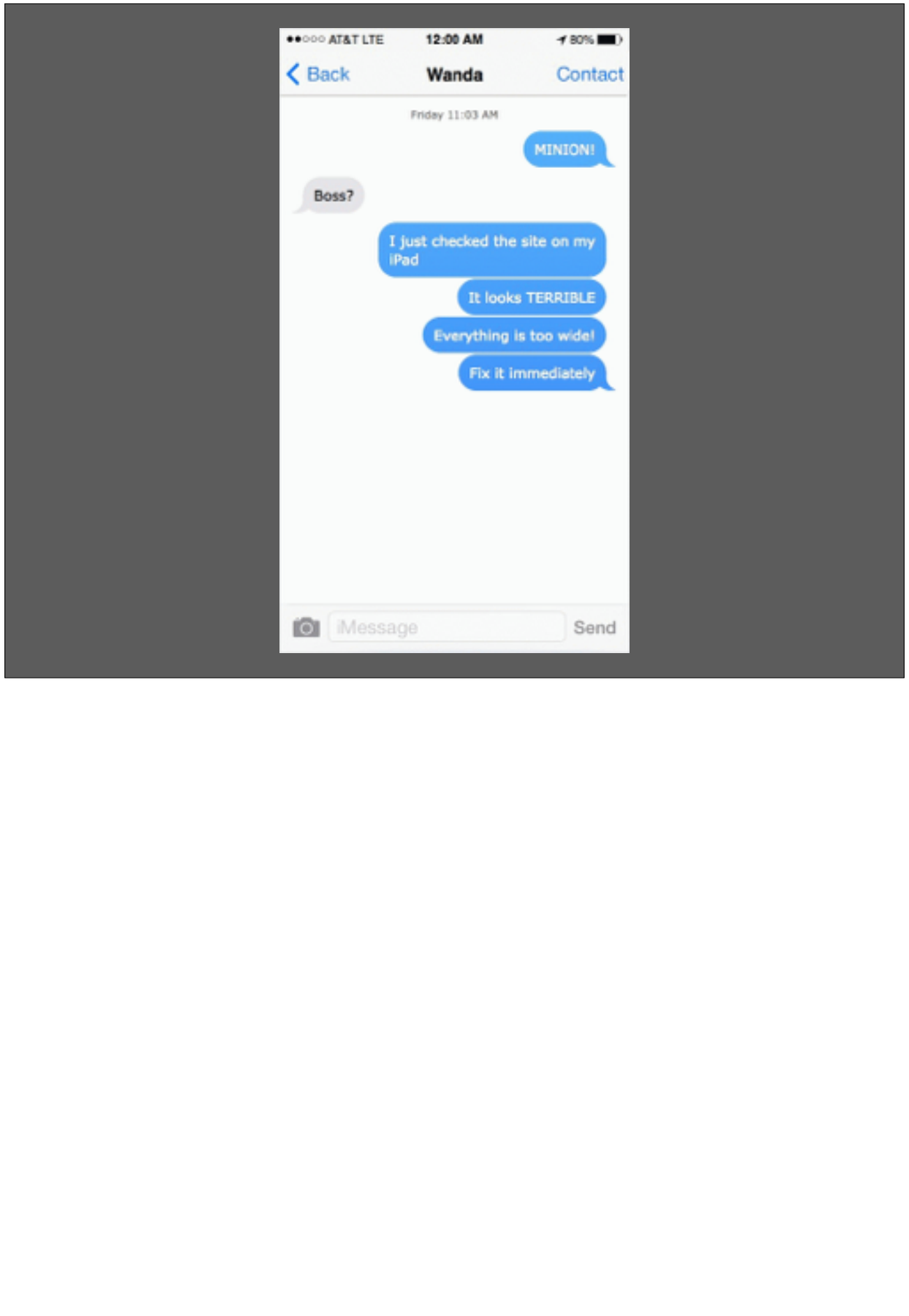


We'll be focussing on the leftmost panels of the Chrome Developer tools working from left to right.

We'll start by fixing responsive issues in the Device Toolbar. Next we'll make use of the Elements panel to fix CSS styling issues. We'll then move onto the Console for analytical work then Sources Panel for debugging code. Finally we'll look at the Network panel to analyze Load performance and solving Ajax problems.

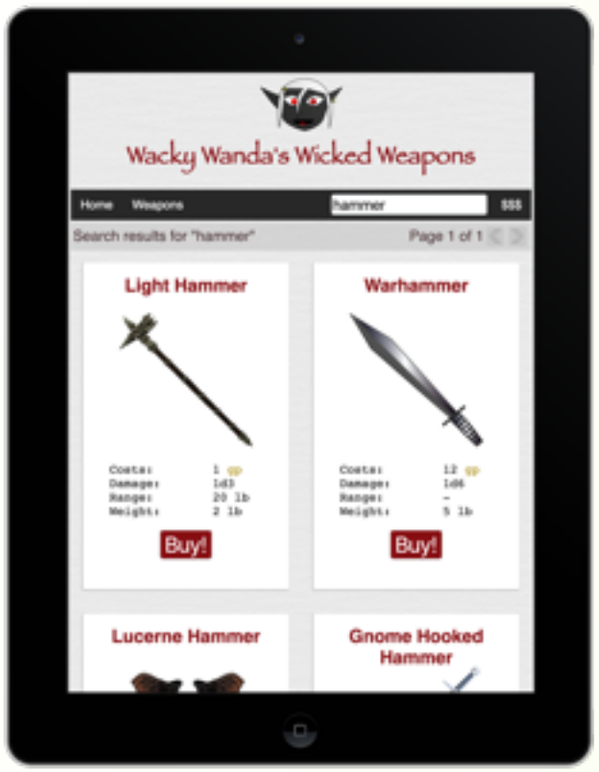
At the beginning we'll break into things very gently, but the topics will become more advanced as we travel from left to right. So if it's a little too slow for you at first hang in there, it'll get better.

Hang on, Wanda is texting me. I'd better take this.

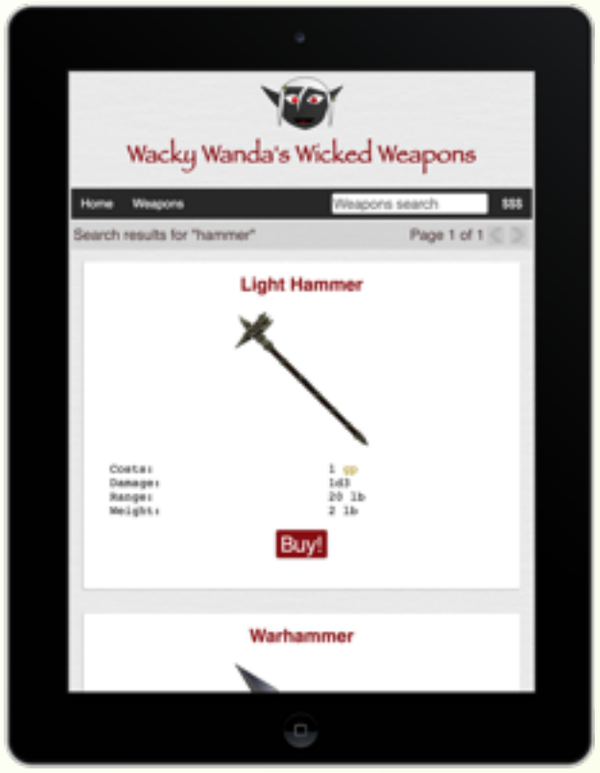


Ok, I’d better get right on this.

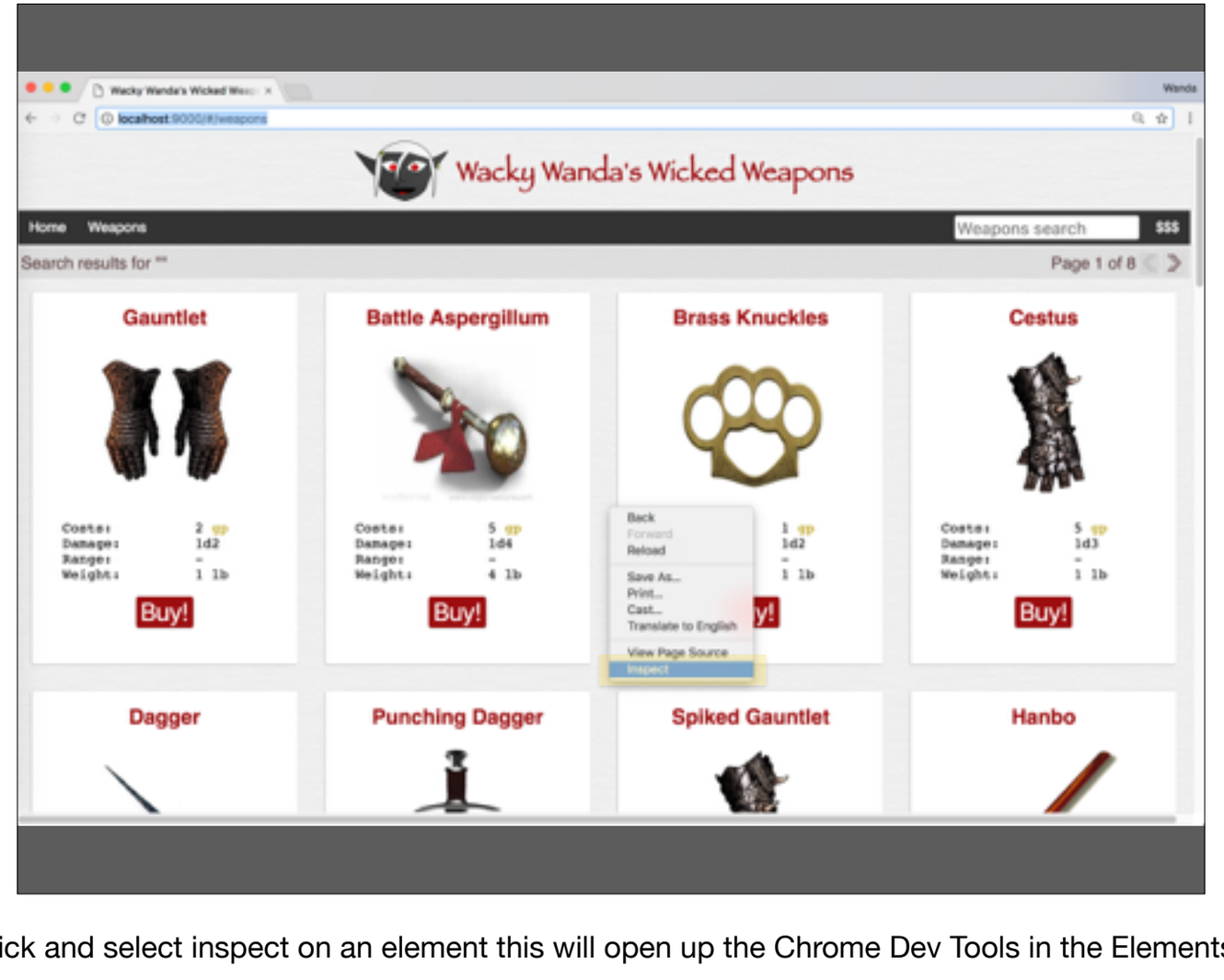
What Wanda Wanted



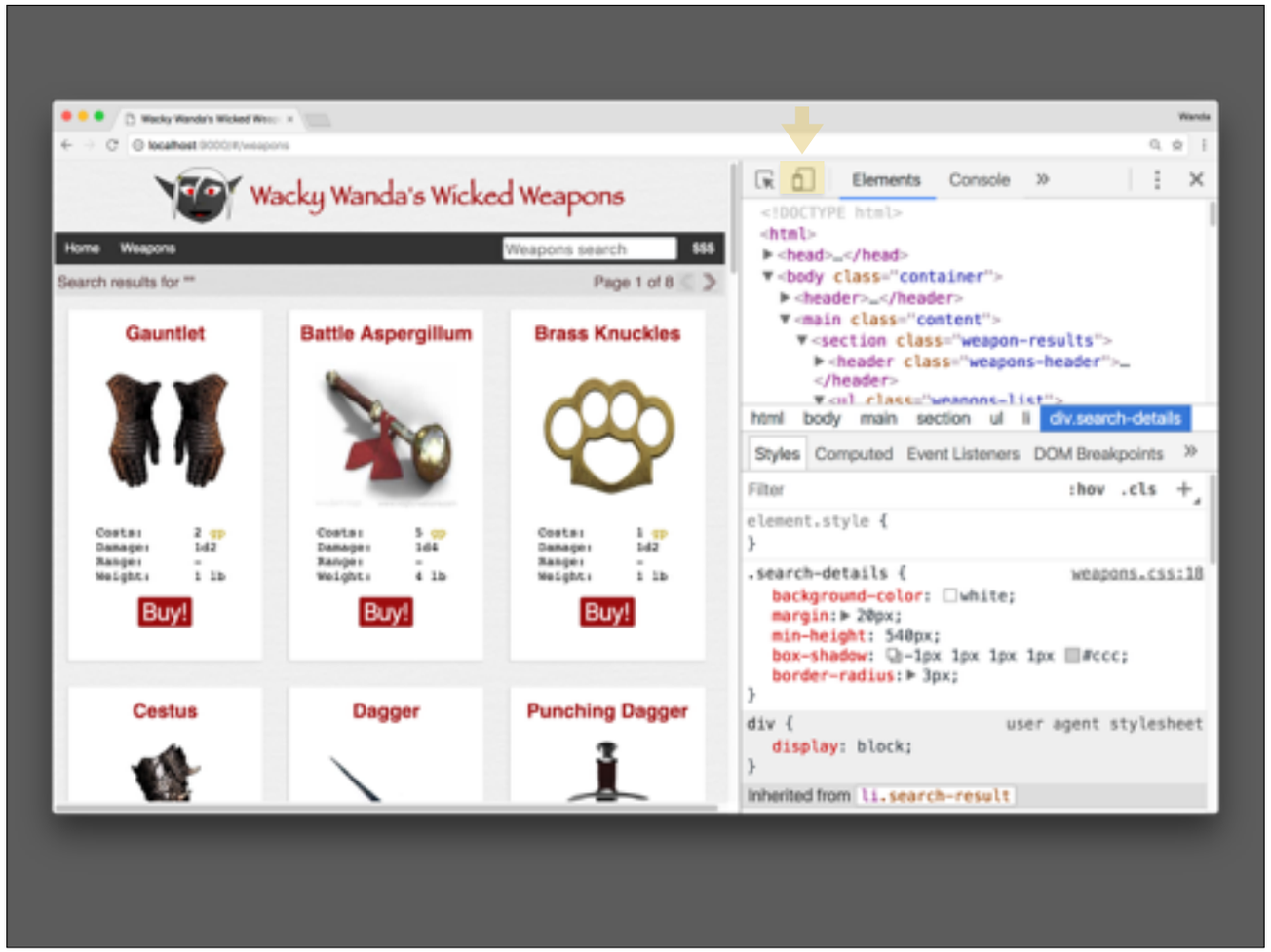
What Wanda Got



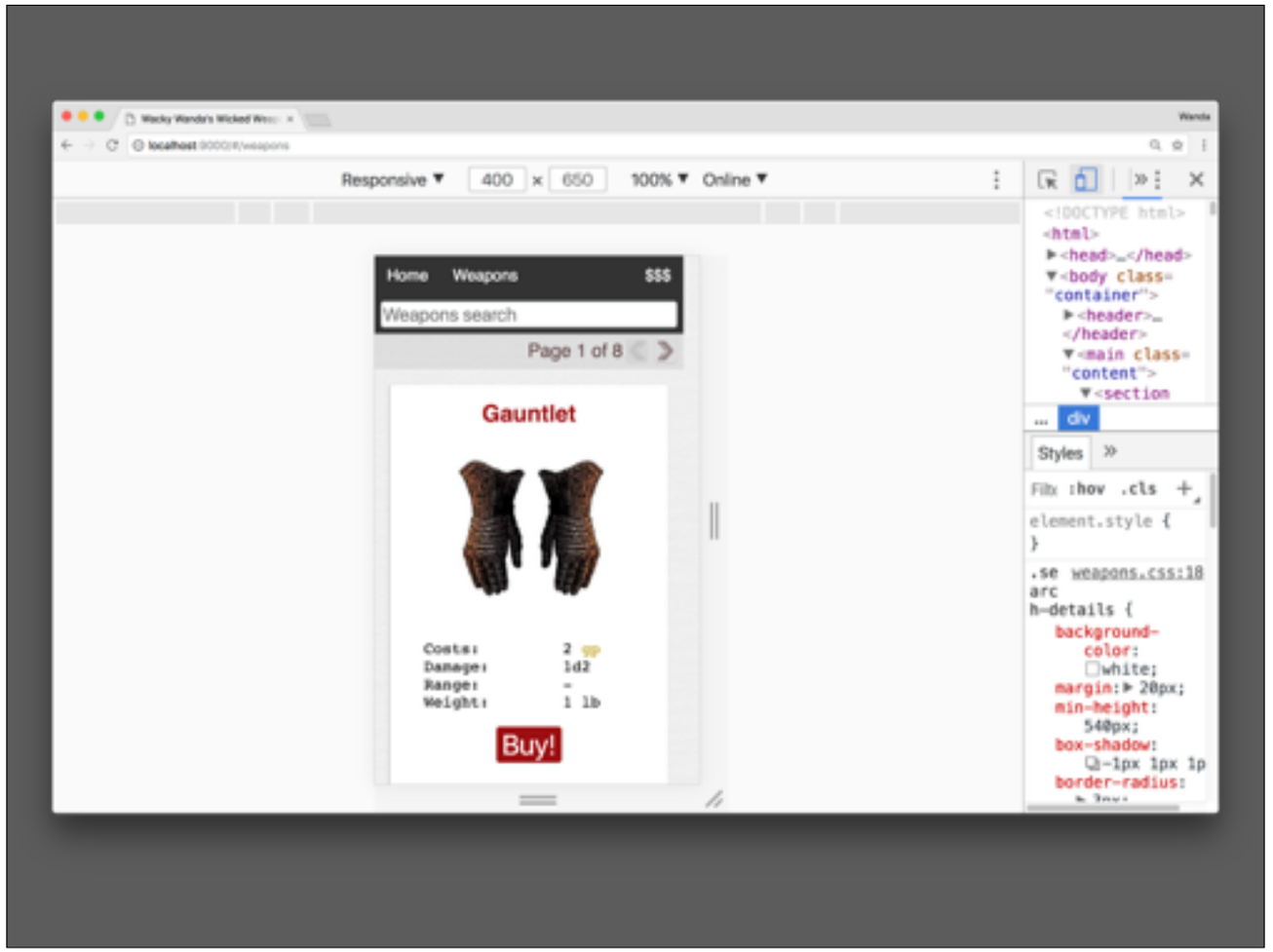
Let’s take a look at the requirements. On the iPad there are supposed to be 2 tiles visible. But in practice it’s only showing one. Pretty ugly…

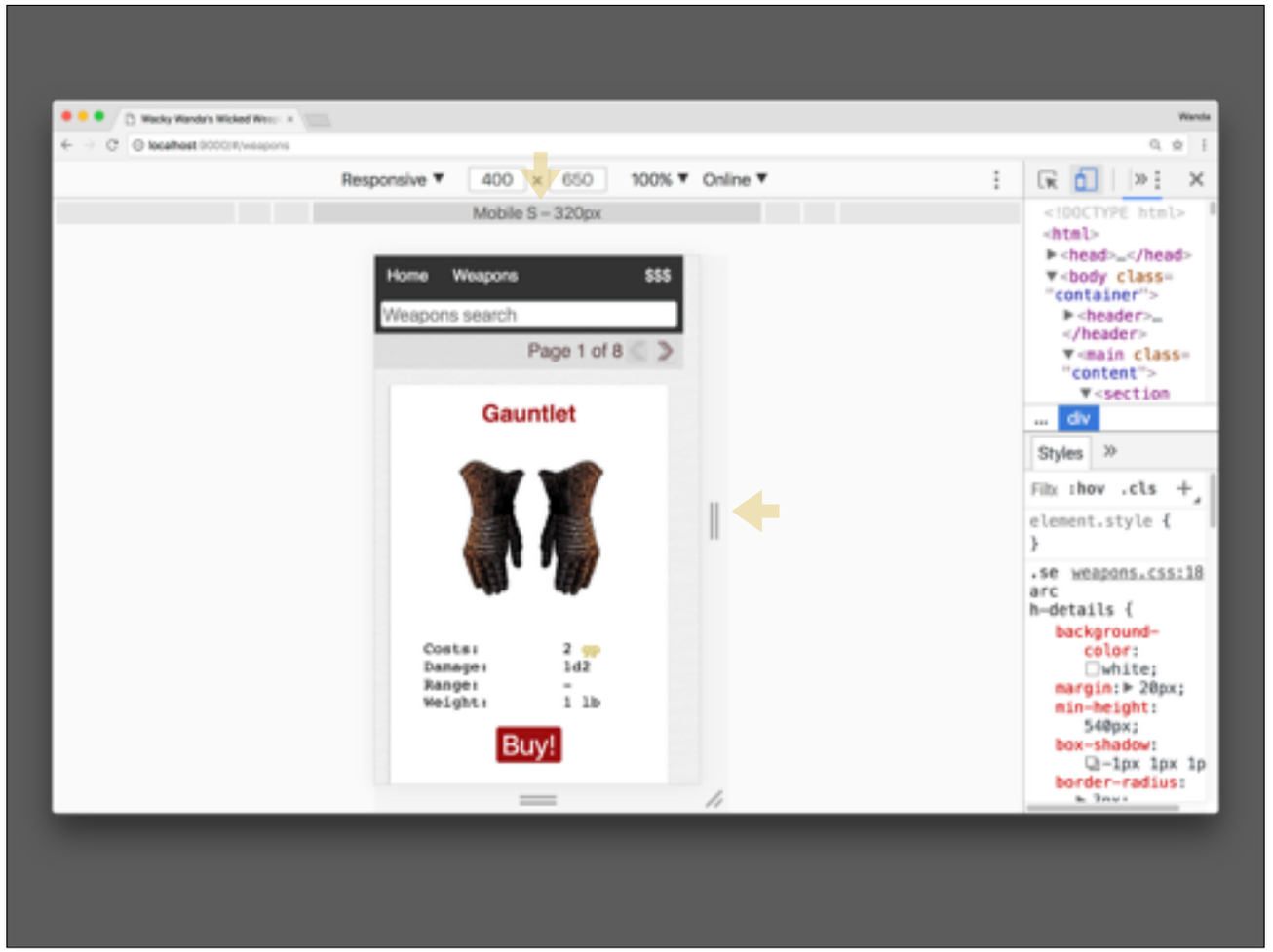


Ok, back to the storefront. If we right click and select inspect on an element this will open up the Chrome Dev Tools in the Elements Panel.

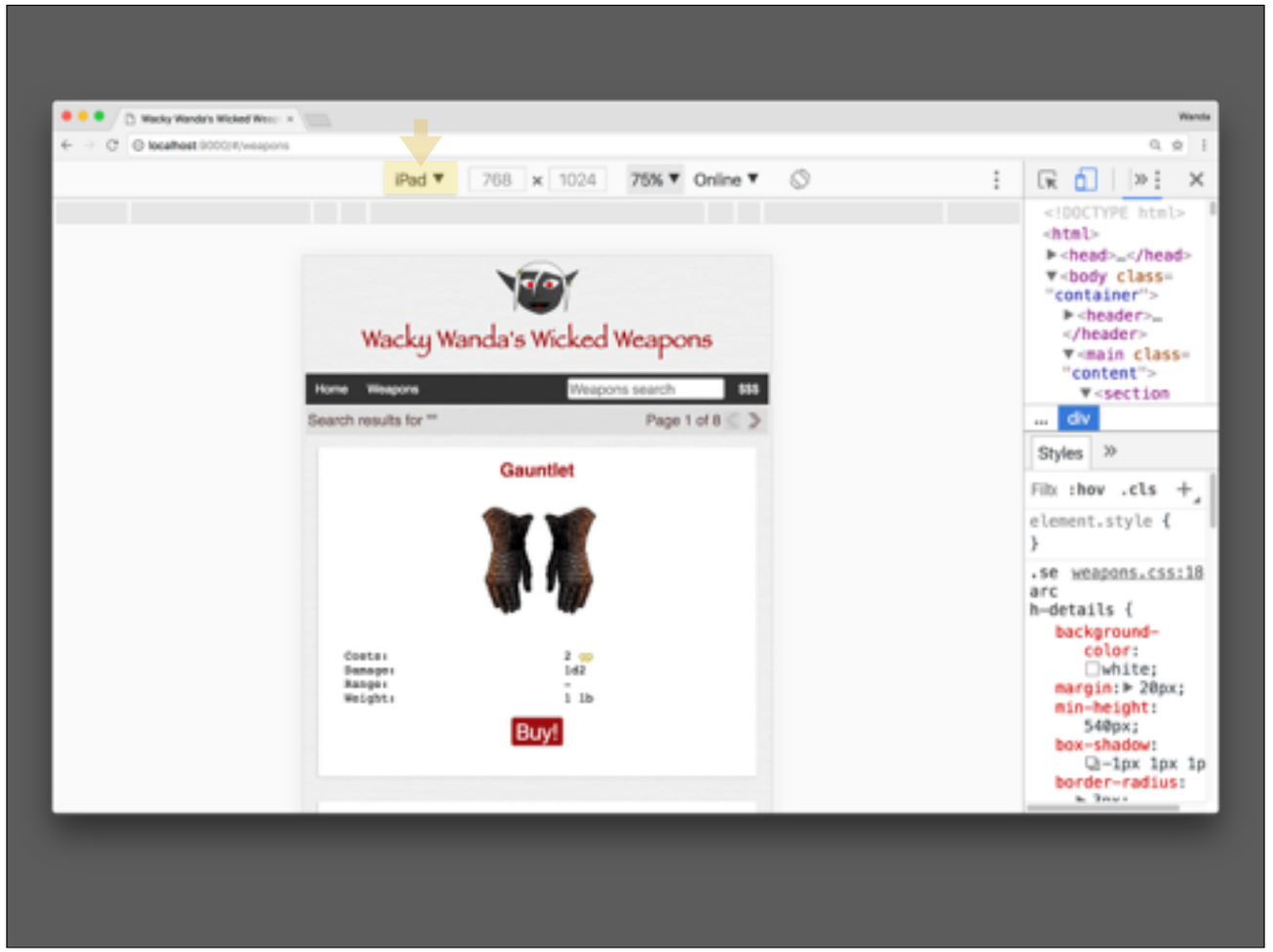


As this is a device based issue we'll need the Device Toolbar turned on. We can do that by click the device icon next to "Elements" in the top of the Dev Tools.


Responsive
400
650
100%
Online
Home
Weapons
$$$
Weapons search
Page 1 of 8
Gauntlet
Costs: 2
Damage: 1d2
Range: -
Weight: 1 lb
Buy!
<!DOCTYPE html>
<html>
<head>…</head>
<body class="container">
<header>…
</header>
<main class="content">
<section
div
Styles
element.style {
}
.se weapons.css:18
arc
h-details {
background-
color:
white;
margin: 20px;
min-height:
540px;
box-shadow:
-1px 1px 1p
border-radius:

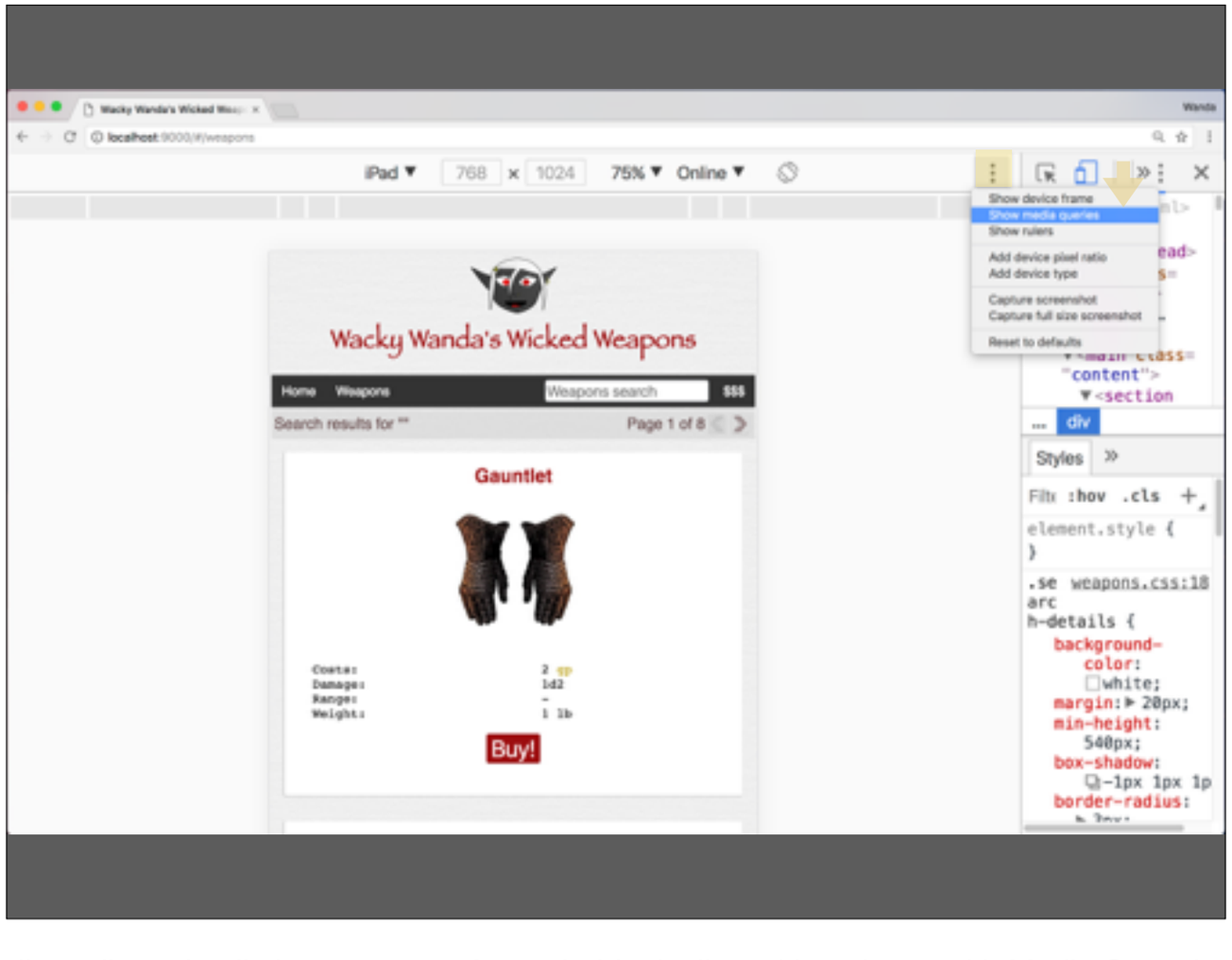


Initially we’re in freeform Responsive mode. We can adjust size with the slider on the right. If we hover over the bar at the top we can see the initial size is slighter smaller than that of a Small Mobile device.

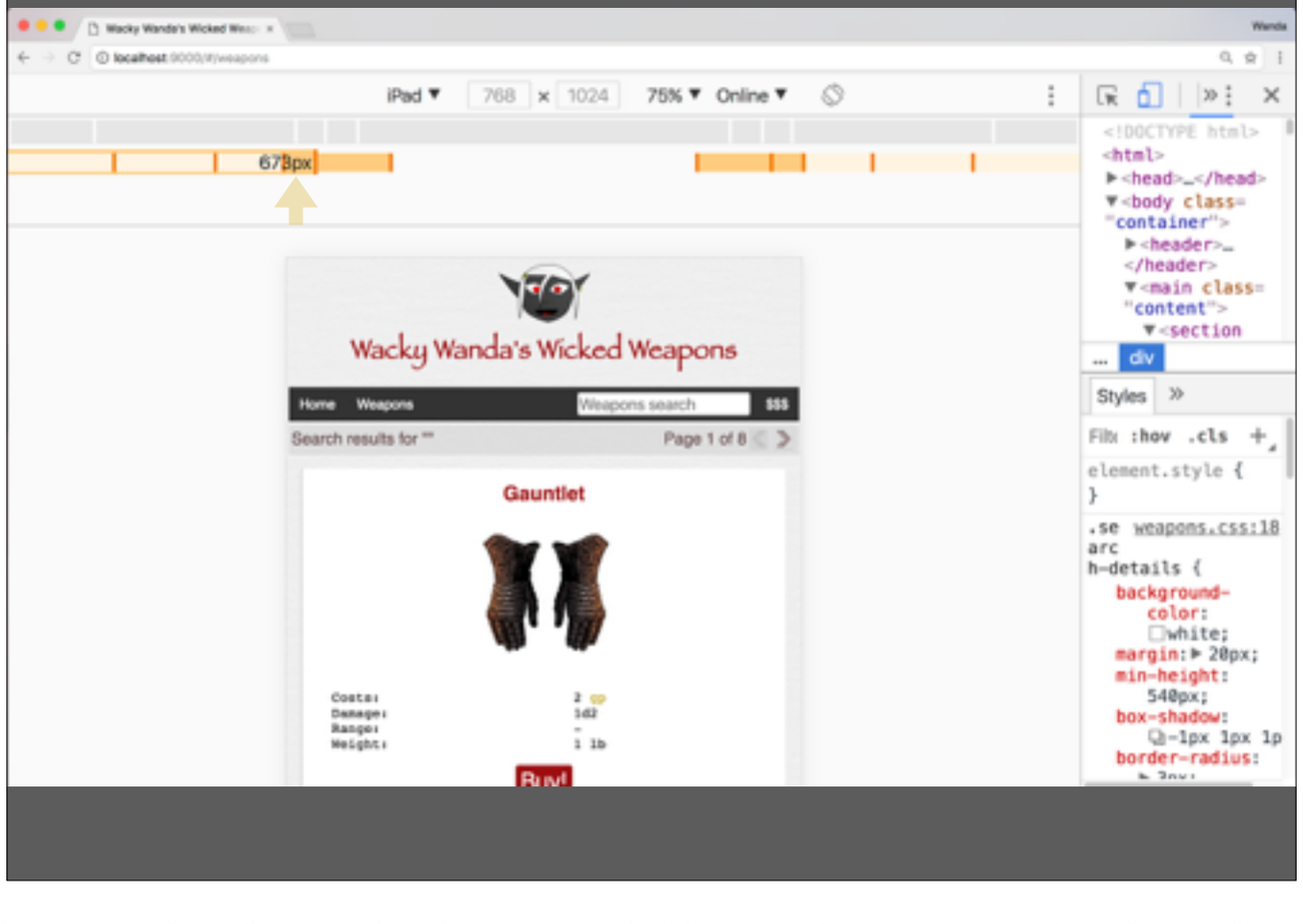


We need to reproduce Wanda's iPad issue, so let's come out of freeform Responsive mode and select the iPad preset instead. This is available from the upper left menu.

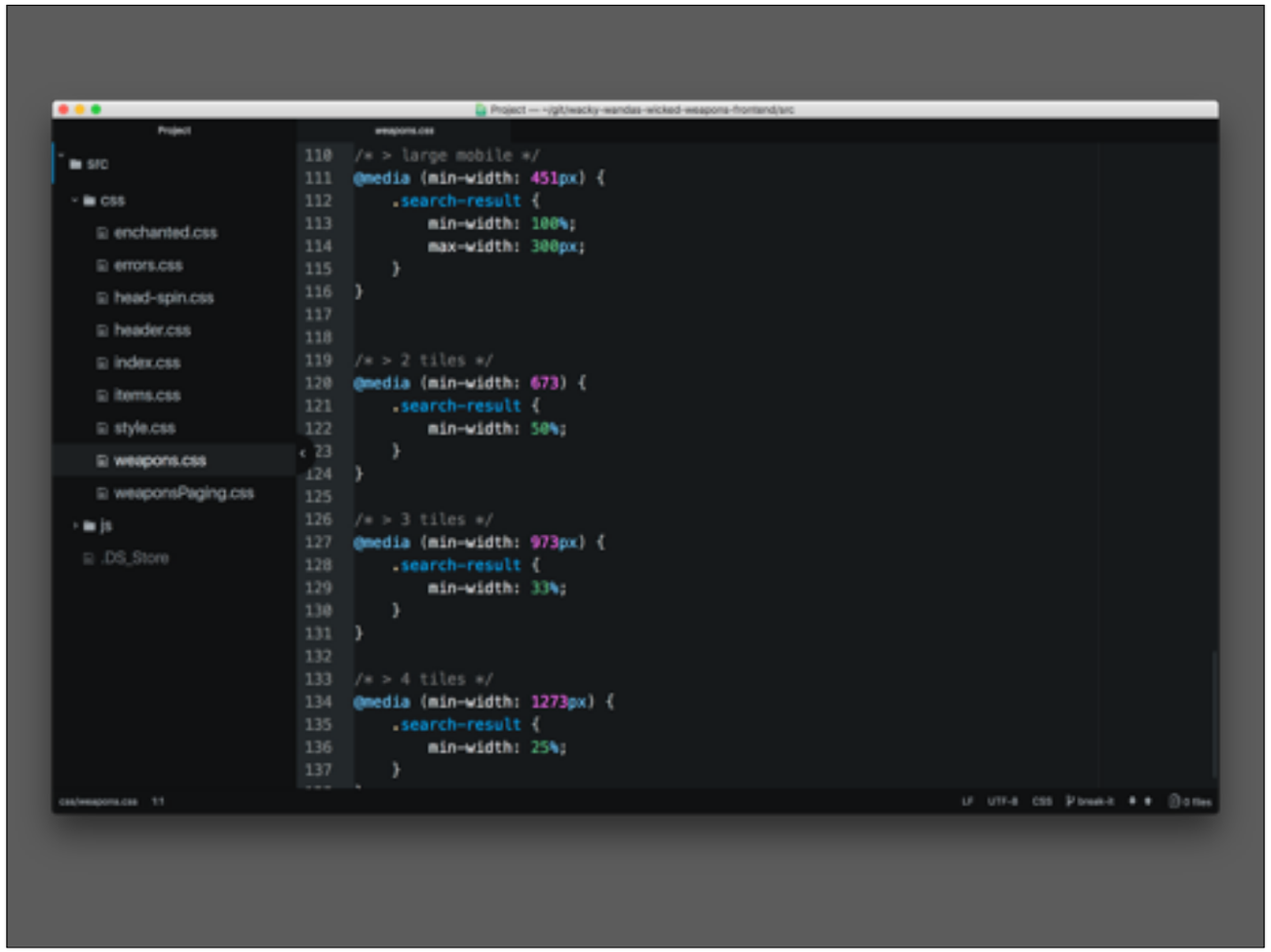
Sure enough the tile is too wide.



We've already seen the app show single tile, 3 tile and 4 tile layouts, so we're probably dealing with an issue with Media Query breaking points. It might be helpful to take a look at which Media Querie break points are in play on this page. We can do that by selecting the triple dot menu on the top right and selecting "Show media queries"

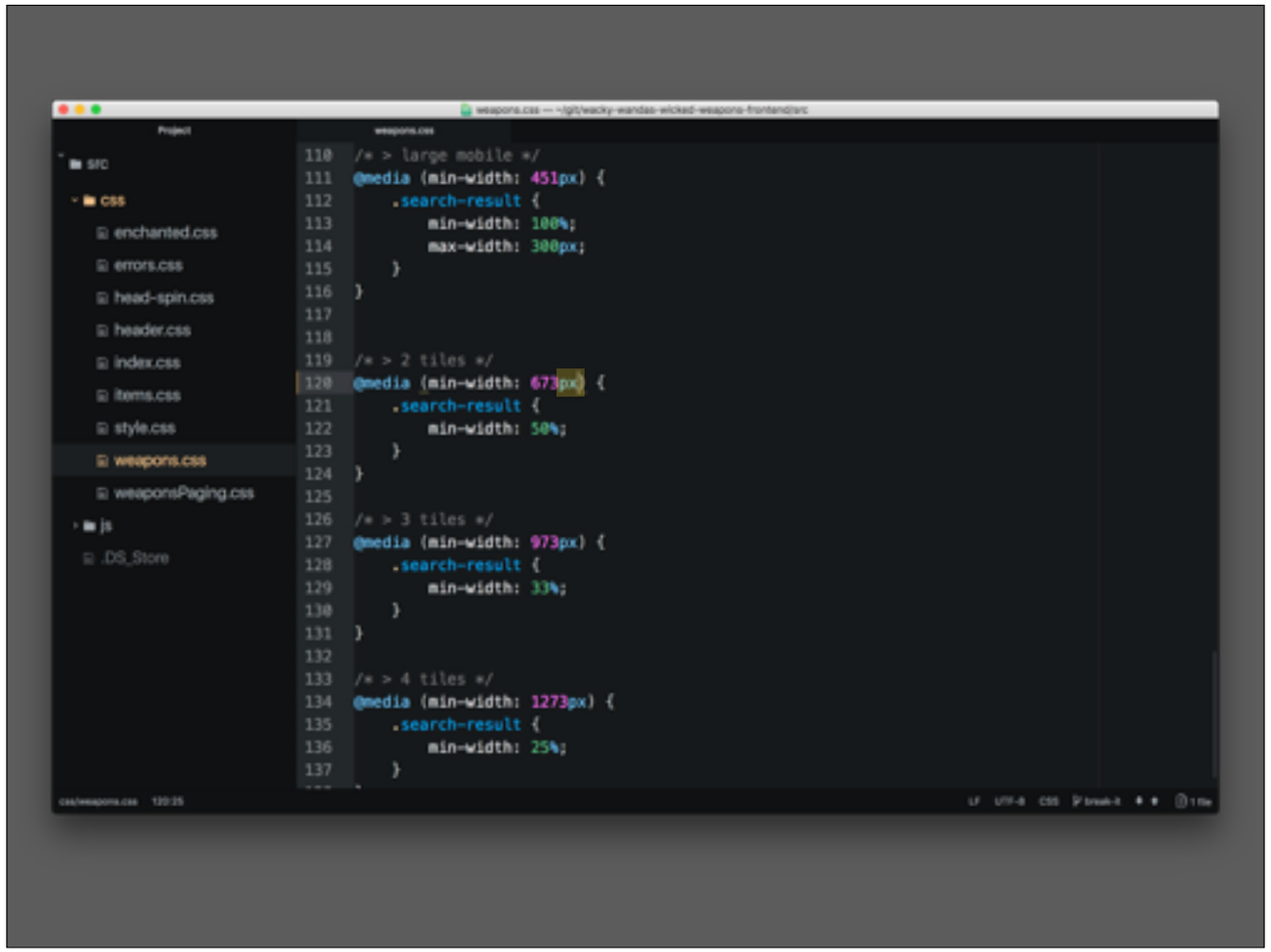


Judging by the dark beige the iPad falls into the 2nd media query break at 673 pixels. Hovering the mouse over the ruler confirms it.

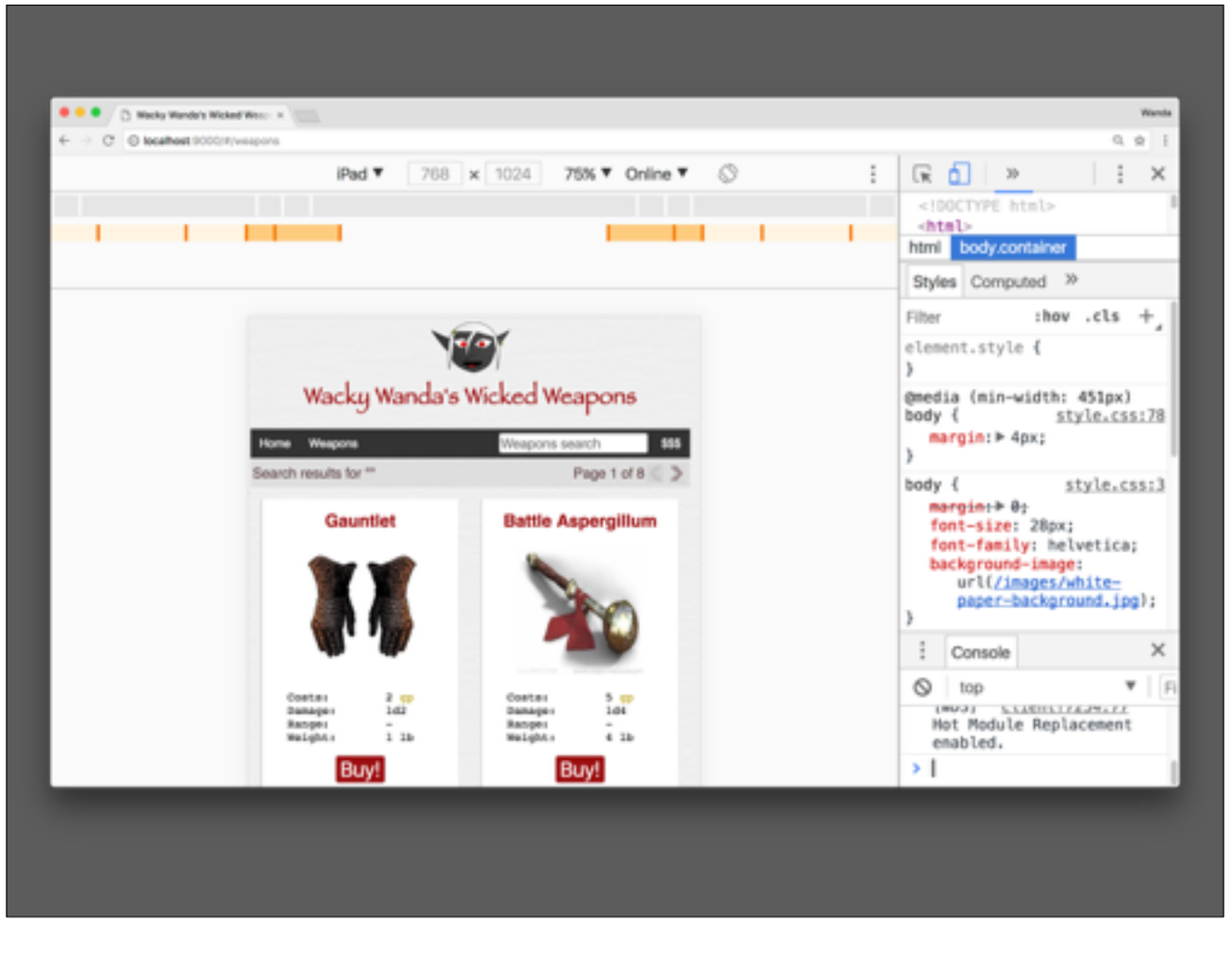


We'll go to my code editor, Atom to fix this. The css for managing the tiles is in weapons.css. The file starts with mobile 1 tiling sizing with media query breaks for when the page grows bigger. It's using a flex grid with the min-width relative to how many tiles we need. So 100% for 1, 50% for 2, 33% for 3 etc.

See the problem with the 2 tile configuration?



It turns out the “px” part got left of min-width: 673px.



Yep, that did it! the iPad now shows 2 tiles as per the specifications.

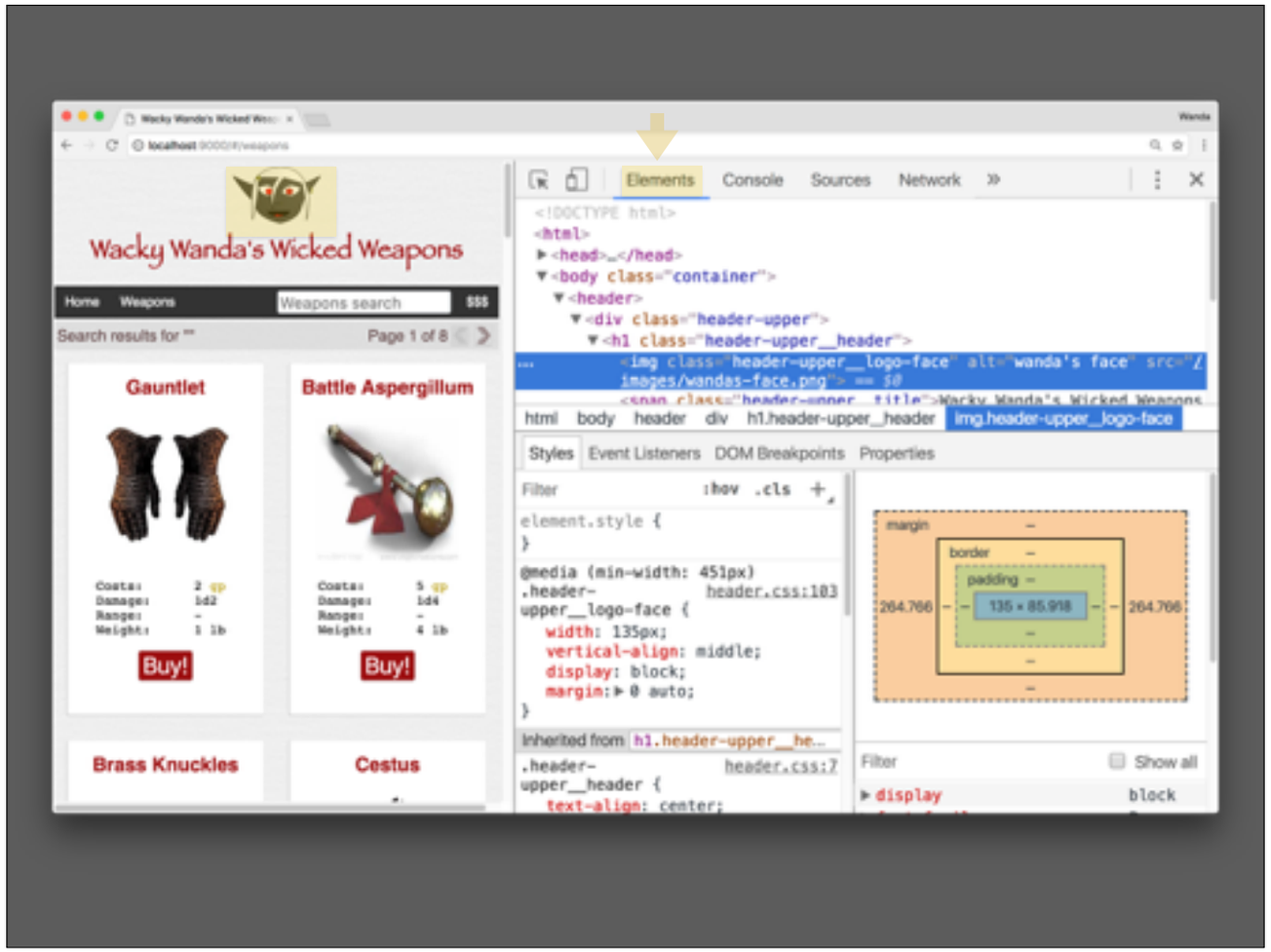
Elements Panel

Make Wanda's Head Spin

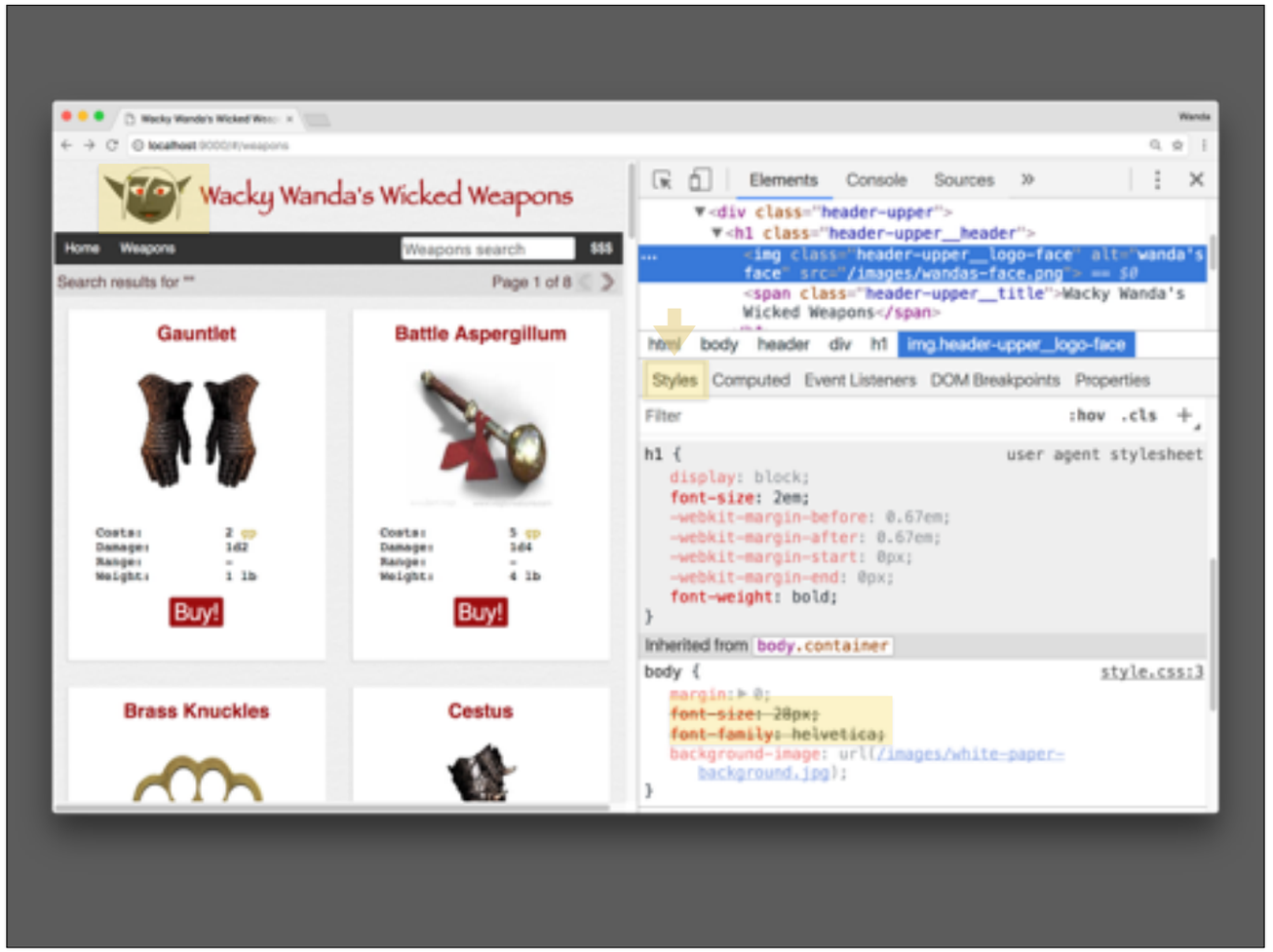
No, really!

Next up, Wanda wants us to make her head spin. No really.

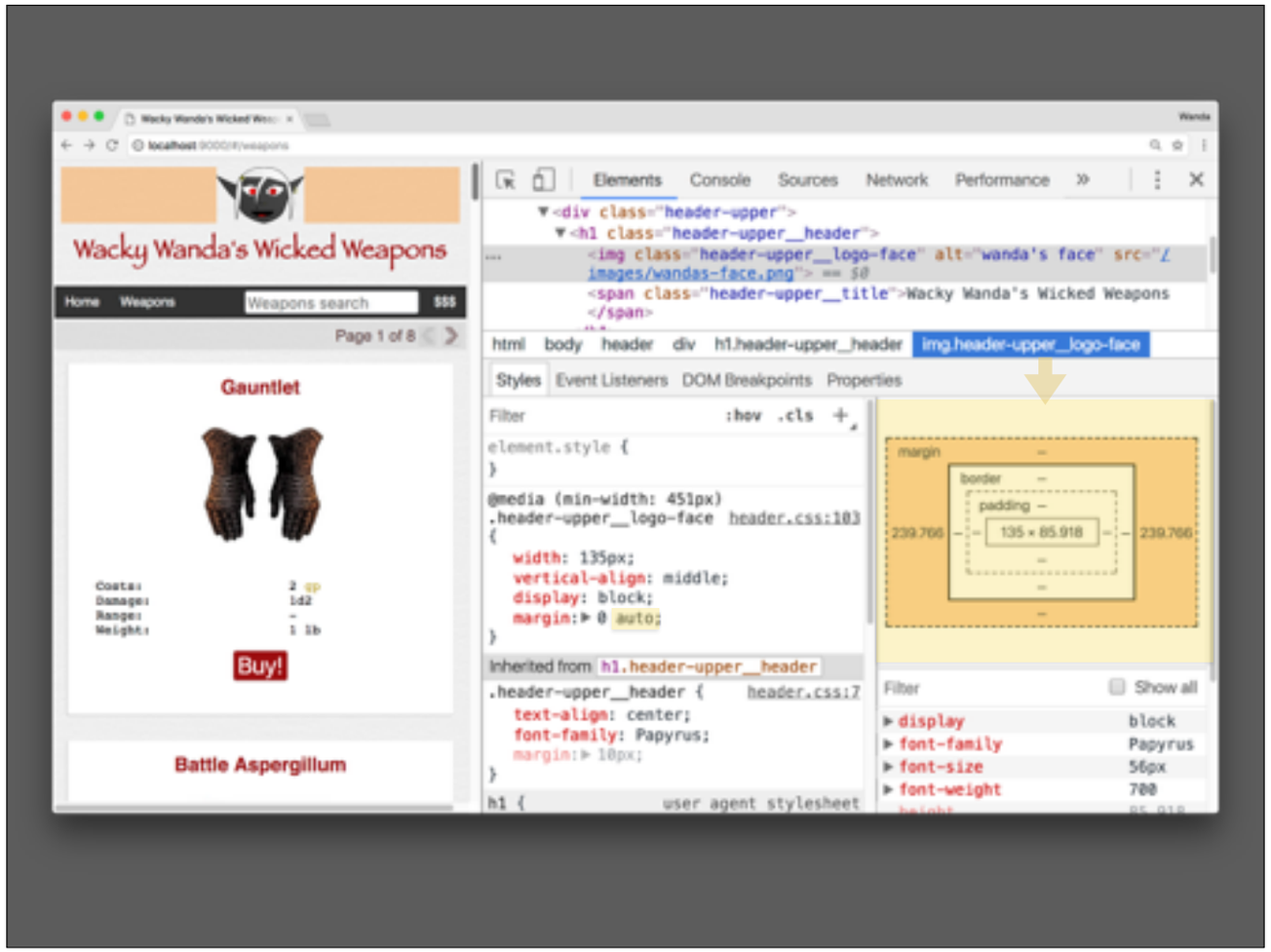
Well… she is the boss…



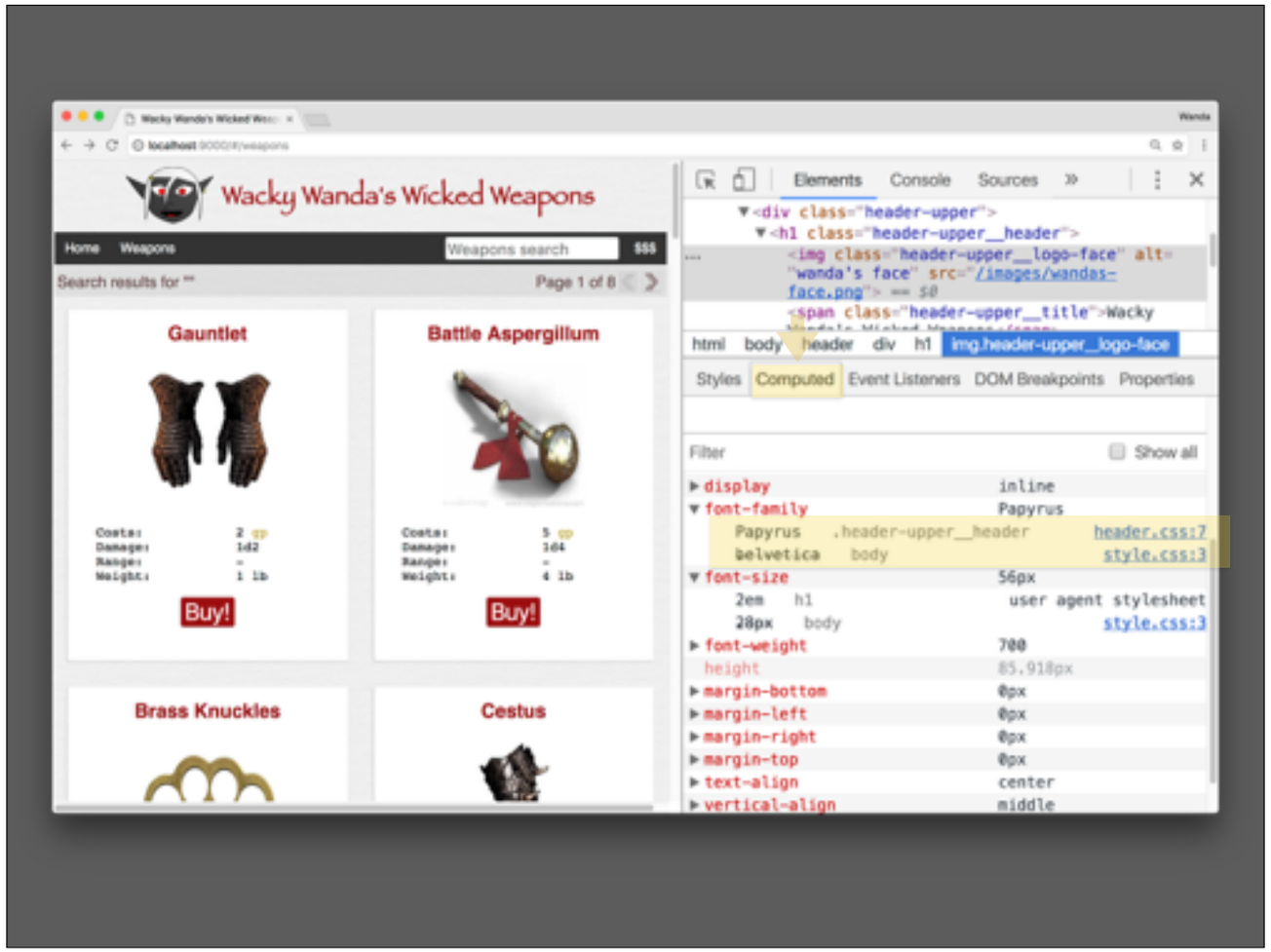
This feature is purely for mouse driven inputs, so we might as well turn off the device toolbar. This is purely a Styling problem so we're going to work exclusively in the Elements Panel. I'll go ahead an Inspect Wanda's face which will take me right there.



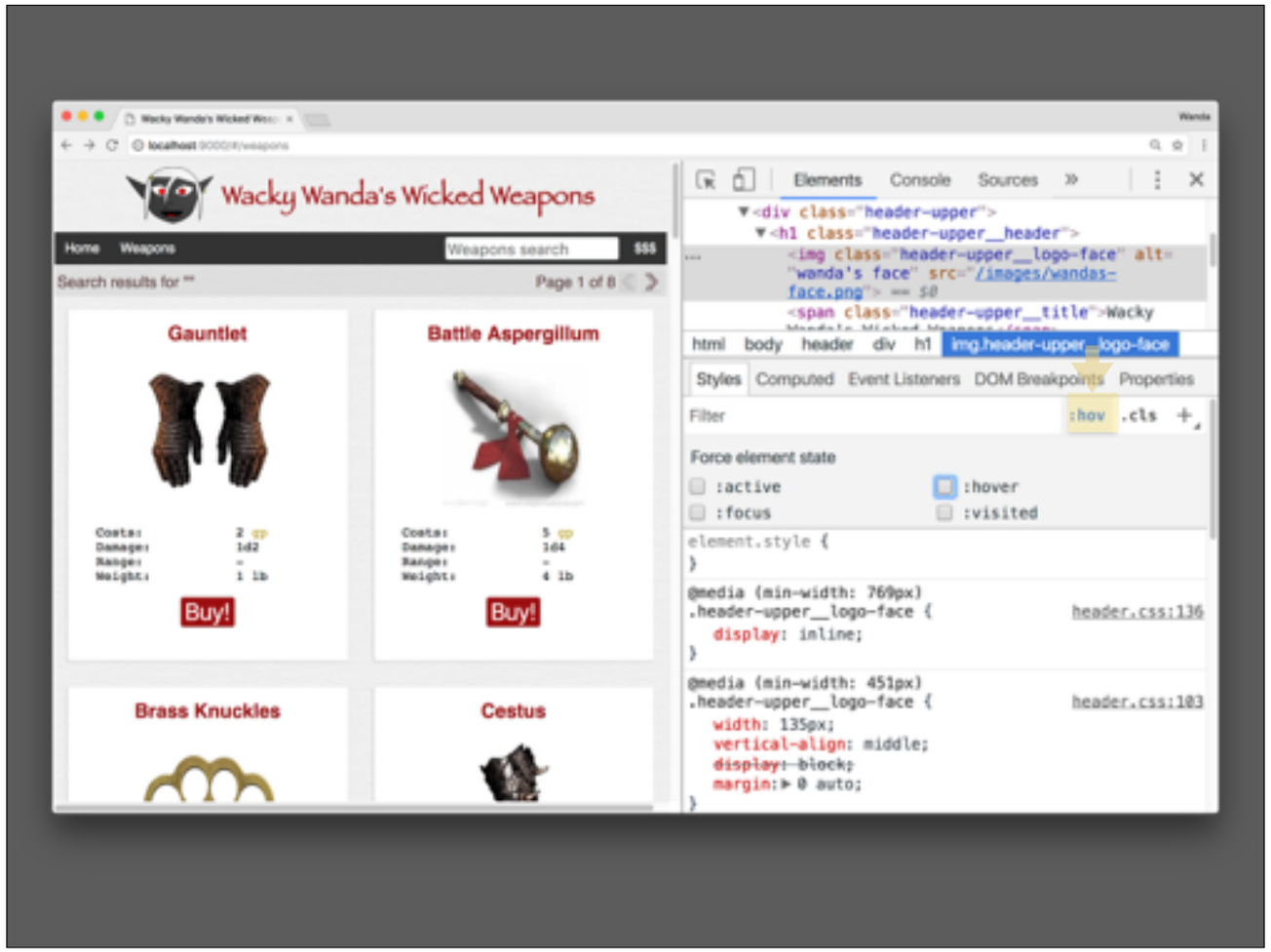
Before we get to work let's take stock of the key features on this page. In the bottom left column we have a list of all active styles for the selected img element. Some of the styles will be inherited. Some inherited styles will be overridden. For example, we can see that the font-size and font-family body styles have been override with more specific styles. Note that each style is live editable and that they each include a hyperlink to the stylesheet defining the rule.



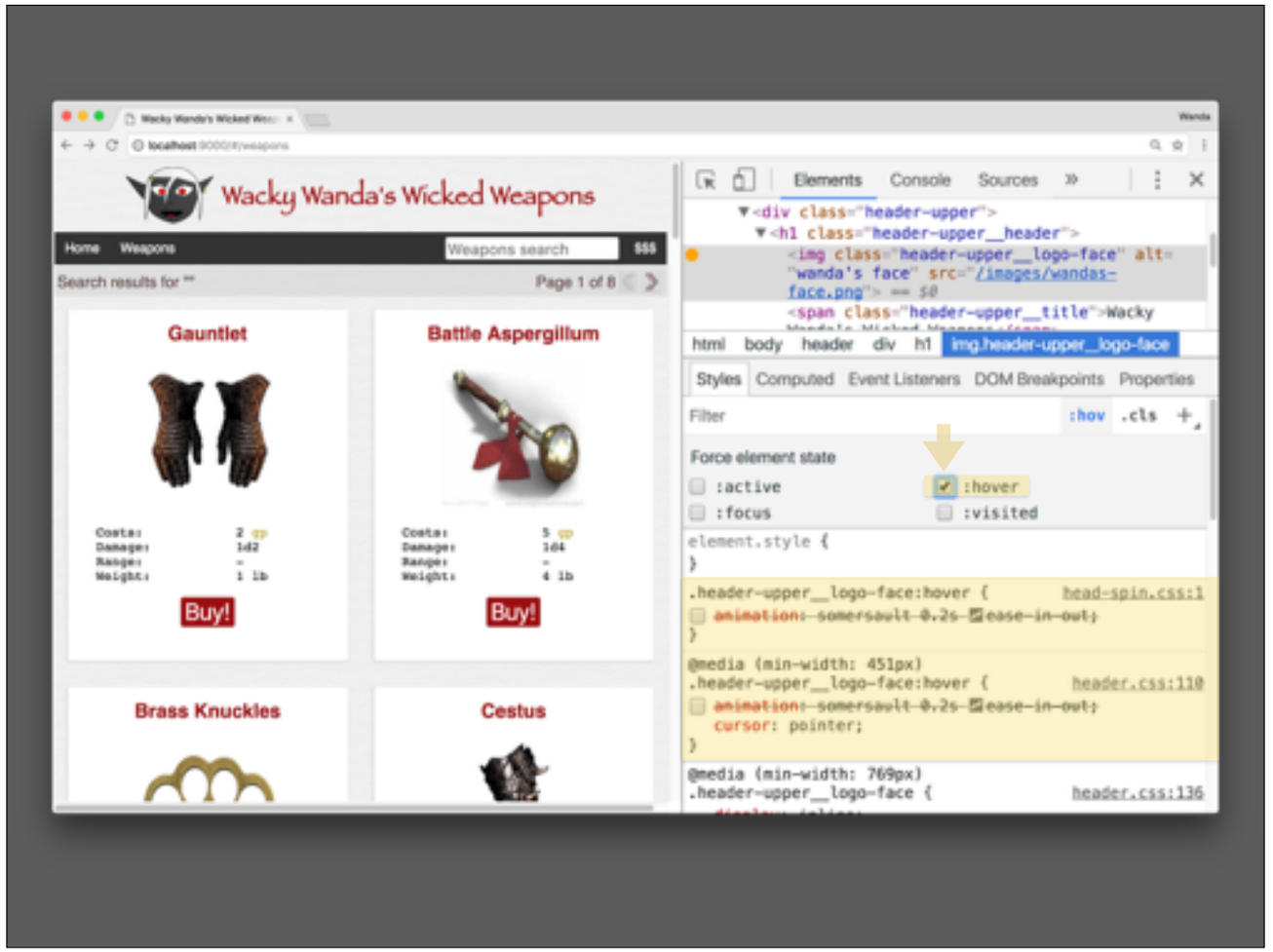
If there is room the Computed Styles will appear in the column right next to the styles. Otherwise it get's it's own tab. The first we can see is a breakdown of the sizing for the omg. If I hover over the margin part we the area it covers on either side of Wanda's Face when it appear above the text. A clear indication then an "auto" value is in play.



The main contents of the Computed section show a break down of each style in play, and what the competing values are.



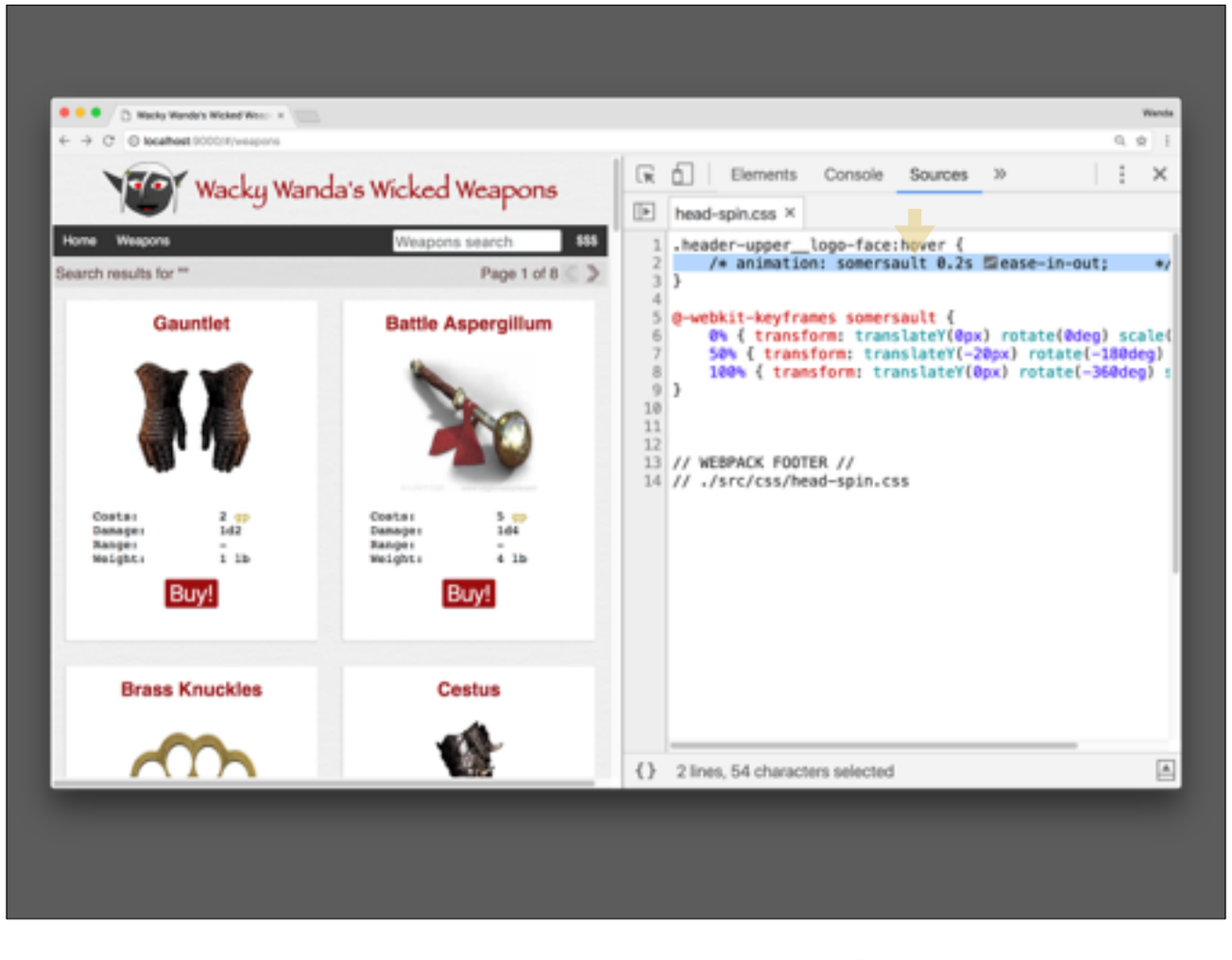
So back to the matter in hand! We want Wanda's head to jump up spin when the mouse hovers over it. The styling we can see is for the static state, not the hover state. We can force the hover state though by clicking on the ":hov" filter and opening element state filters.



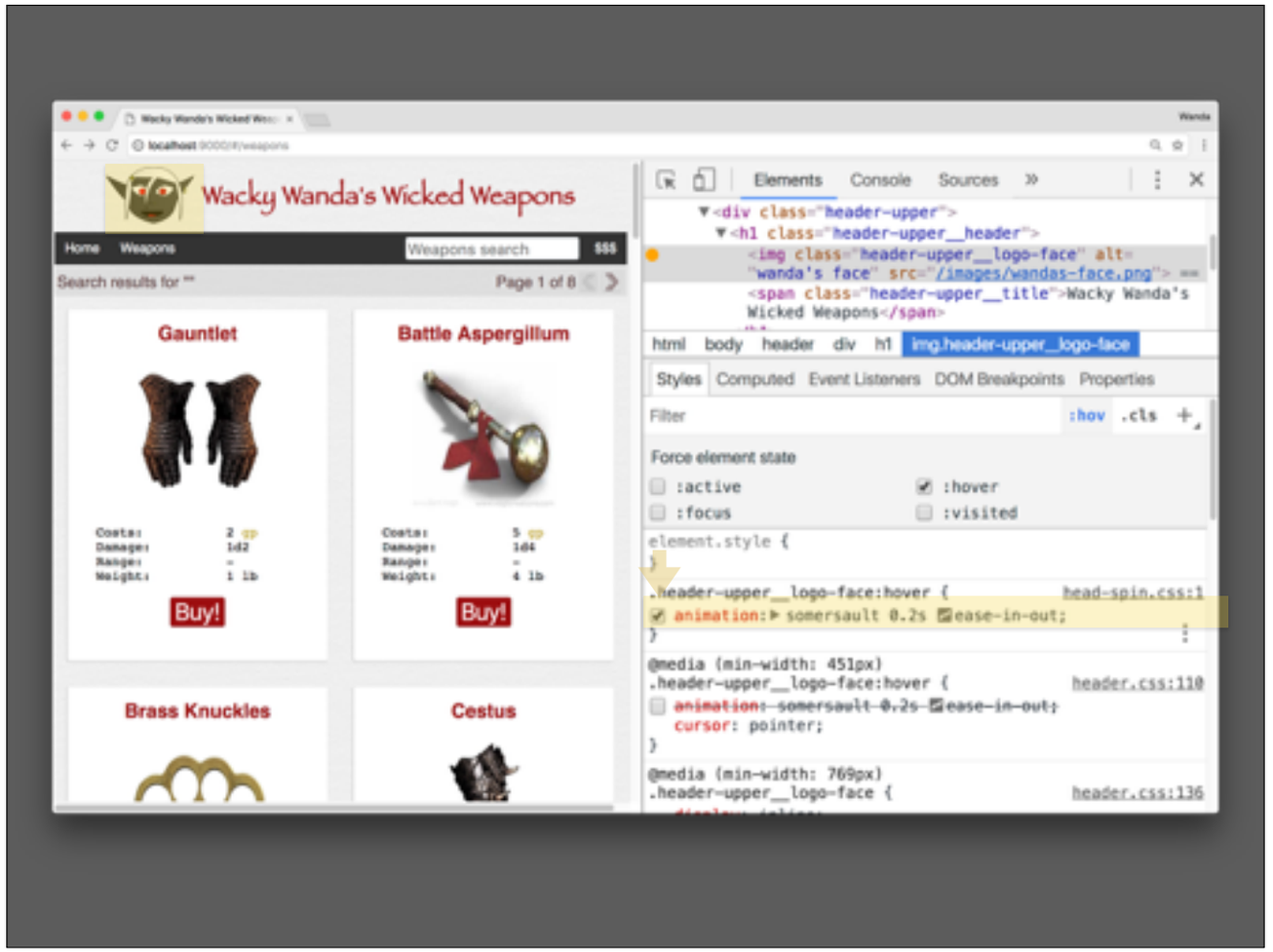
If we click on the hover state checkbox we see an immediate change in the styles.

Something interesting has shown up. What are these crossed out “Somersault” animations that are now showing up?

We can go find out by clicking the “head-spin.css” hyperlink where this entry was defined in css.

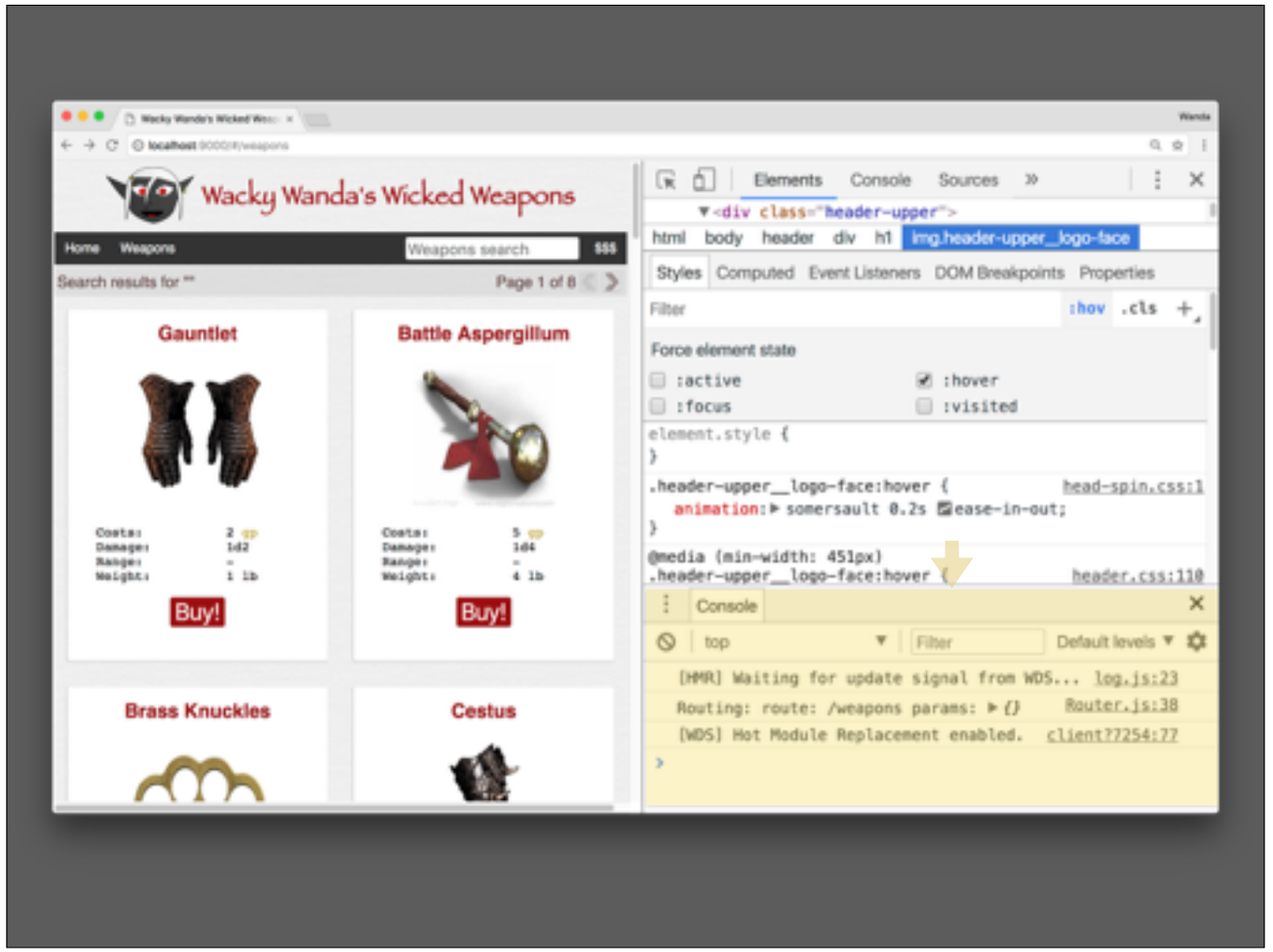


It turns out that Wanda's styling elves had actually made a start on this feature but commented it out. Chrome apparently notices commented out styles and makes them visible, if disabled.

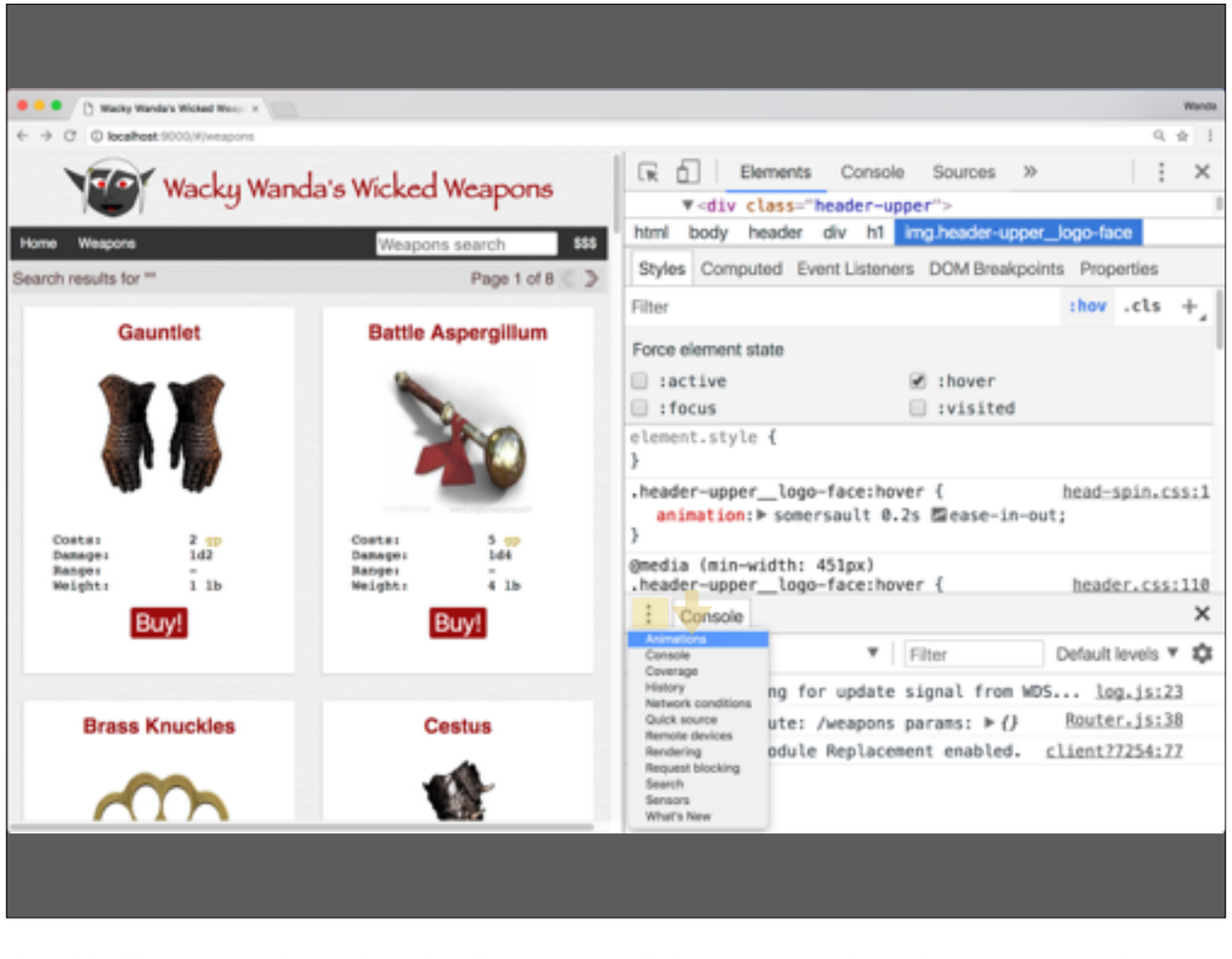


Switching back to the Elements tab, we can try enabling the style using the checkbox and… something happened!

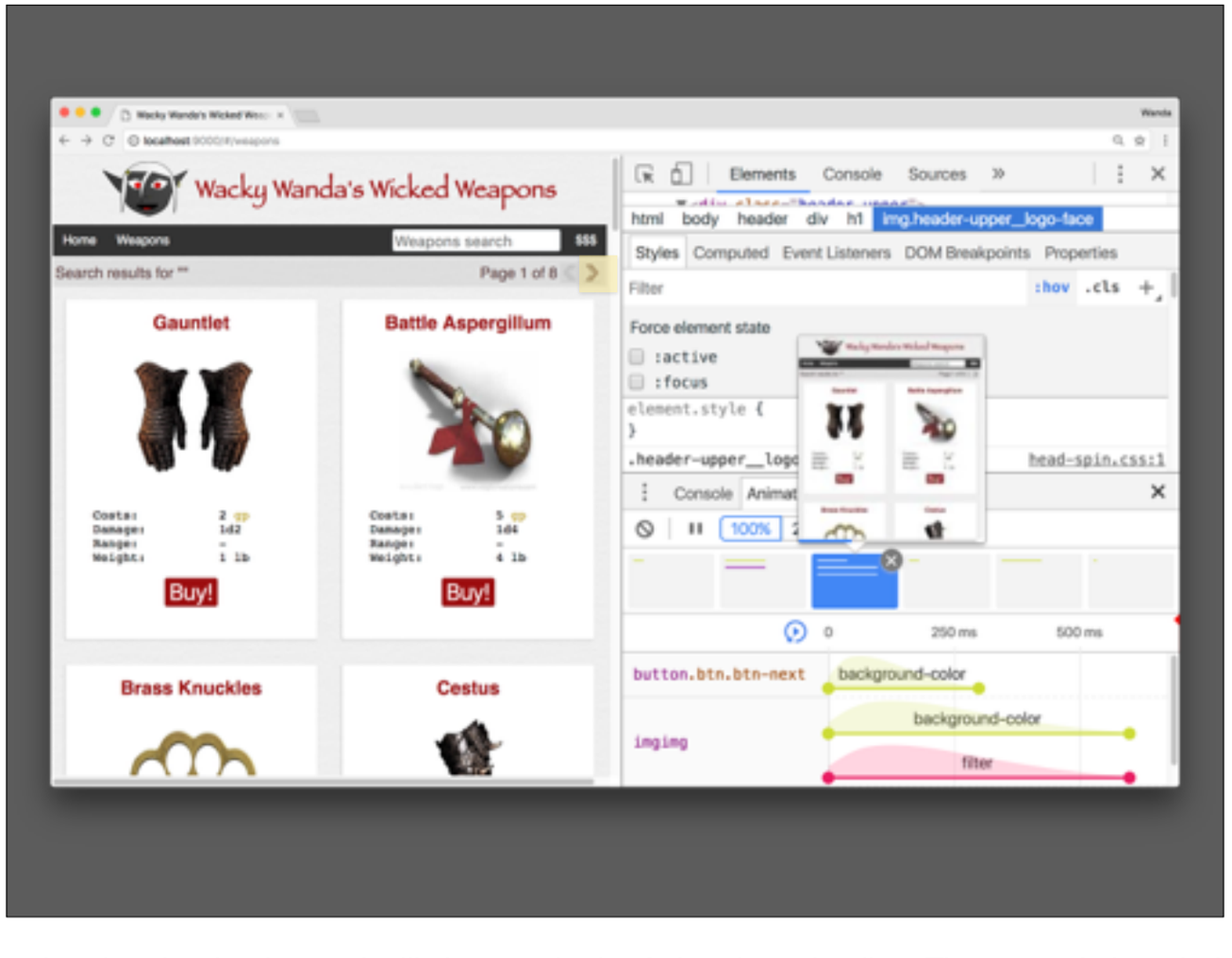
Sorry Speakerdeck viewers, you don't get to see this, but apparently Wanda's head just jumped and spun, really really quickly!



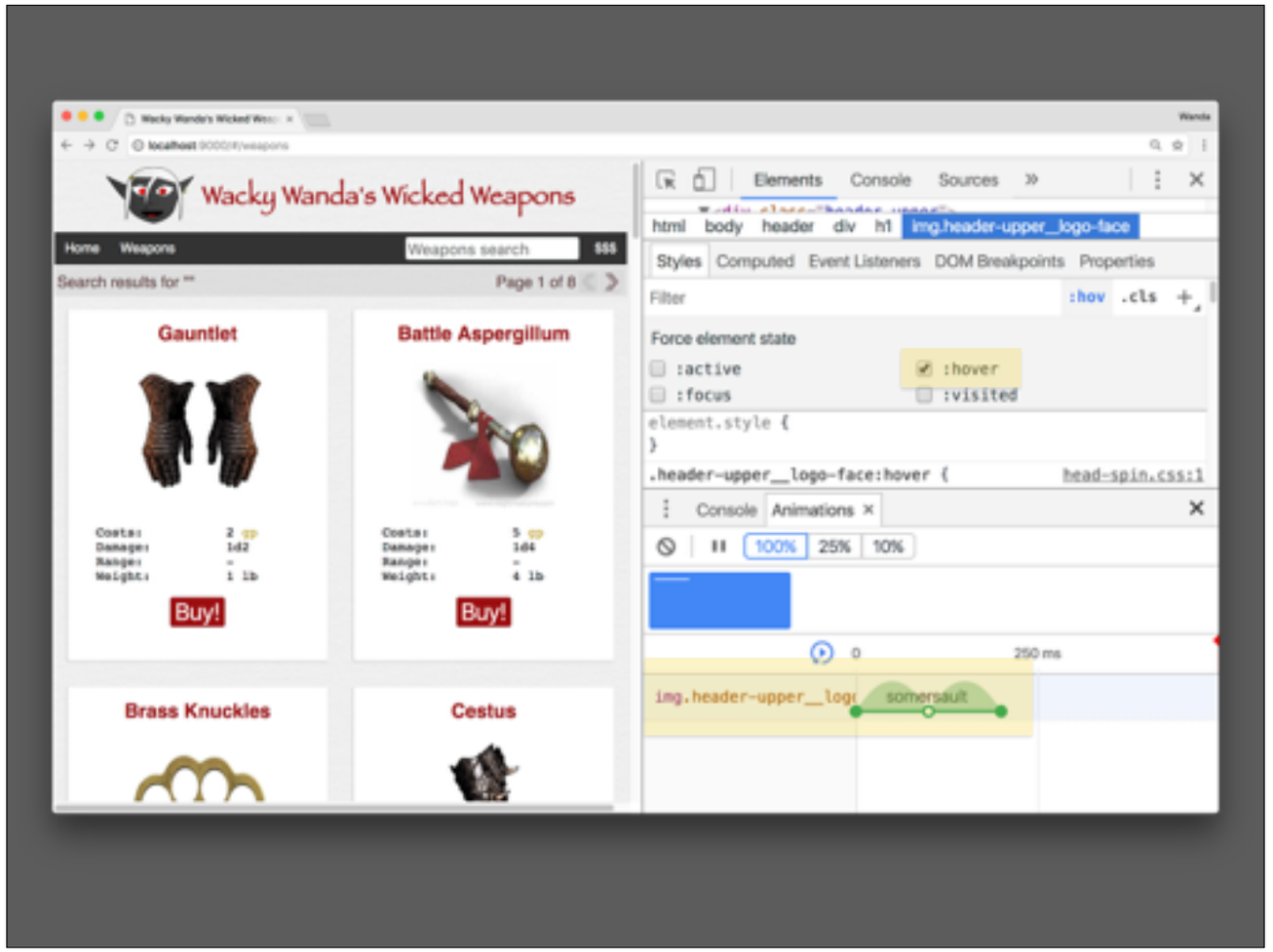
The animation was too fast. Let's see what happens if we slow it down. There's an animation tool available as a Drawer. The drawers pop up at the bottom of Dev Tools if you press escape. You may have already seen that the Console can pop up in any panel. That's the Console Drawer.



The Console is not the only Drawer available. We'll open up the Animation Drawer by clicking on the triple dot men and selecting Animations

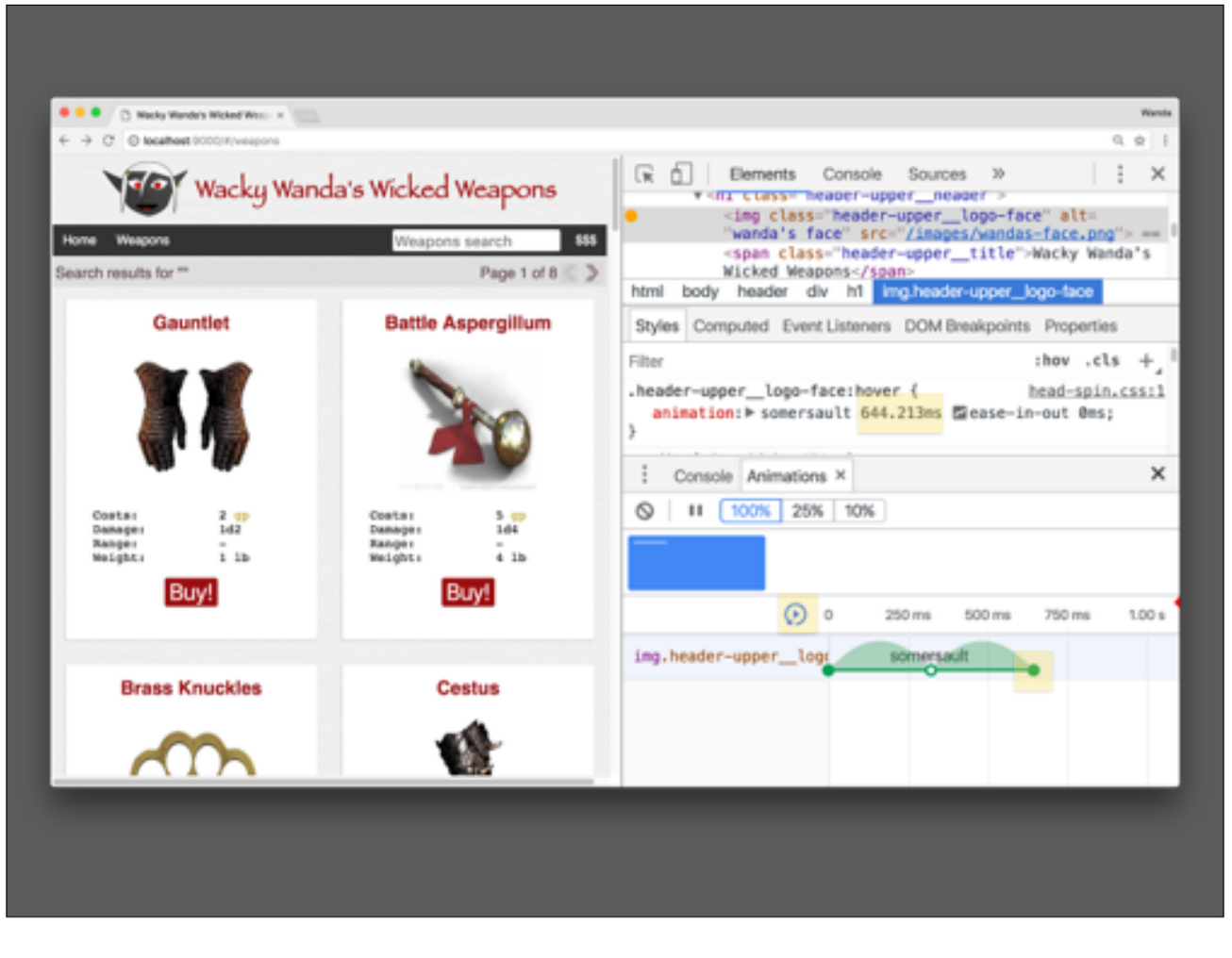


When the Animation Drawer is open any animation that is triggered will show up in the Animations timeline. The example here is for the Next button animation. It can be replayed at will. We can also fine tune the phases of the animation.



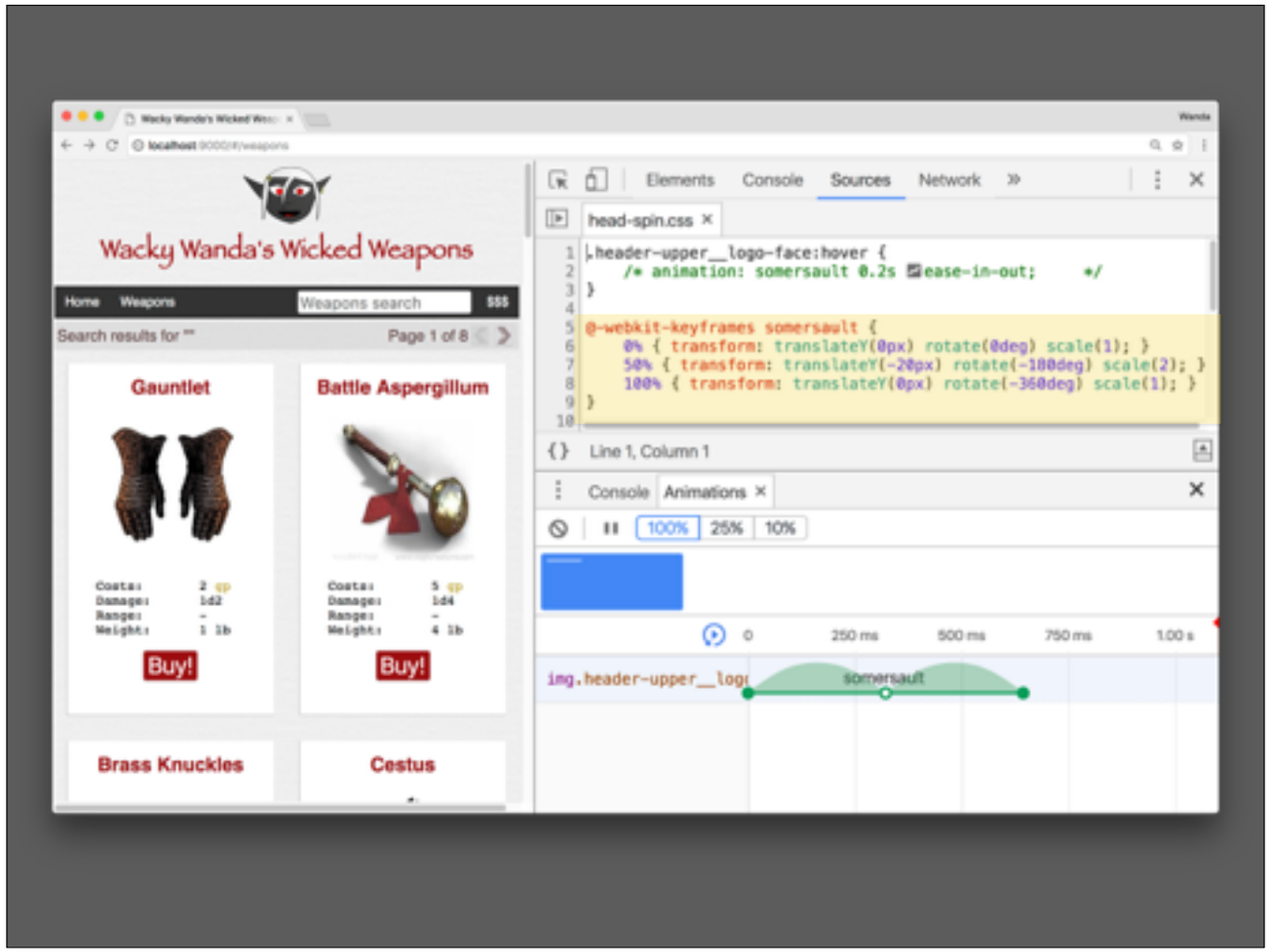
The animation we're interested in though is for Wanda's head when we hover. I'll go ahead and clear away the recorded animations and use the hover checkbox to replay the desired animation.

It looks like the somersault animation run in less than 250ms which is way too fast.



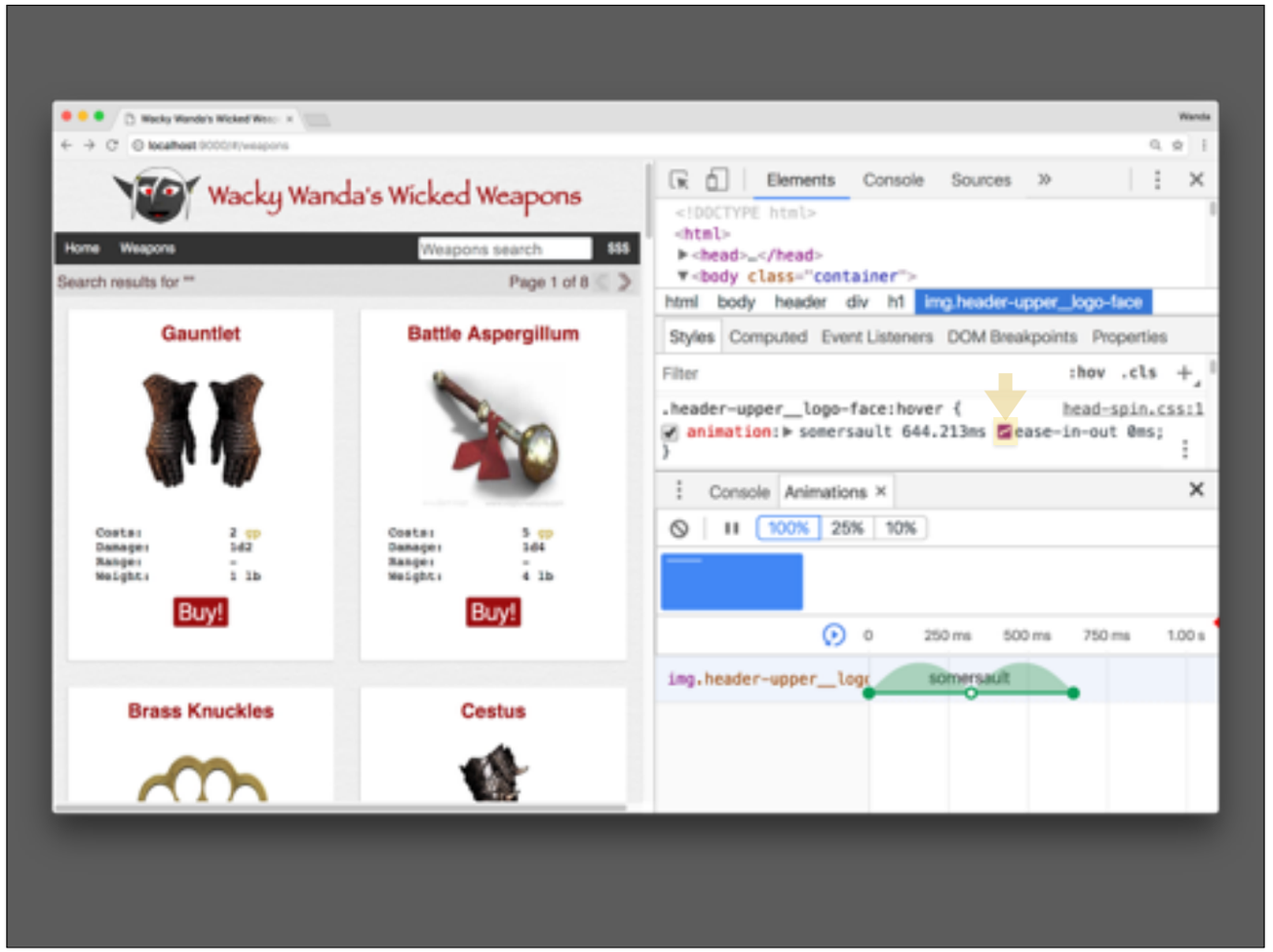
A little experimentation shows that 650ms seems to work much better. I hit the play button overtime I want to see a replay. Also the styling updates itself as I make these changes. Although it'll only last until the page is refreshed.

There is still an issue though. Wanda's somersault seems to slow down and pause half way through. The camel shaped graph implies that the animation is a keyframe animation with 2 distinct phases.

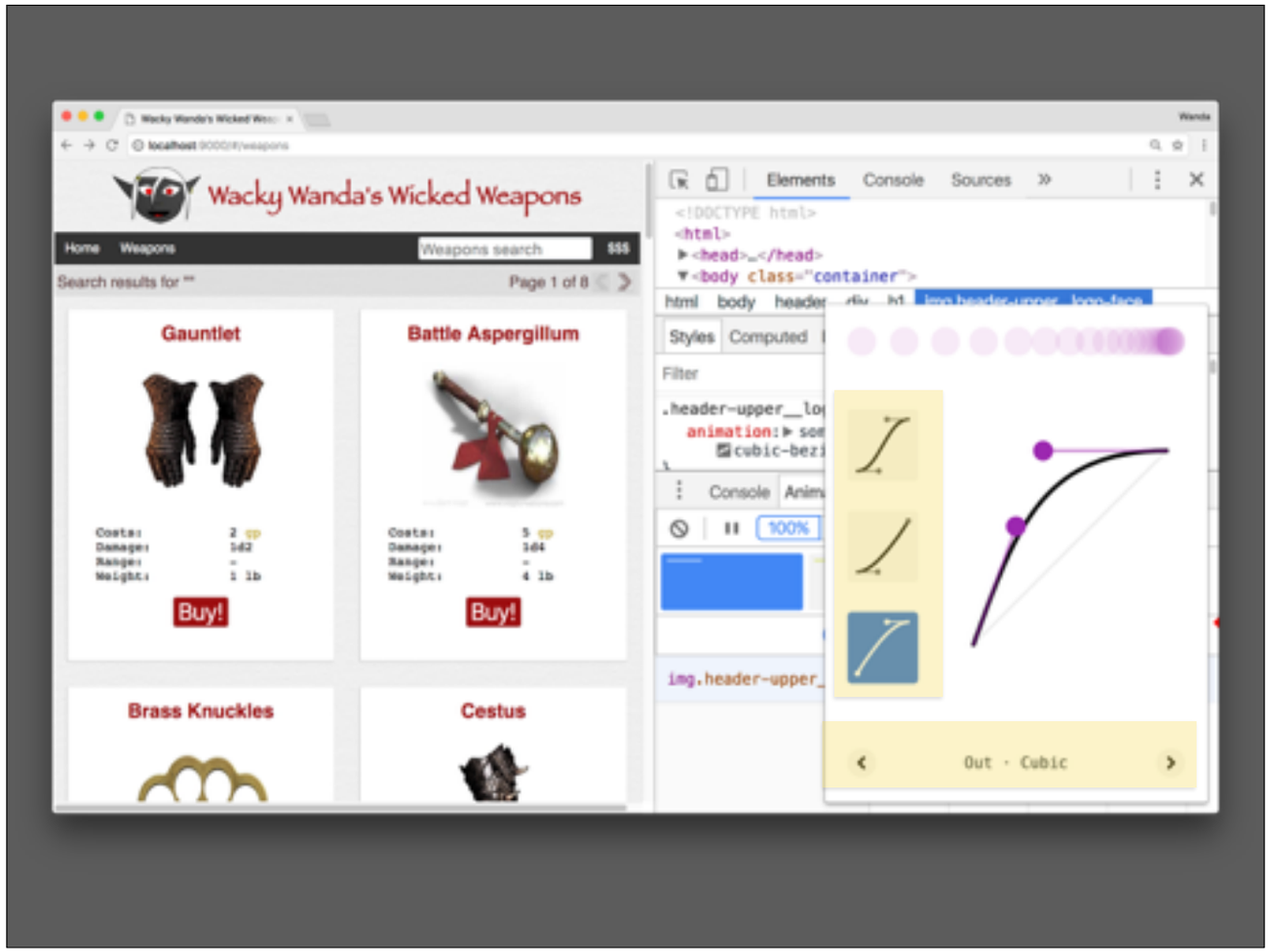


Checking the css file again confirms this. It apparently is a keyframe animation with 3 points:

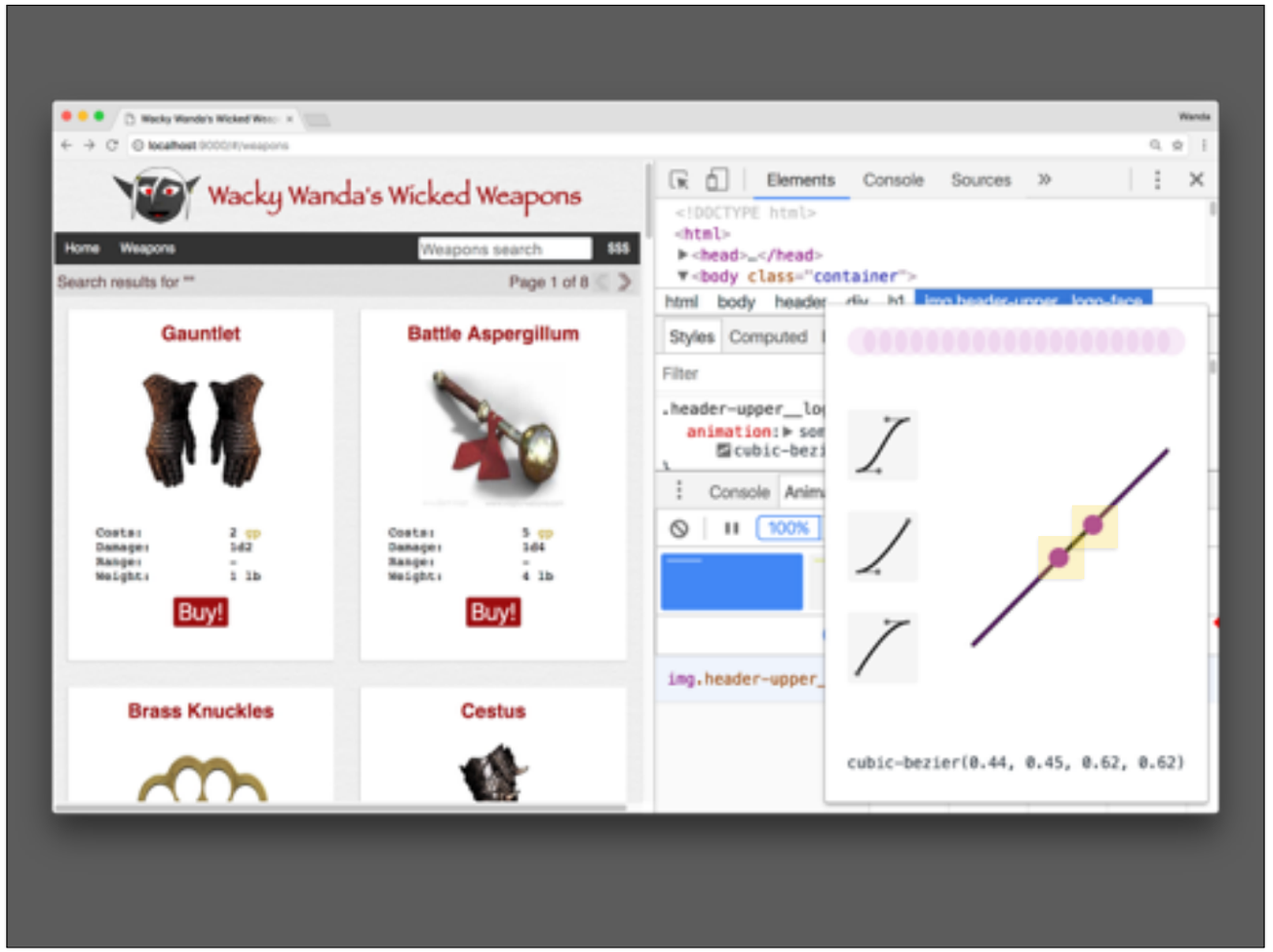
At 0%: No movement yet
At 50%: Wanda’s face is transformed so her face is in the air and rotated so that it is upside down
At 100%: A reverse transformation occurs. Wanda’s head continues the rotation back to origin.



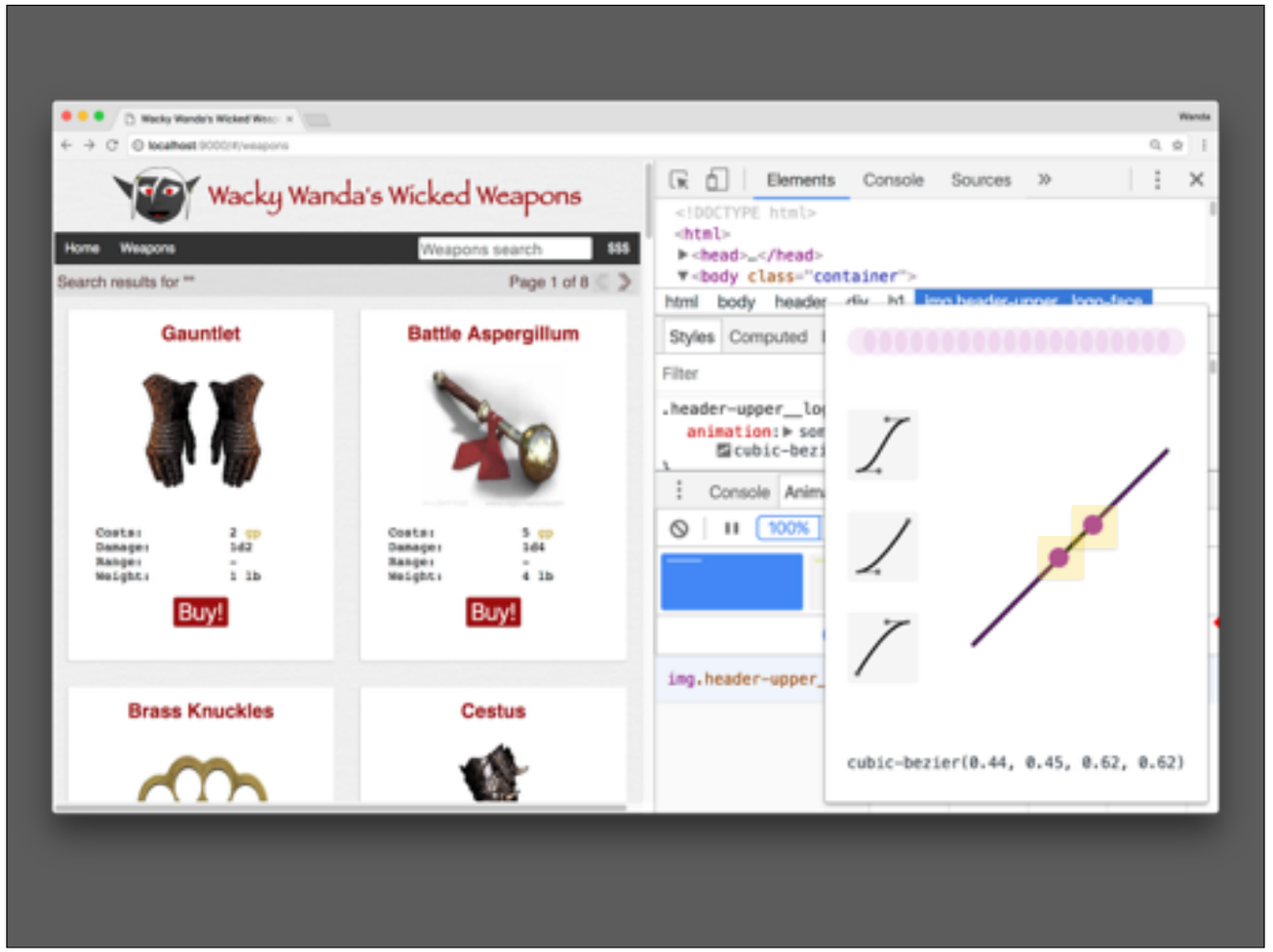
Returning to the Element's Panel it looks like we need to change the animation graph to something that looks jarring when repeated. Many of the styles have little editors built in. In the case of animations there is a Cubic Bezier Editor. Just click on the Sine Wave icon next to "ease-in-out".



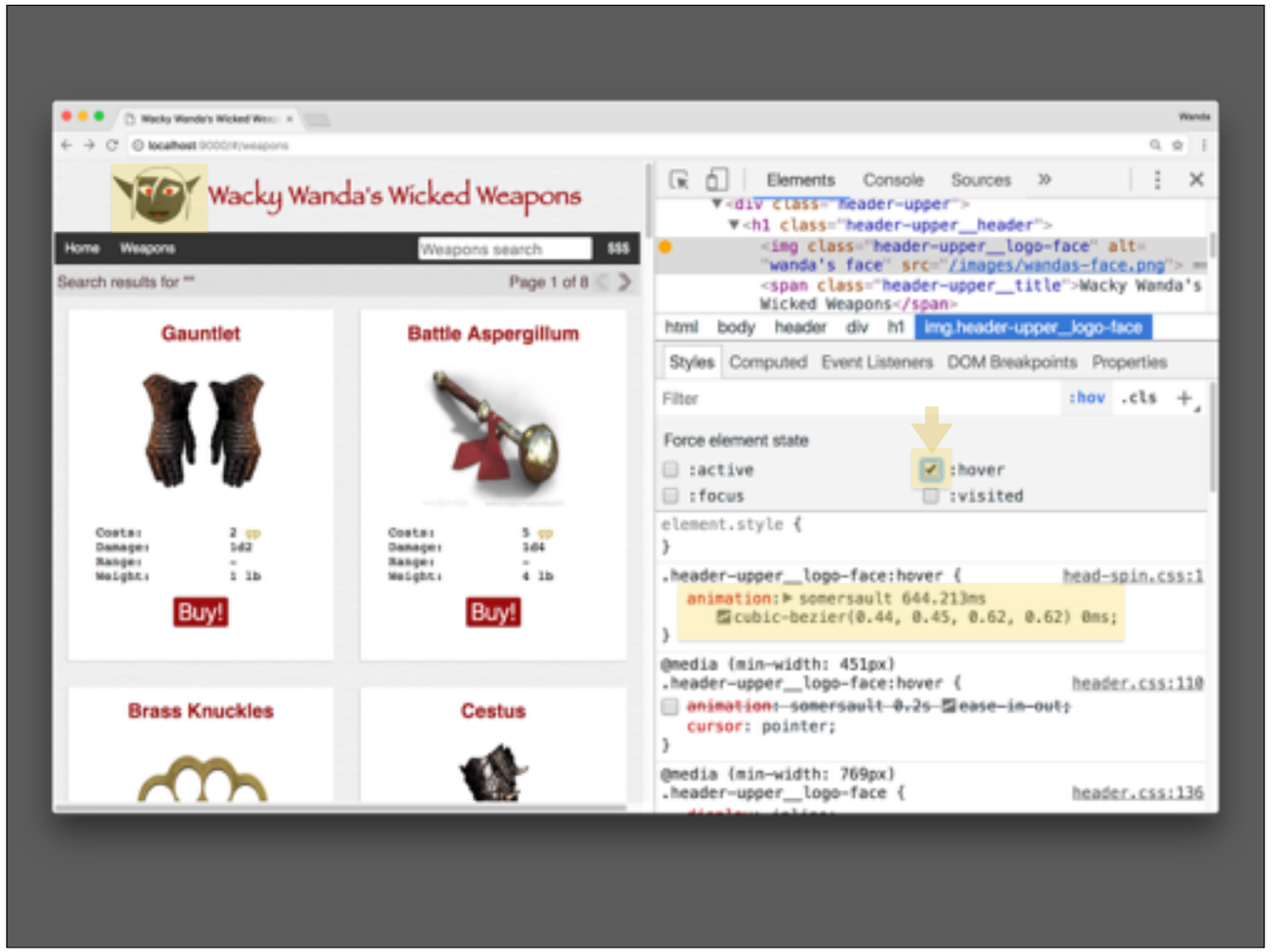
From the Cubic Bezier Editor there are a number of animations we can cycle through. None of them look like what we need though.



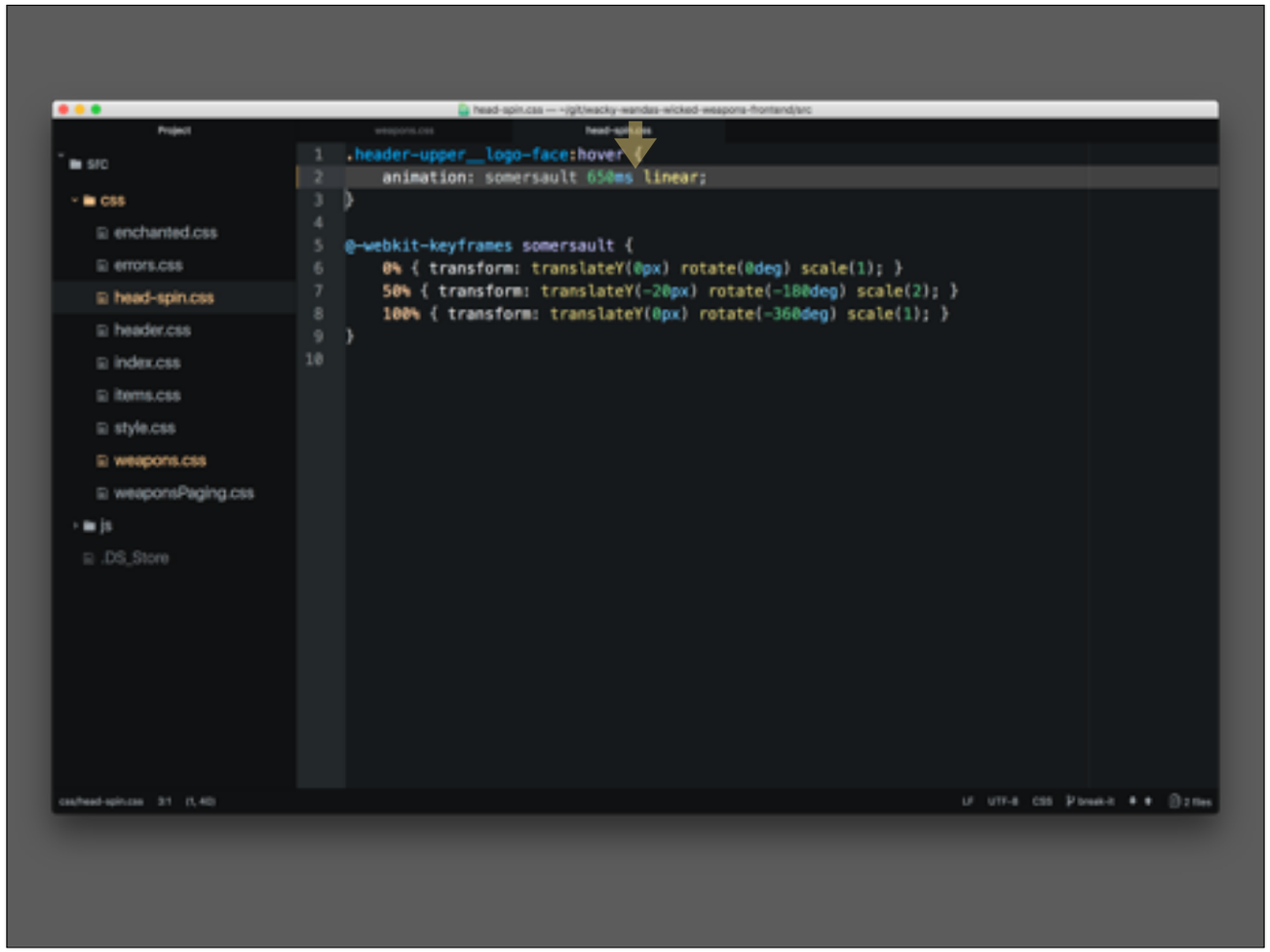
I think what something more linear, so I just go ahead and edit the graph to form a line.



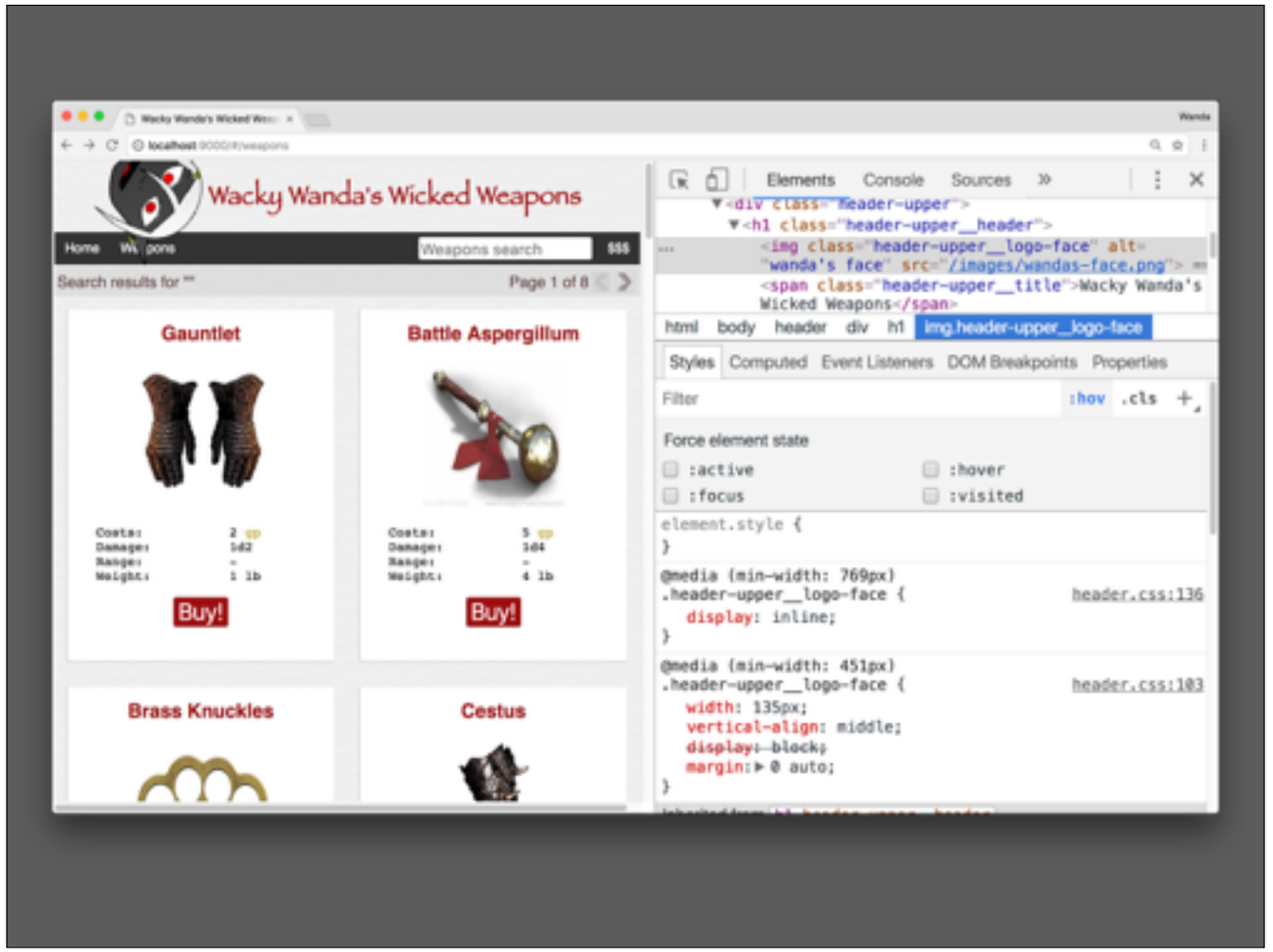
I think what something more linear, so I just go ahead and edit the graph to form a line.



The animation doesn’t exist in the recorded Animation shown in the Animation Drawer so I just close it and test using the hover checkbox. Looks good!



Now we just need to apply the fix which I do from Atom. The css is in head-spin.css. I uncomment the animation definition and based on our experiments in Chrome I change the duration to **650ms**, and the animation type to linear. I could have used the **cubic-bezier(0.44, 0.45, 0.62, 0.62)** definition, but I happen to know that's the same as a **linear** animation, so might as well just use that.



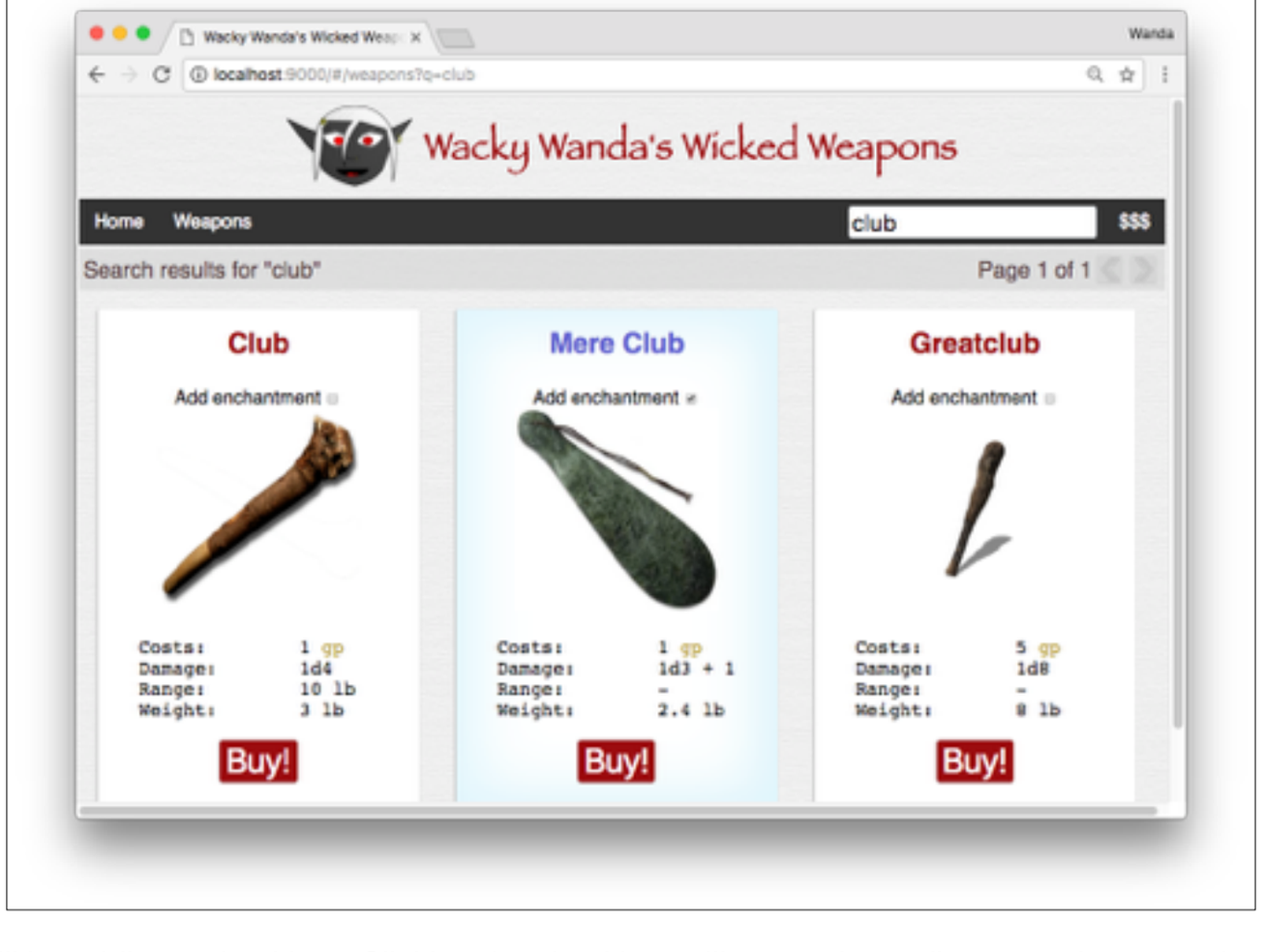
And if we go back to Chrome, refresh the page and test by hovering the mouse over Wanda's face? Looks like it works!

I even managed to capture it mid-jump just for you folks following from Speakerdeck!

Console Panel/Drawer

Enchanted Weapons Part 1

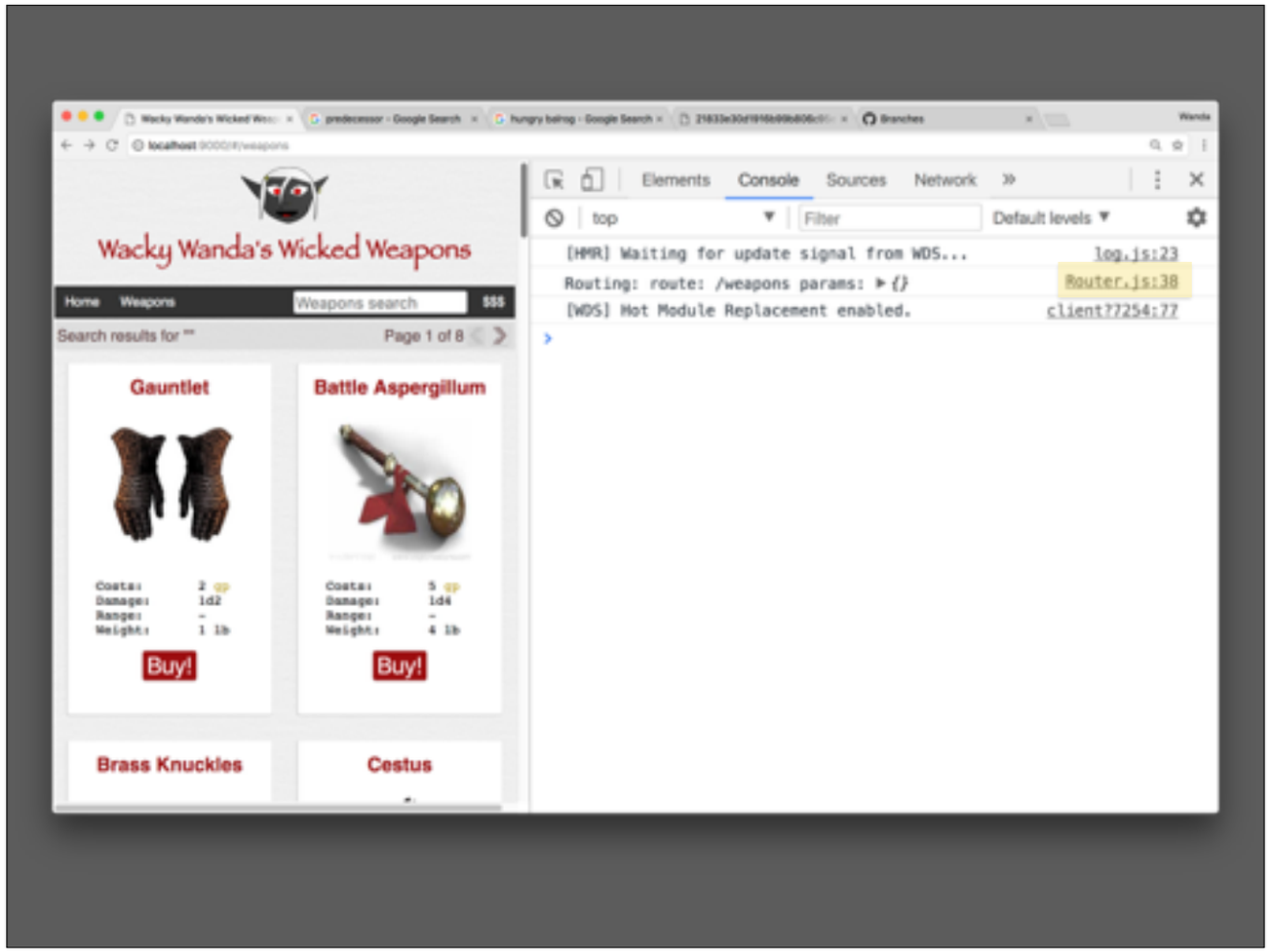
Time to move onto the next task. Wacky Wanda’s Wicked Weapons is expanding to support enhanced weaponry



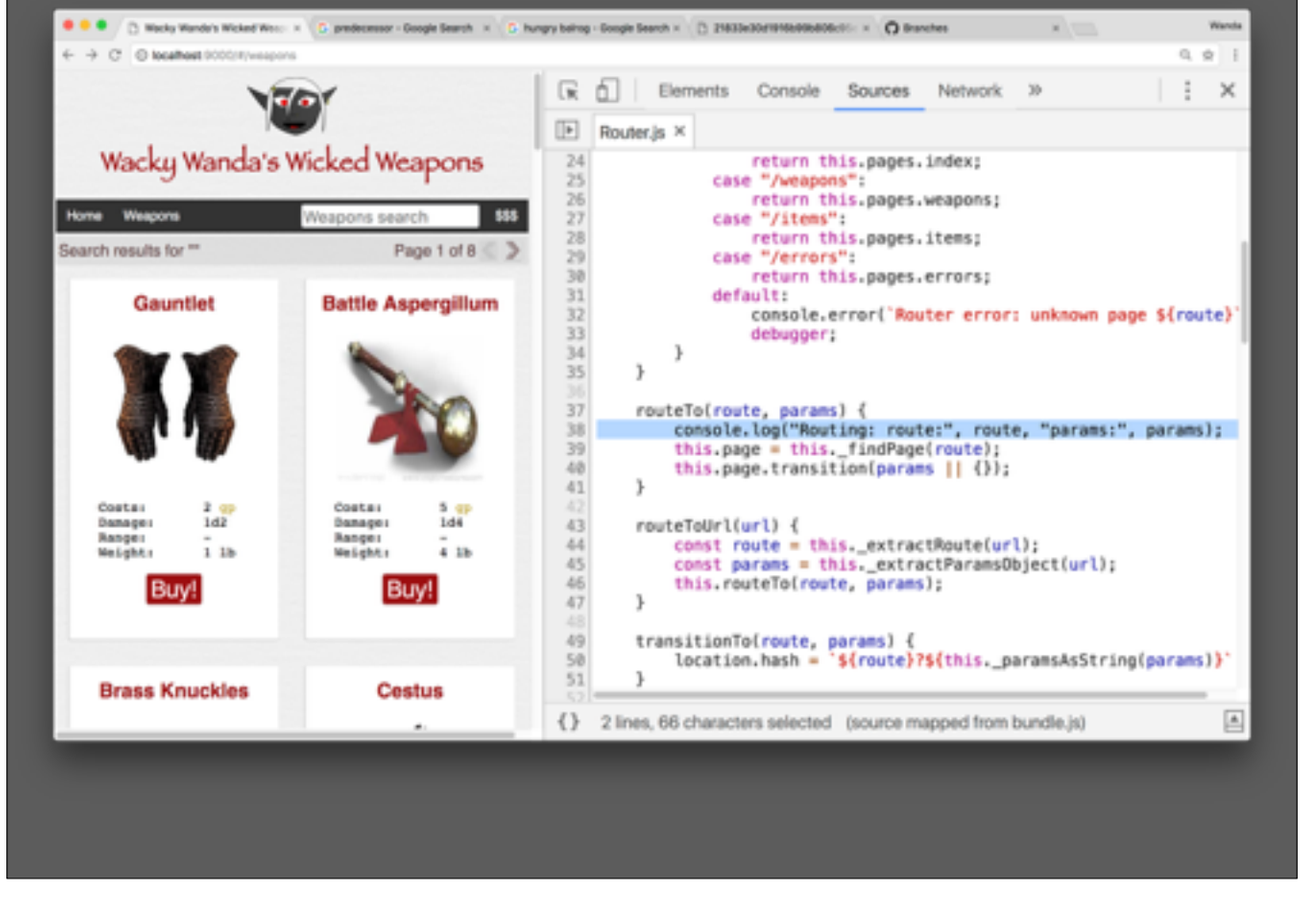
This is what we're want to build out. We'll be adding a checkbox. Clicking the checkbox will cause the attributes to change and magical glow to appear.

Work has actually already begun on this but tragically my predecessor was tragically eaten by a balrog from the accounts department.

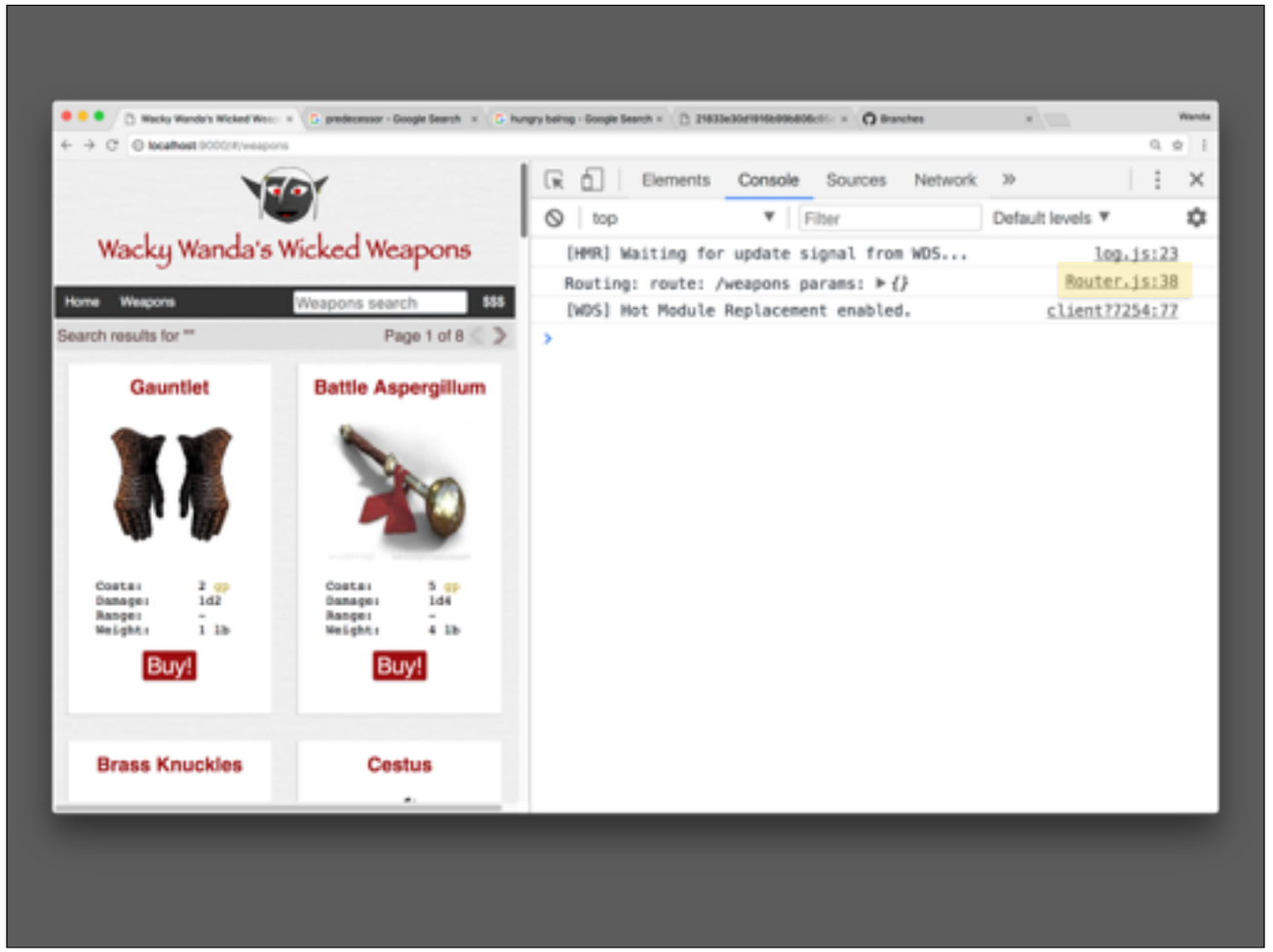
It's my tasks find out far Jasper got with implementing the Model before his untimely workplace accident. For this first phase we'll be working from Console Panel and sometimes the Console Drawer to enchant Mjolnir as proof of concept.



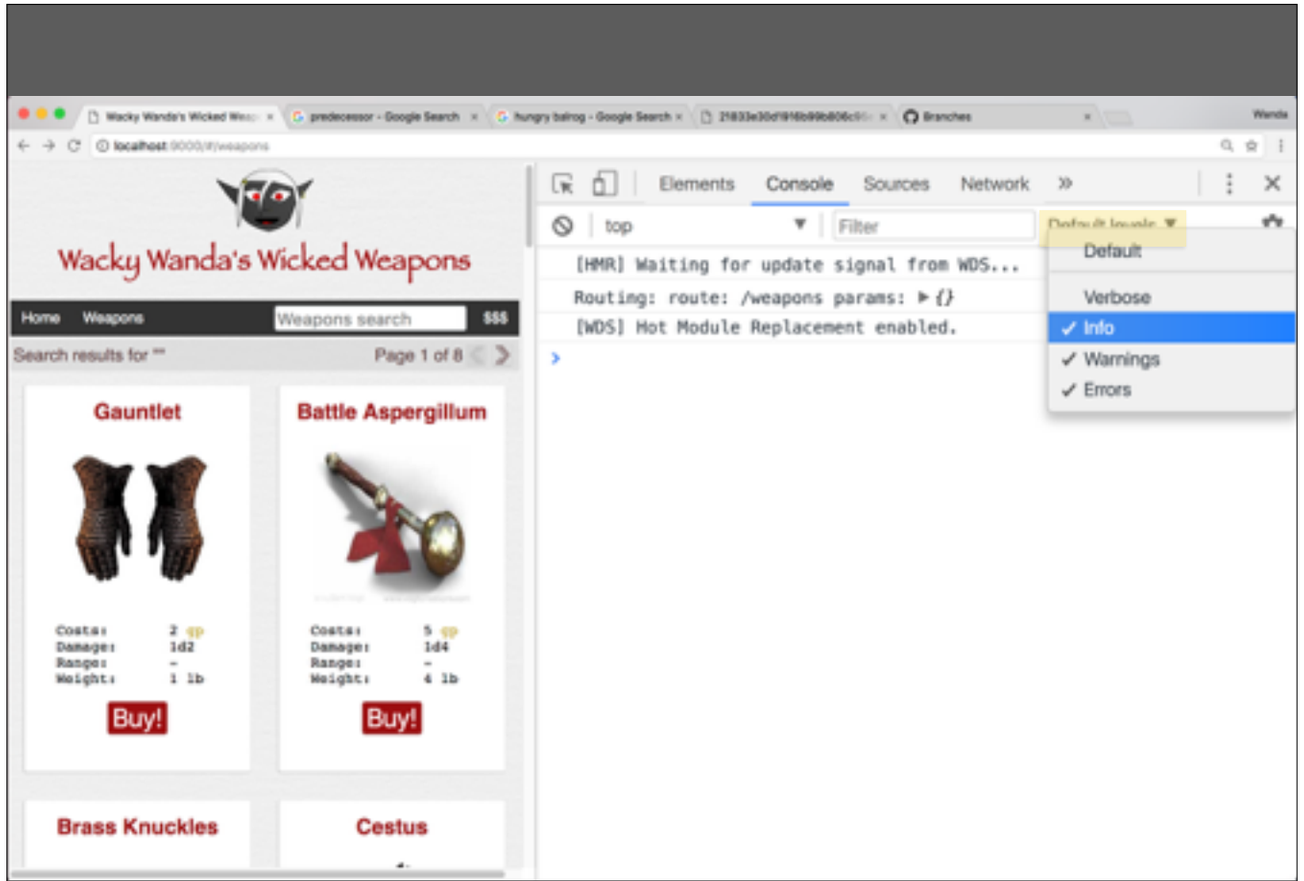
To the Console Panel! Before we get started there seems to be tracing information showing up here from the webpack dev server and also from our router. Let's click the router link to see where that's coming from.



Yep, it’s coming from console.log. This method of logging works cross browser. You can also use console.warn and console.error which are displayed a little differently, depending on the severity.

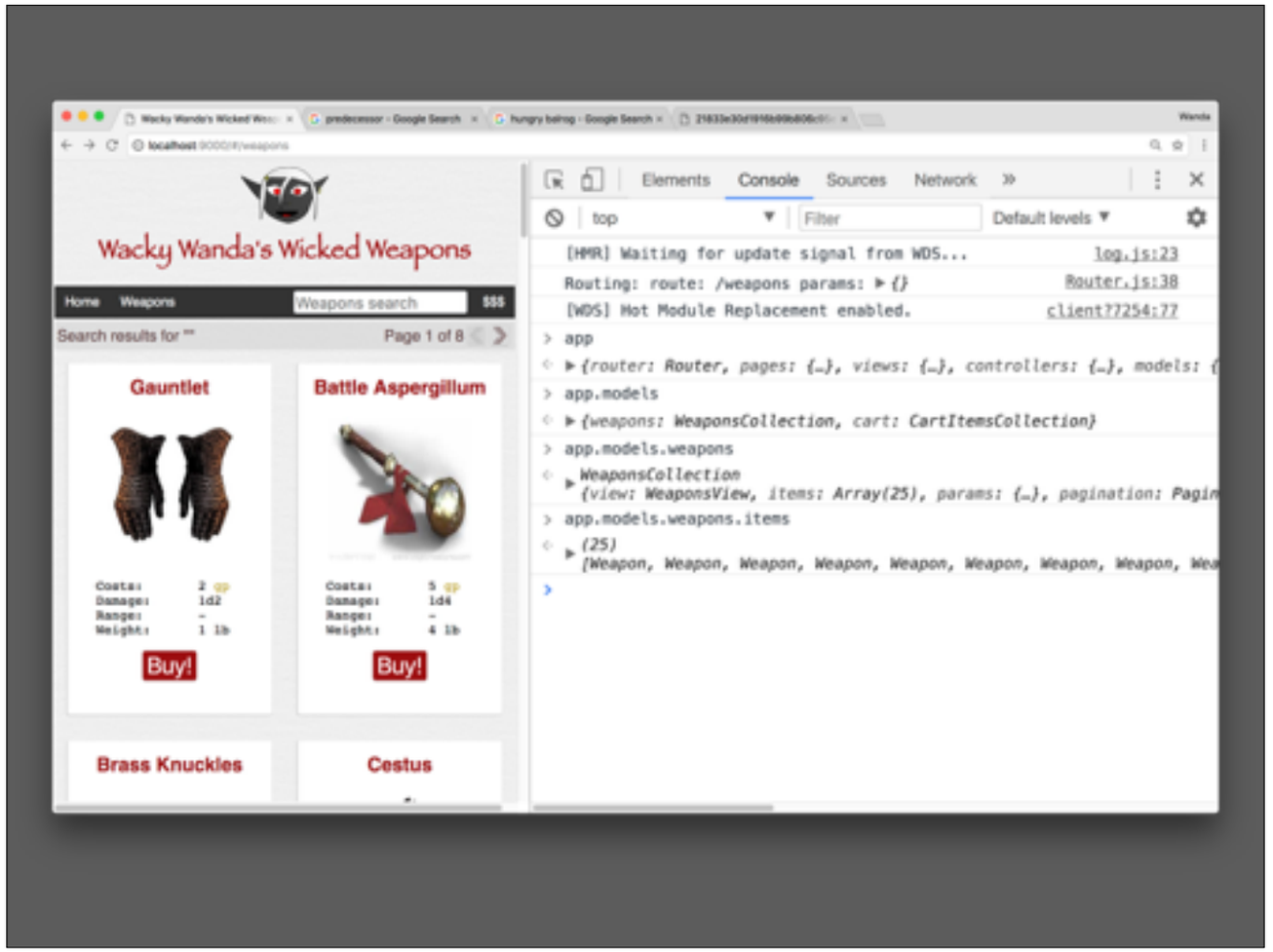


To the Console Panel! Before we get started there seems to be tracing information showing up here from the webpack dev server and also from our router. Let's click the router link to see where that's coming from.



If the trace information ever gets to be a bit much we can dial it down a bit by making use of the “default levels” filtering. Unchecking the “info” will get rid of these messages.

But we’re going to be needing the feedback today so we’ll leave it on for now.

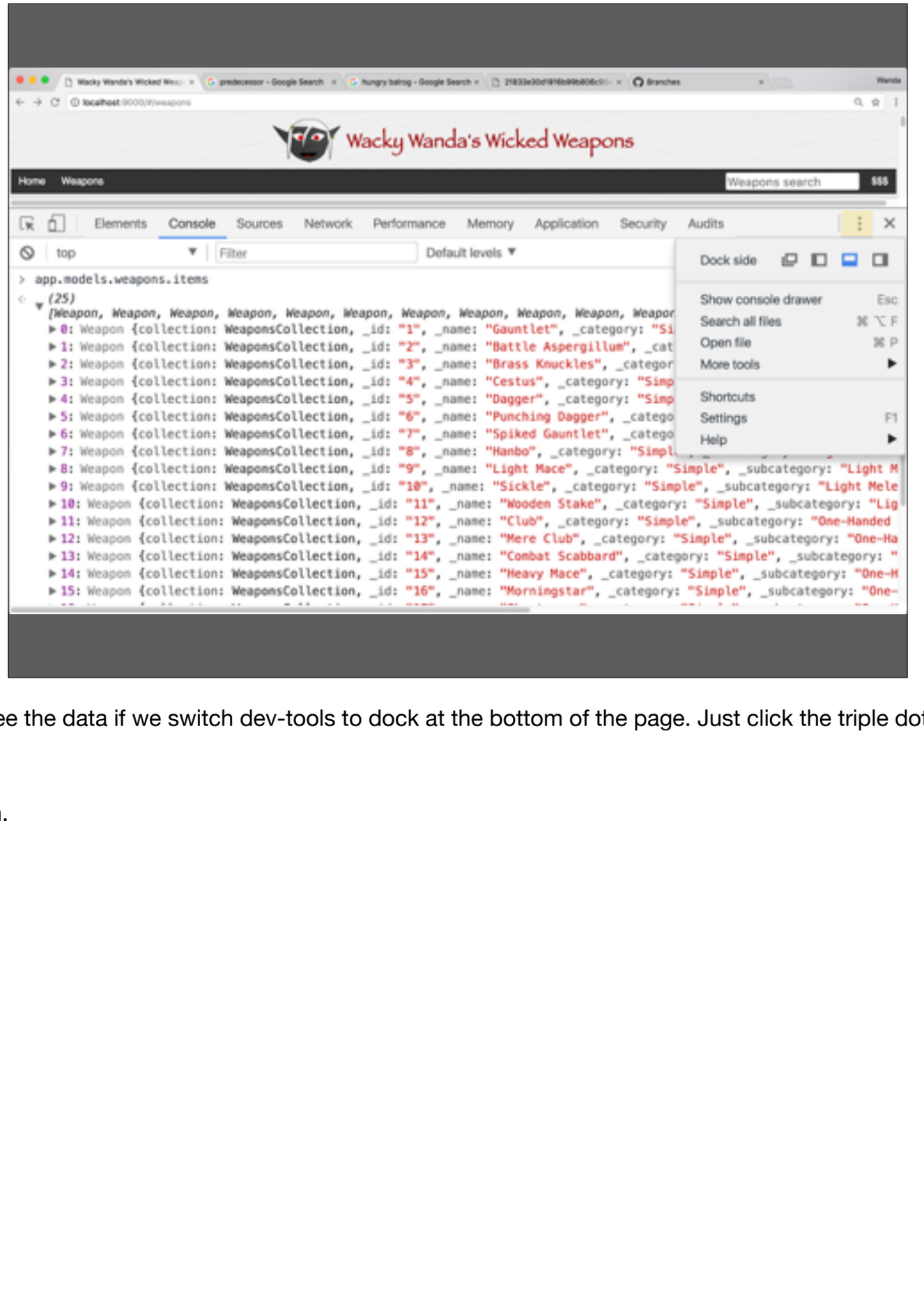


One upside of this application's MVC model, everything is attached to the **window.app** object. Let's start by querying the **app** object.
-> app
Routers, pages, views, controllers…. models, we want models.

->app.models
Weapons or CartItems? Weapons I think

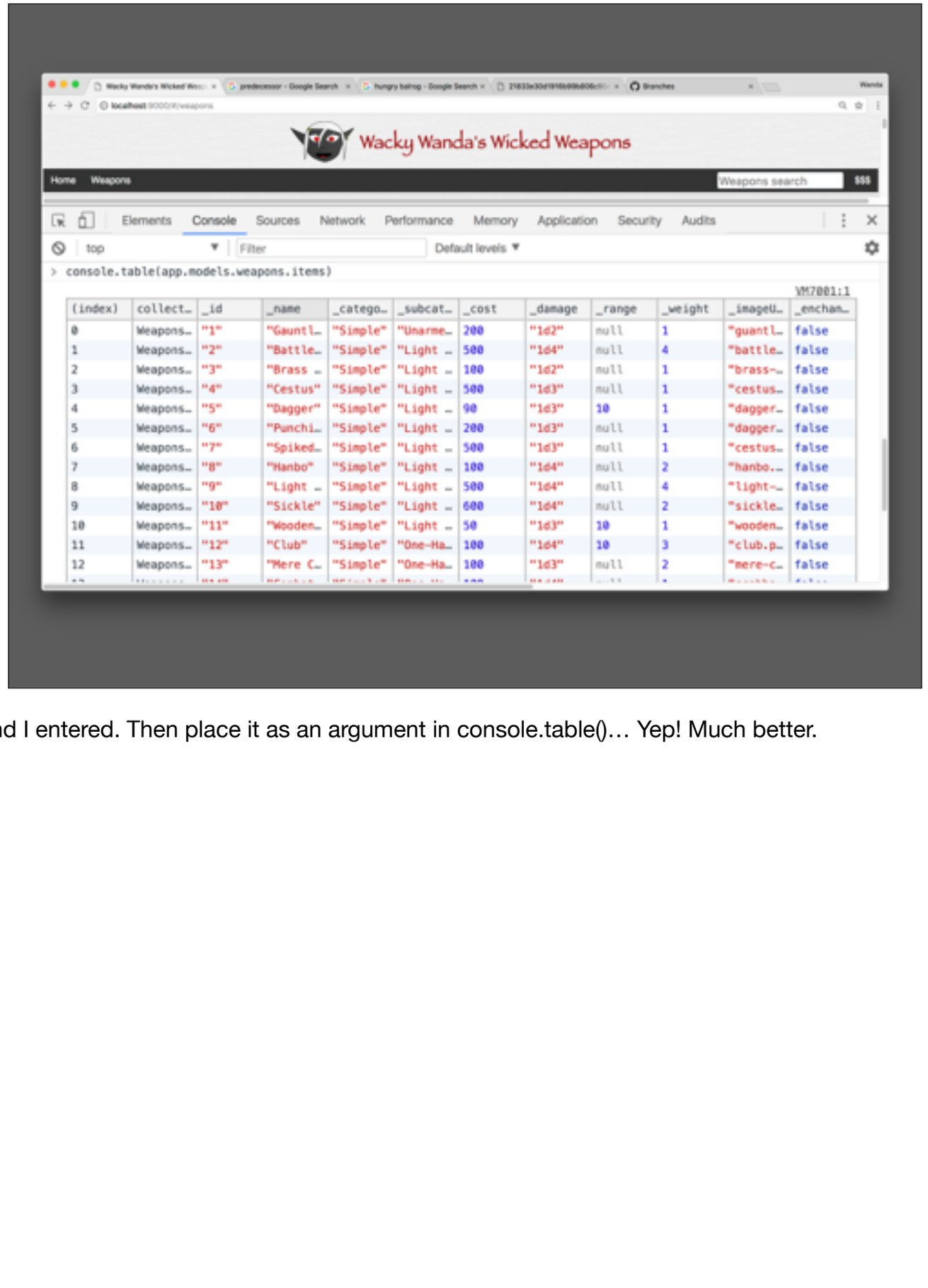
->app.models.weapons
Ok, this looks like the weapons manager. It has 25 items. Problem the individual weapons…

->app.models.weapons.items
Yep, there they are

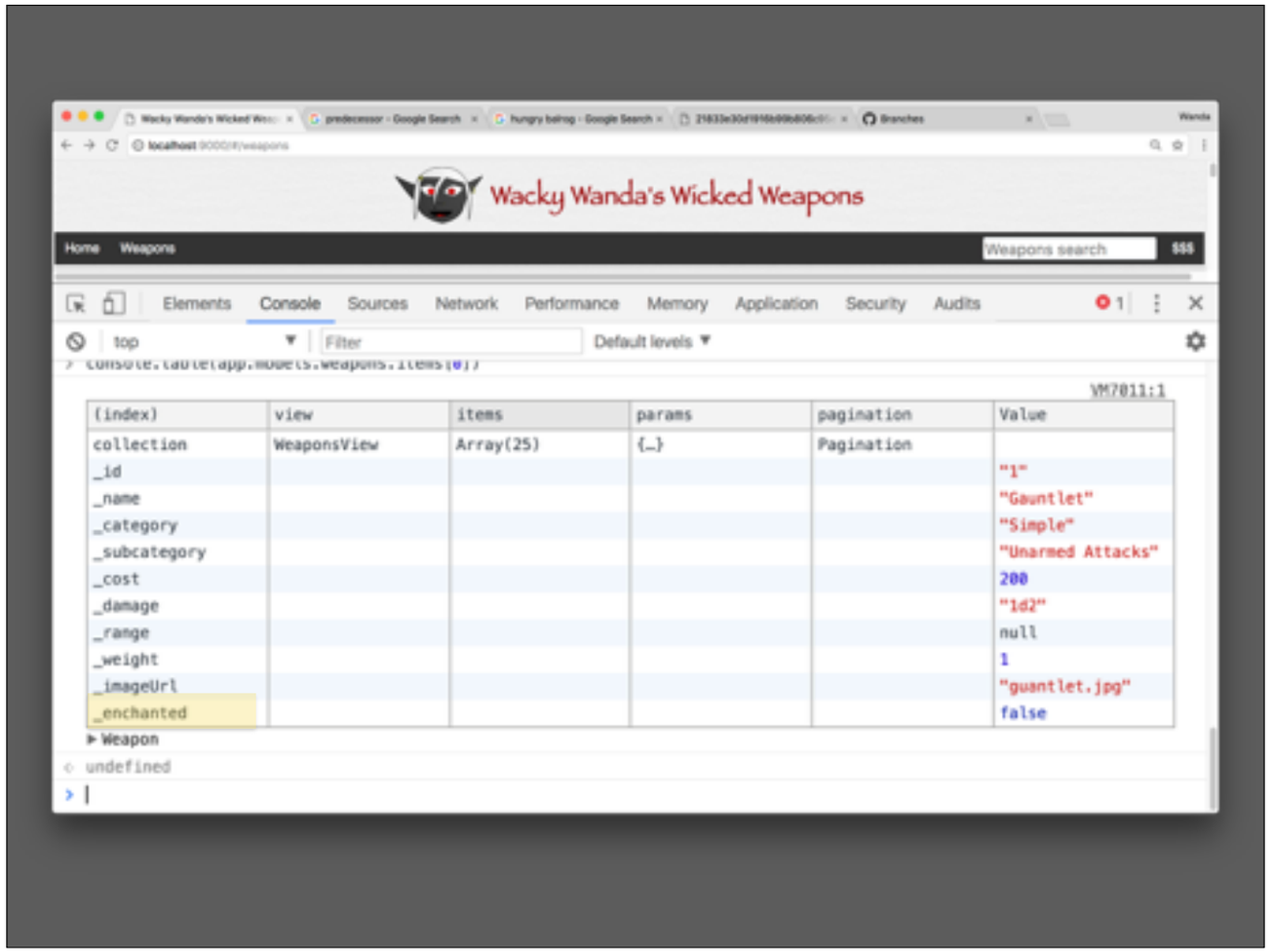


Yep there they are. Might be easier to see the data if we switch dev-tools to dock at the bottom of the page. Just click the triple dot menu and select the 3rd Dock Side icon.

It’s still a little hard to read those though.



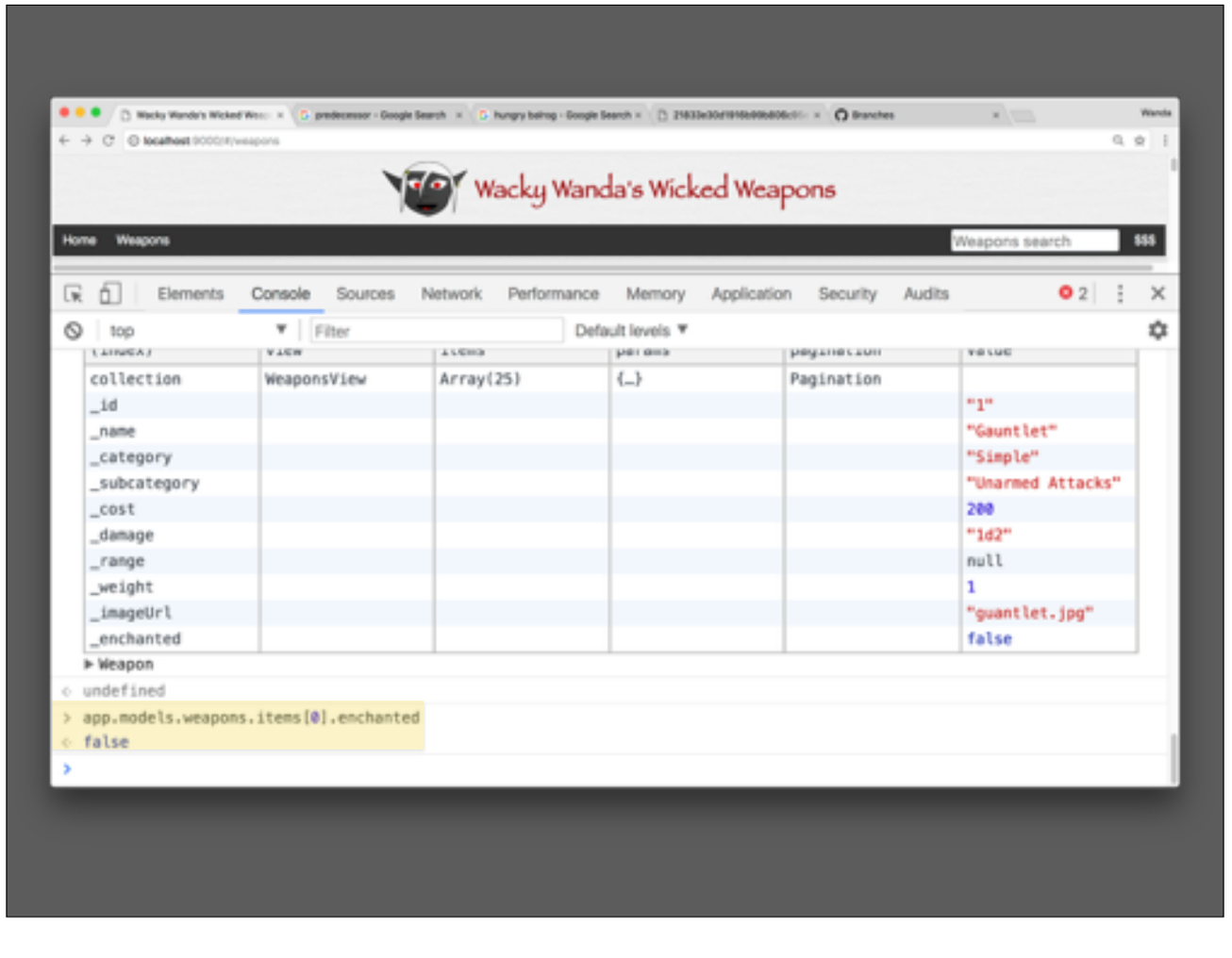
If I push up to bring up the last command I entered. Then place it as an argument in console.table()… Yep! Much better.



Let’s take pull down the first weapon specifically and keep that in a table too:

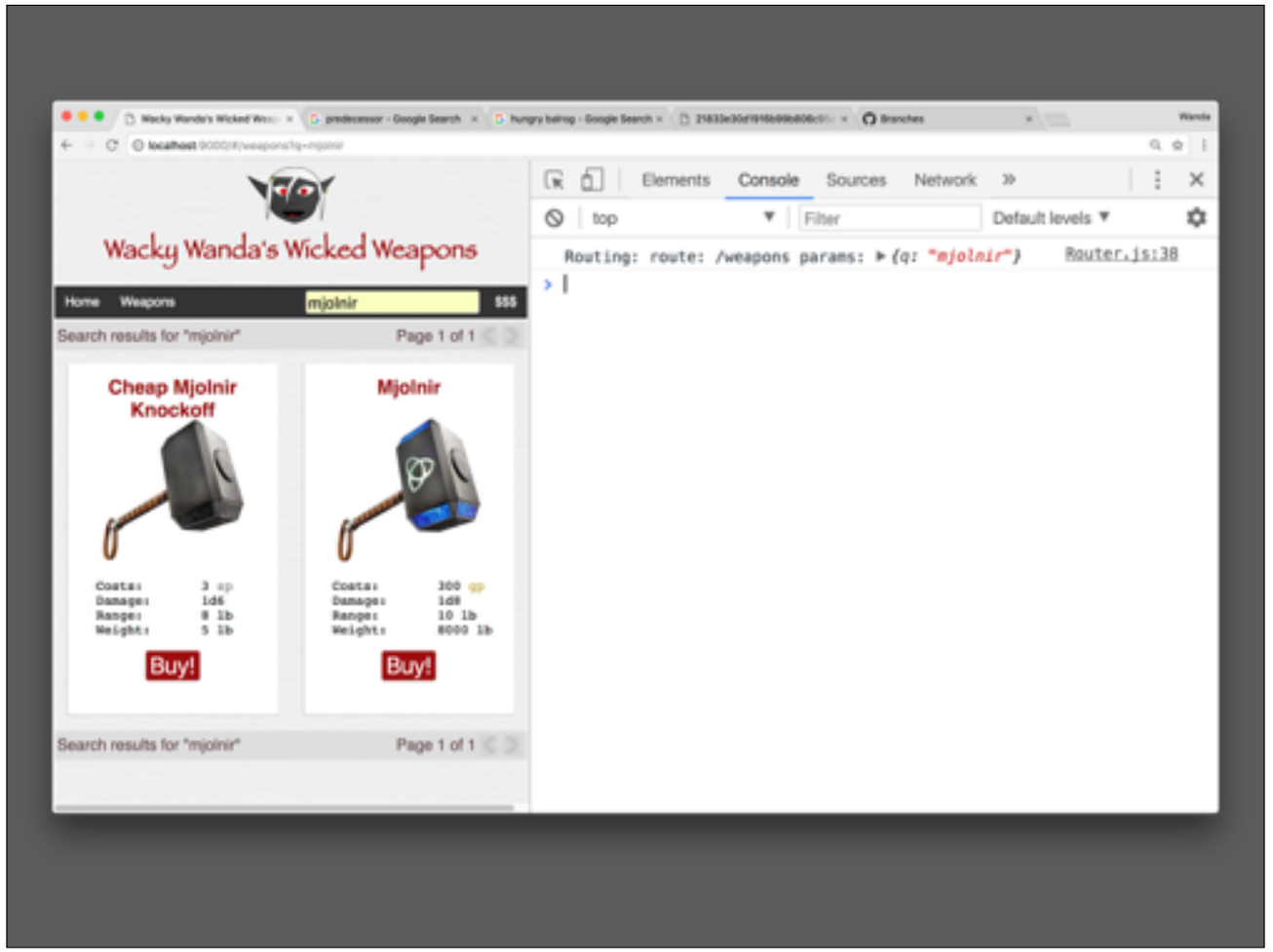
=> console.table(app.models.weapons.items[0])

There we go! Looks like a weapon has an id, name, category, a subcategory, cost, damage, range, weight an image url and… an **enchanted** attribute!

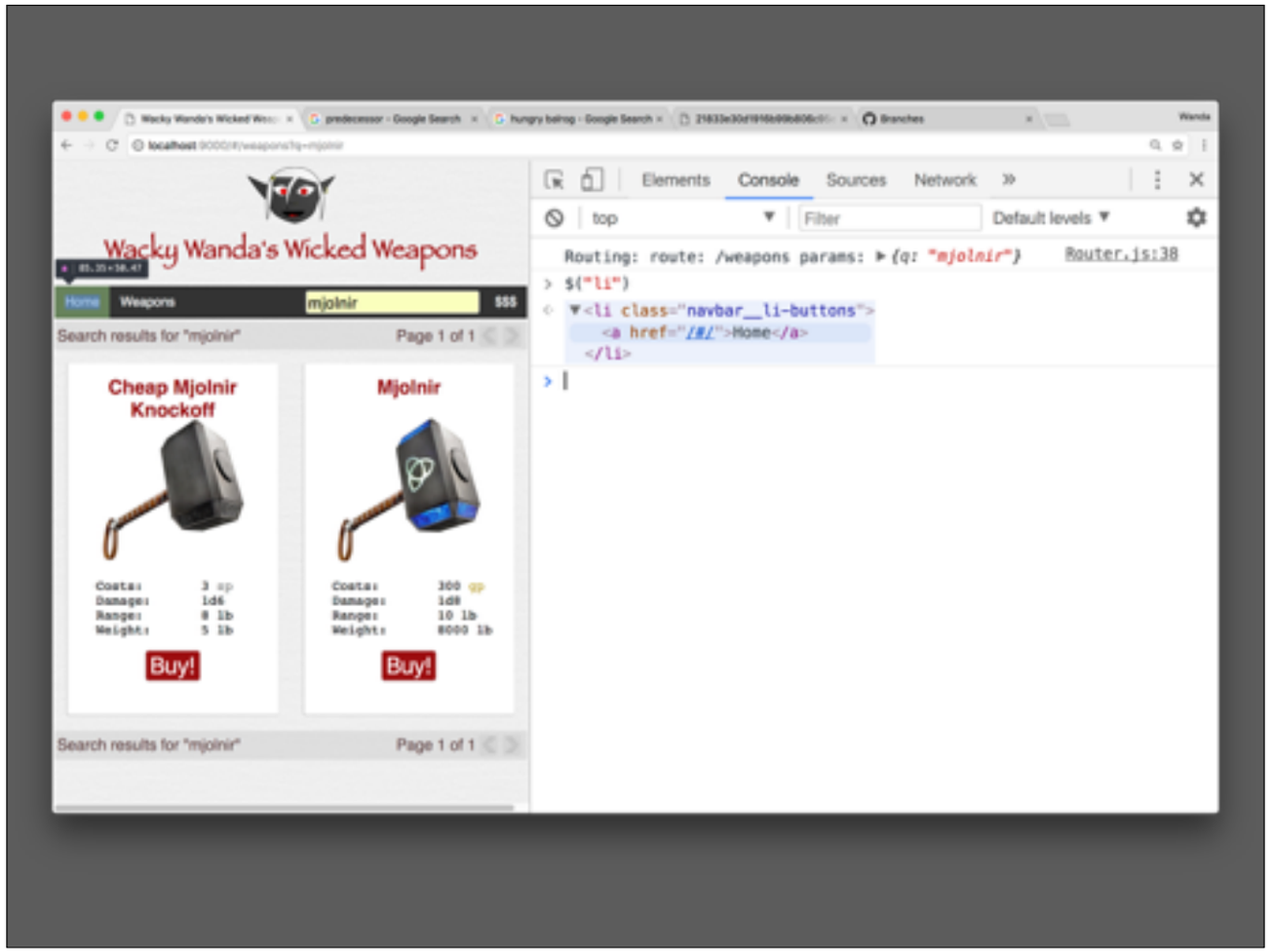


Yep, if I query that enchanted value it brings back **false**.

Now we know

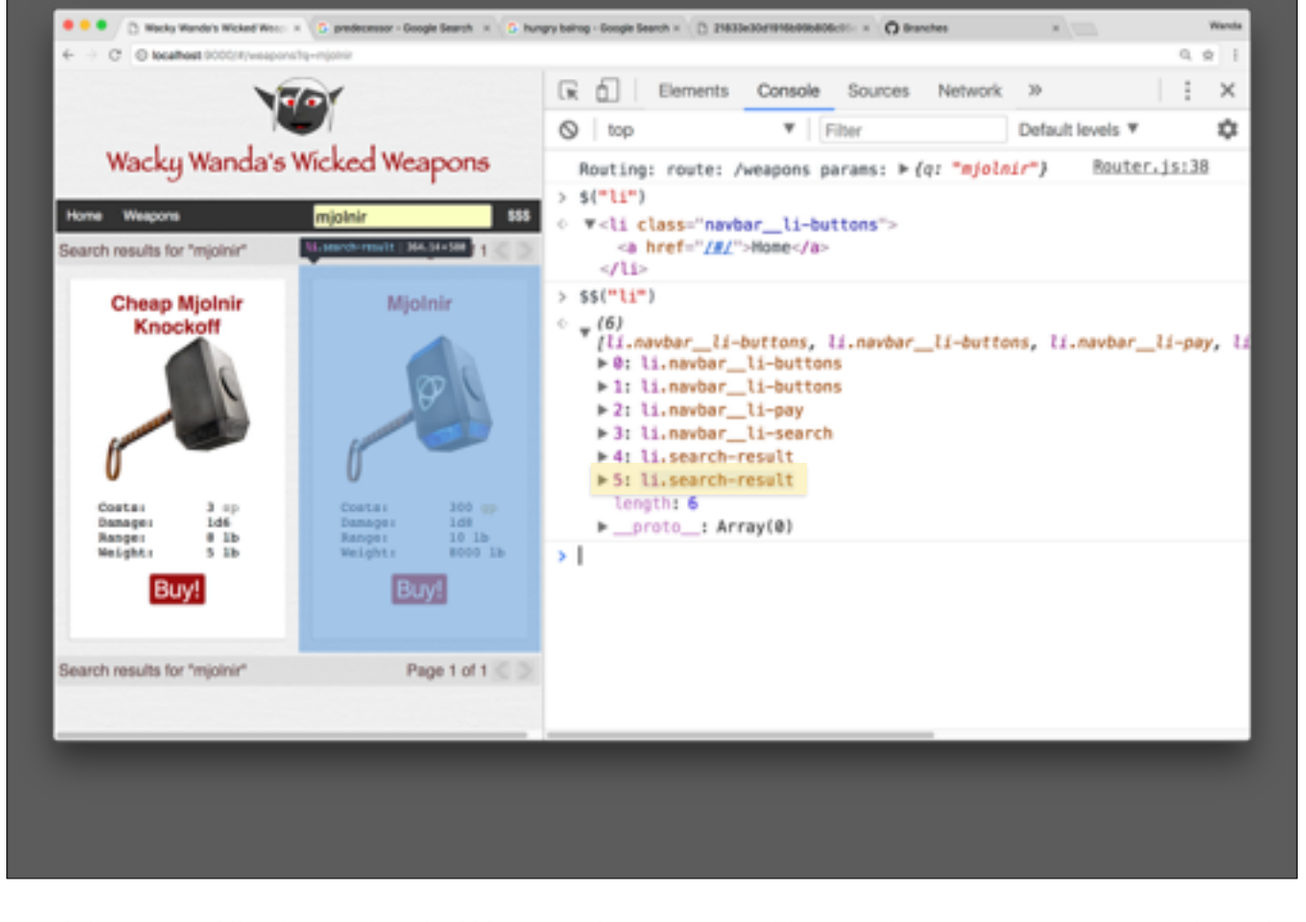


Let’s switch back to side docking again. I can use **CMD + Shift + D** to toggle back to my last Dock position on a mac. Then we’ll go fetch Mjolnir using the weapons search on the page.

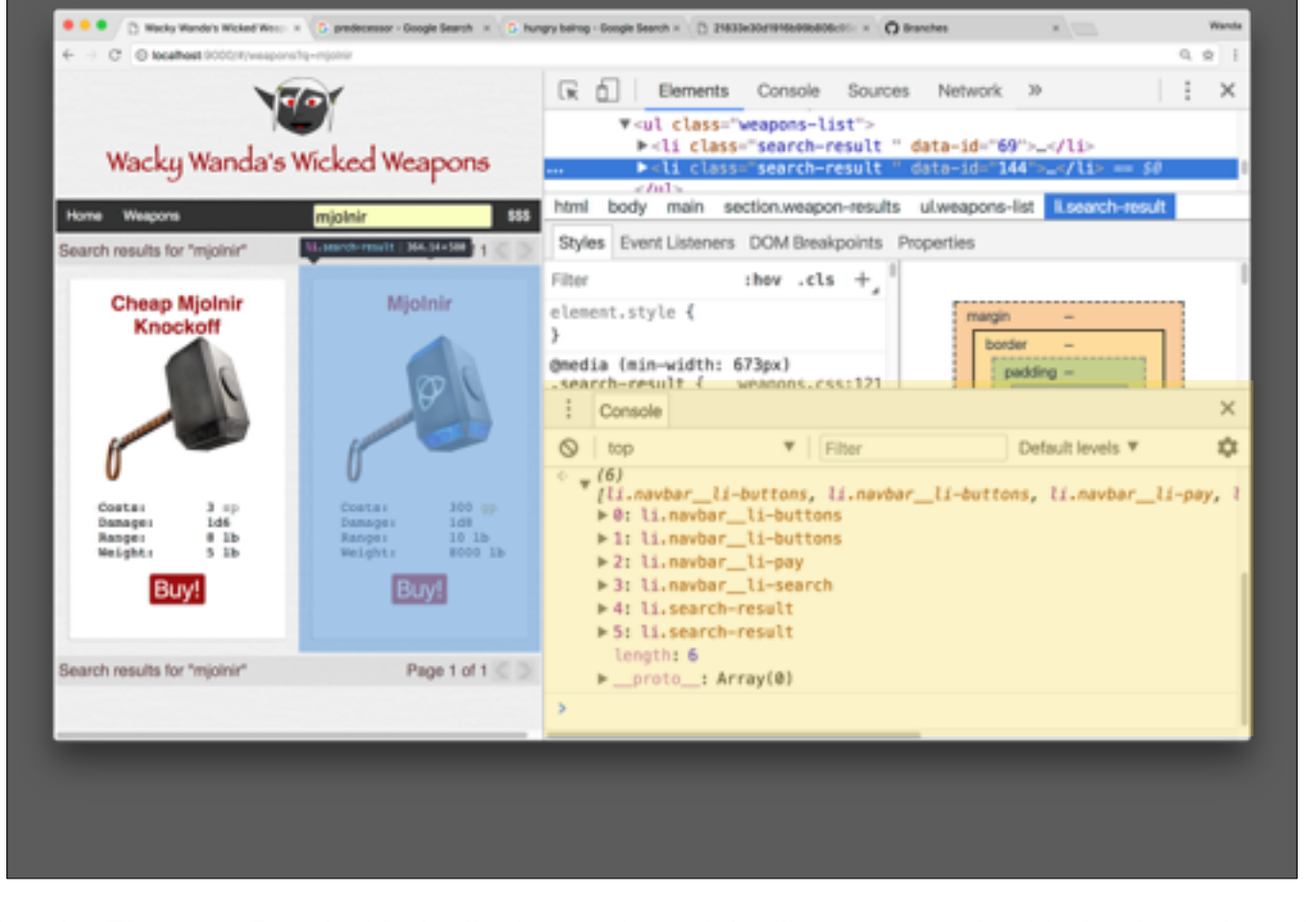


We now need to look up Mjolnir's id. If we had jQuery we could just do a query for all the list items on the page by using **$("li")**.

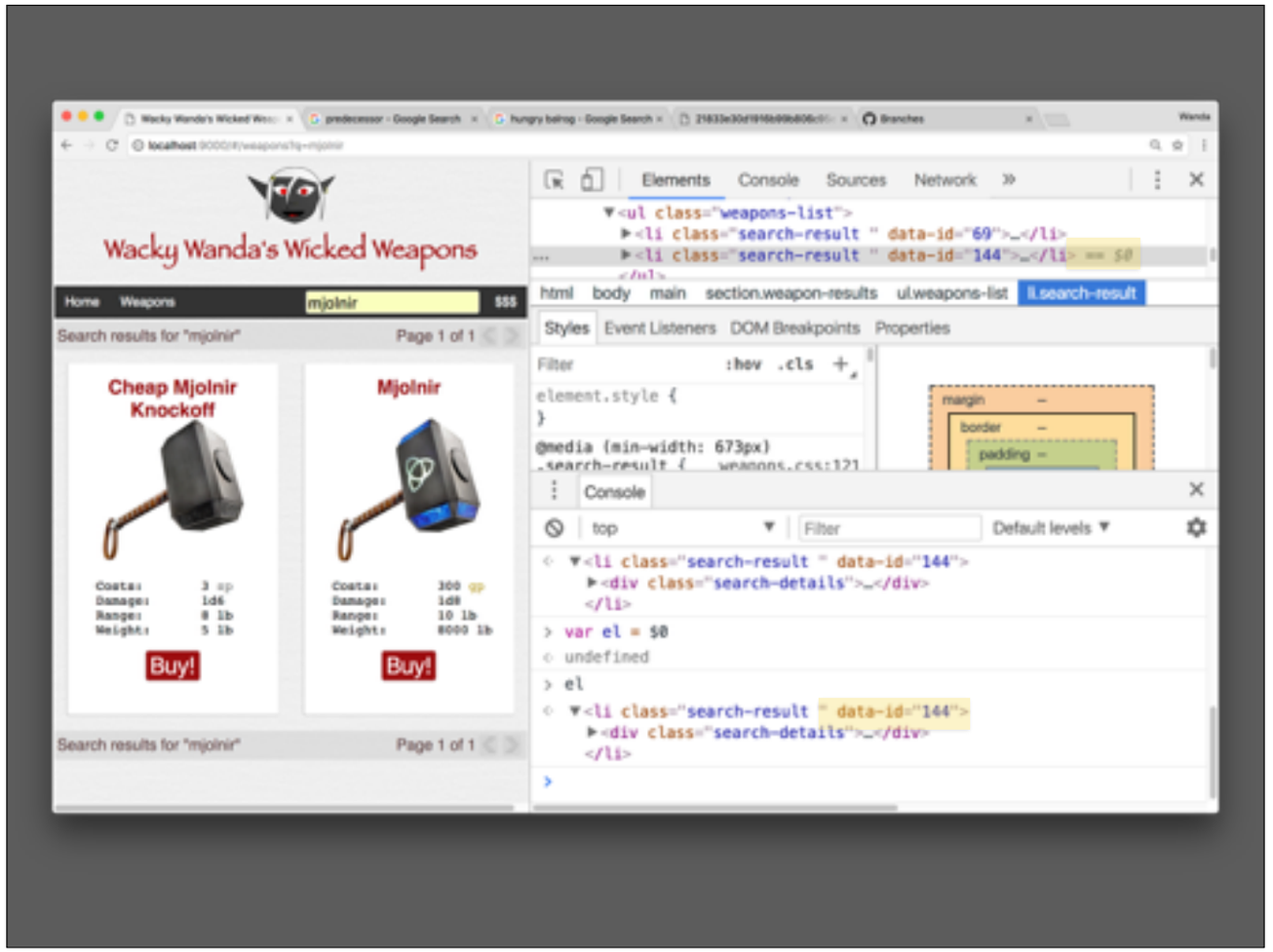
=> $("li")
… well it almost worked anyway! It turns out that Chrome does the equivalent of javascript's document.querySelector() when using the $ function to select elements. The only trouble is that only gives us the first list item, and it's the wrong one. It's one of the nabber list items.



No matter! If we try again using $$ we get a full array of list items back. If I open the array and hover the mouse over each item in turn it will highlight each element in turn on the page. Looks like Mjolnir is the final list item.



If I click on that list item I'll be redirected to the Elements Panel with the list item selected. I still want to work with the Console so I'll just open up the Console Drawer by pressing Escape.

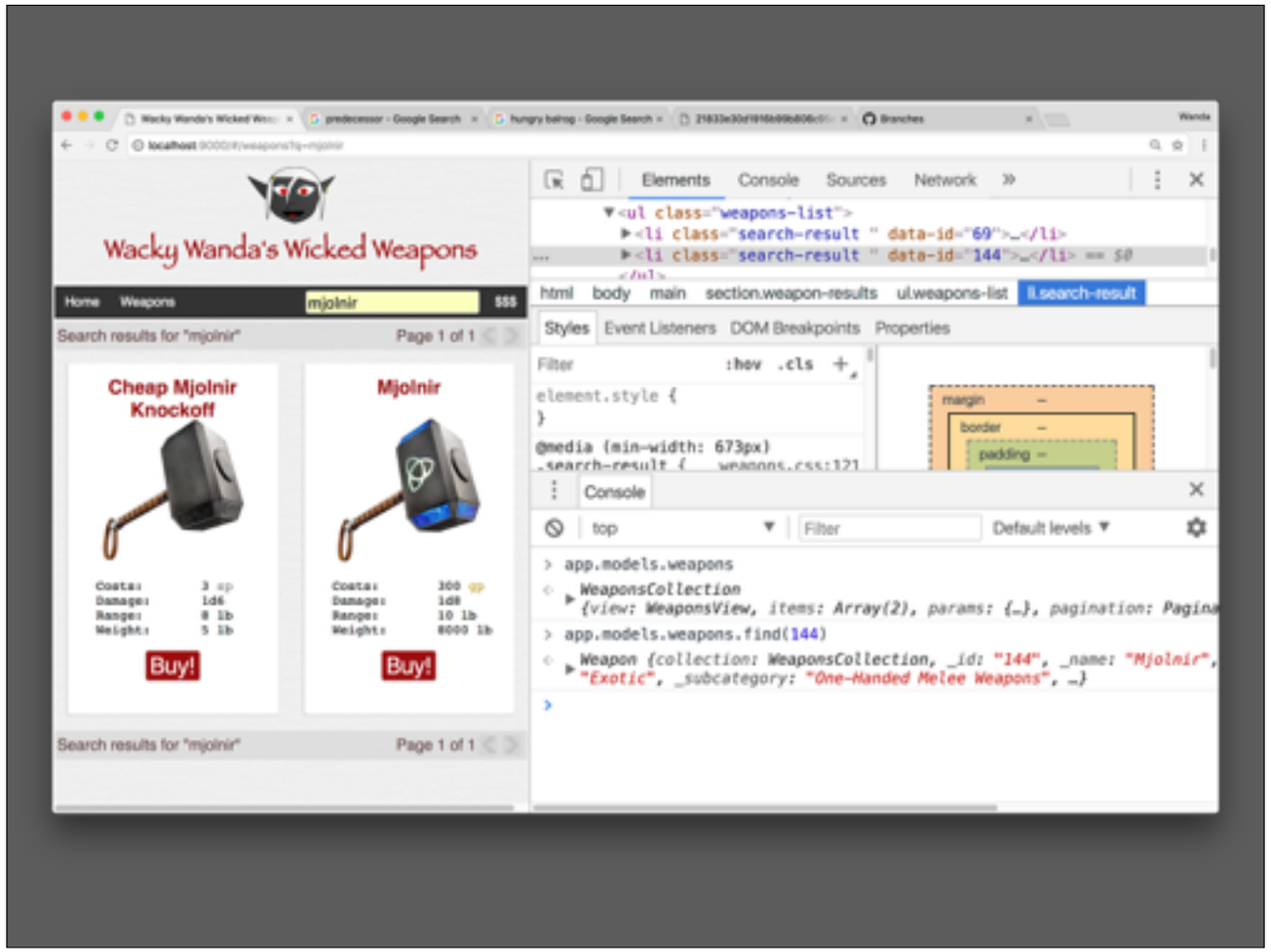


Ever noticed that "**== $0**" shows up at the end of any element that get's selected? Well this means the element is accessible from the console. You can also access previously selected elements with $1 through $4, but I'm not sure why you would.

Let's make use of $0:

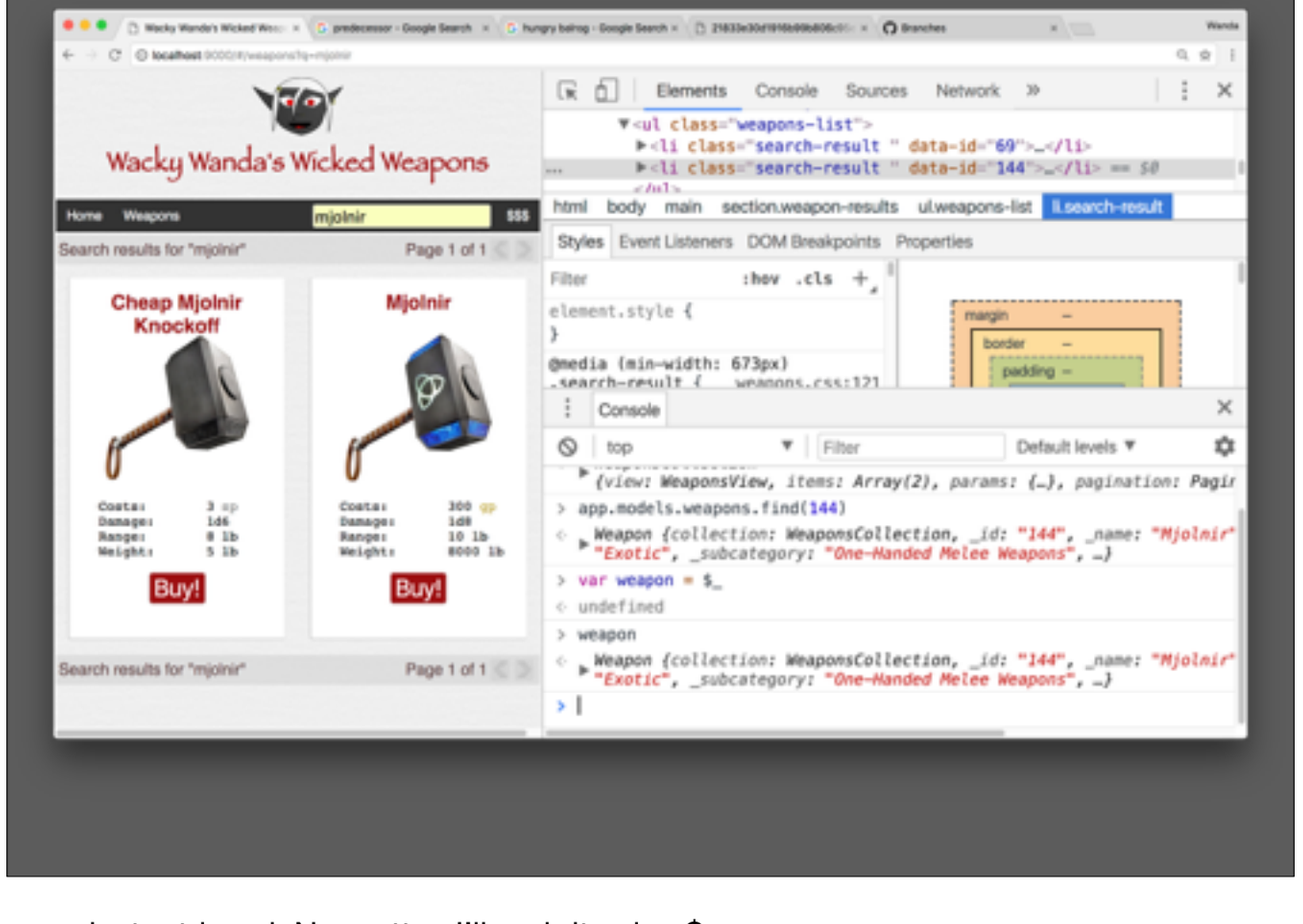
=> **var el = $0**
=> **el**

Looking at our captured element there seems to be a data-id of 144. We can make use of that to retrieve and enhance Mjolnir.



So as we learned earlier we can look up the weapons collection model with **app.model.weapons** object.
We can use the finder method to pull back item no 144:

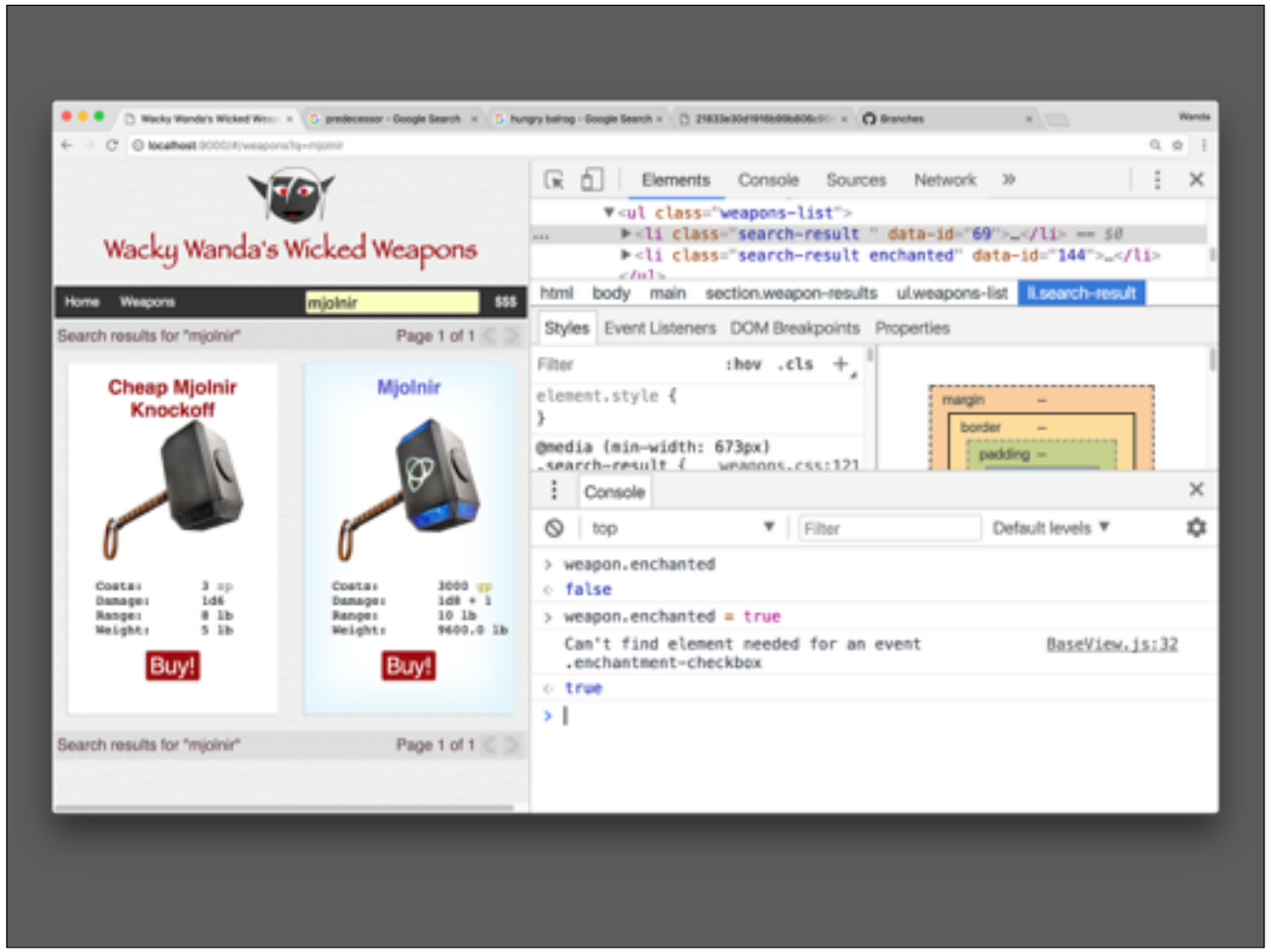
=> app.models.weapons.find(144)



I forgot to assign a variable to the weapon we just retrieved. No matter, I'll grab it using **$_**

=> **var weapon = $_**
=> **weapon**

$_ will always give you the result of the expression last evaluated.



And now we have Mjolnir at our disposal we should be able to enchant it:

=> **weapon.enchanted = true**

Yep, Mjolnir is all glowy, buffed… and completely impossible to lift. 9600 pounds, wow. Oh well, that’s someone else's problem.

Sources Panel

Enchanted Weapons Part 2

Now we just need to wire up our fronted to make use of these enchanted weapons. This involves writing code and making use of the debugger so for this we will be using the Sources panel.

Sources Panel

Enchanted Weapons Part 2

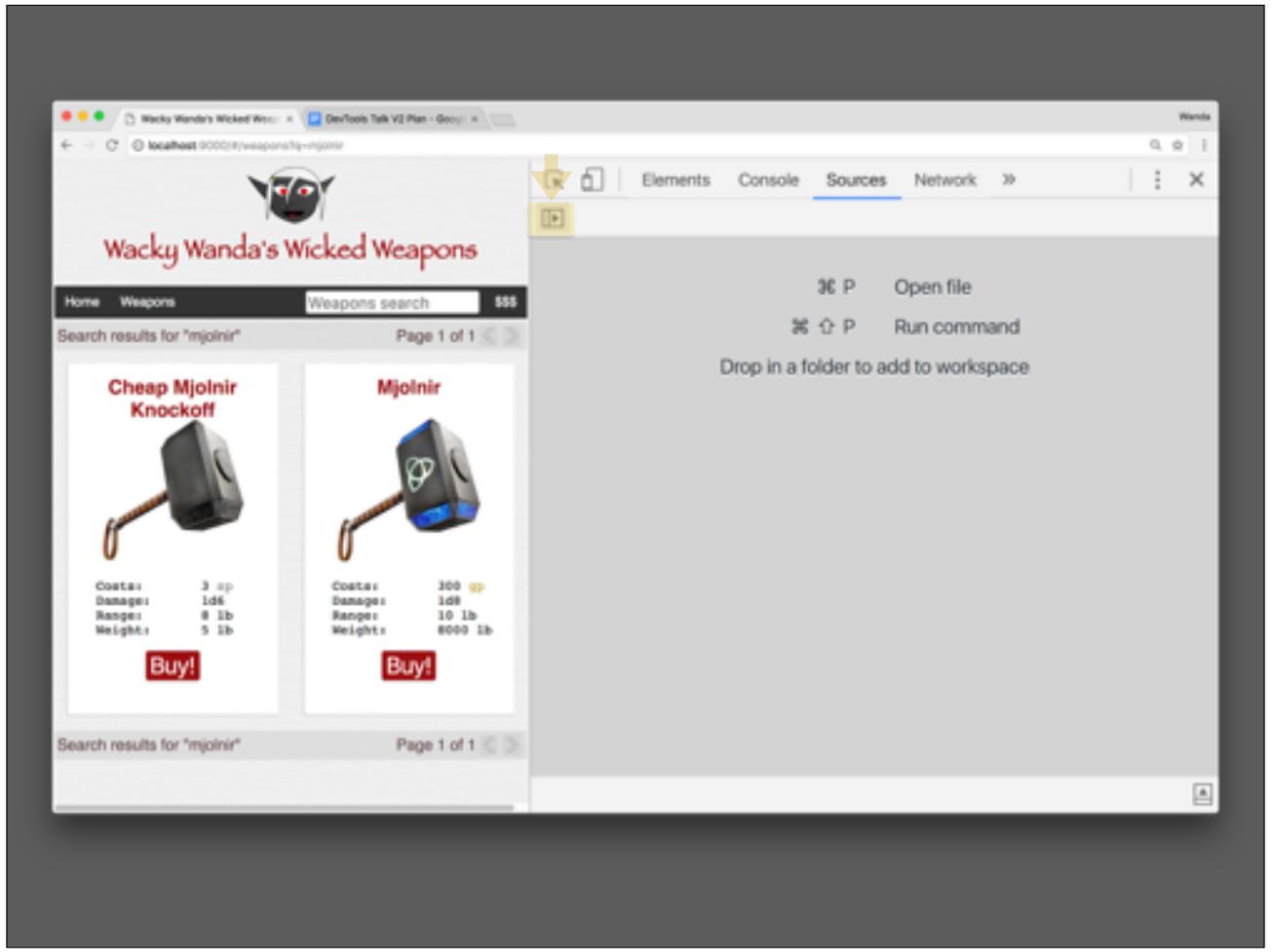
Tasks:

Template - Add a checkbox

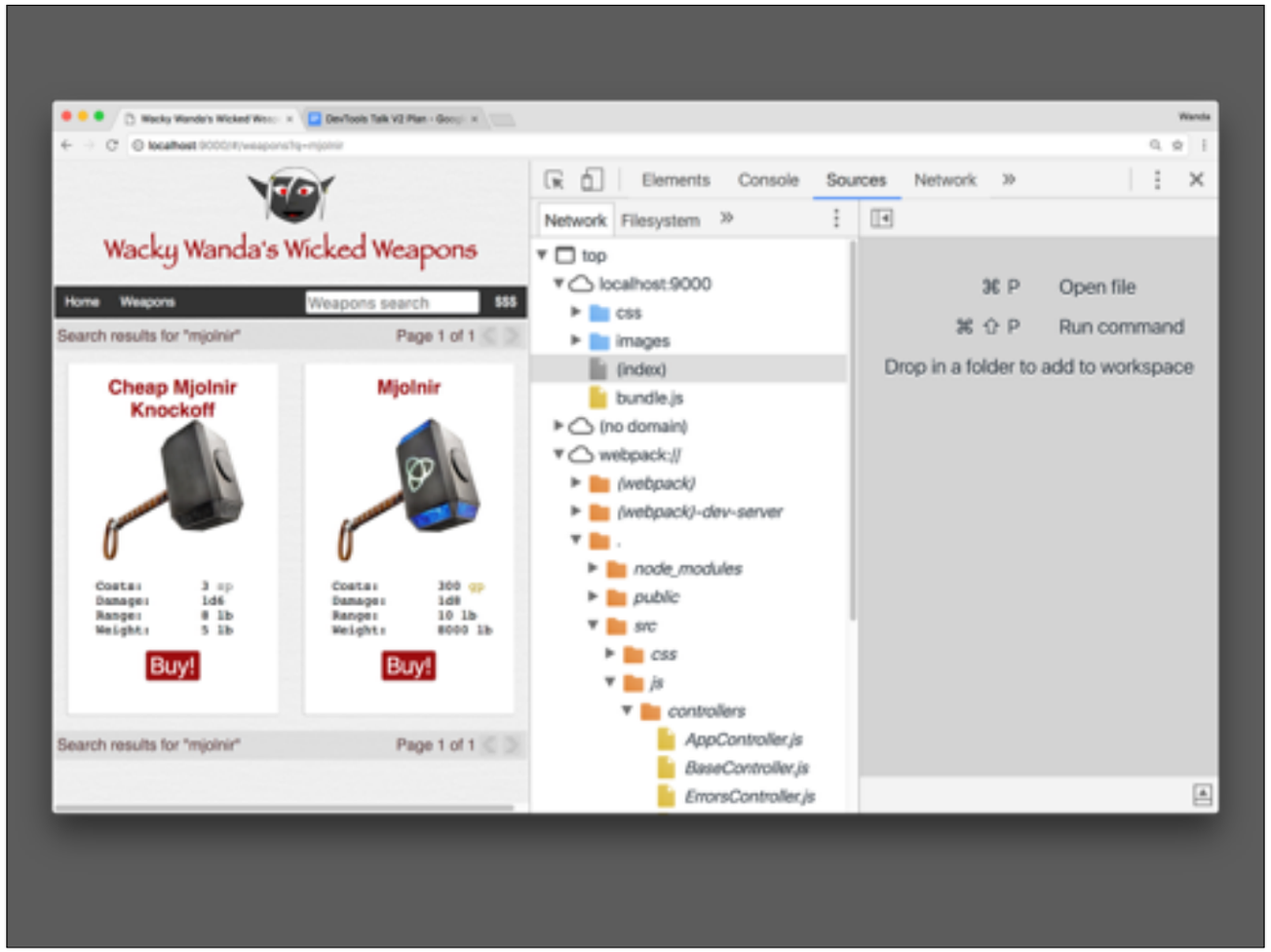
View - Add a checkbox event listener

Controller - Update the weapon model

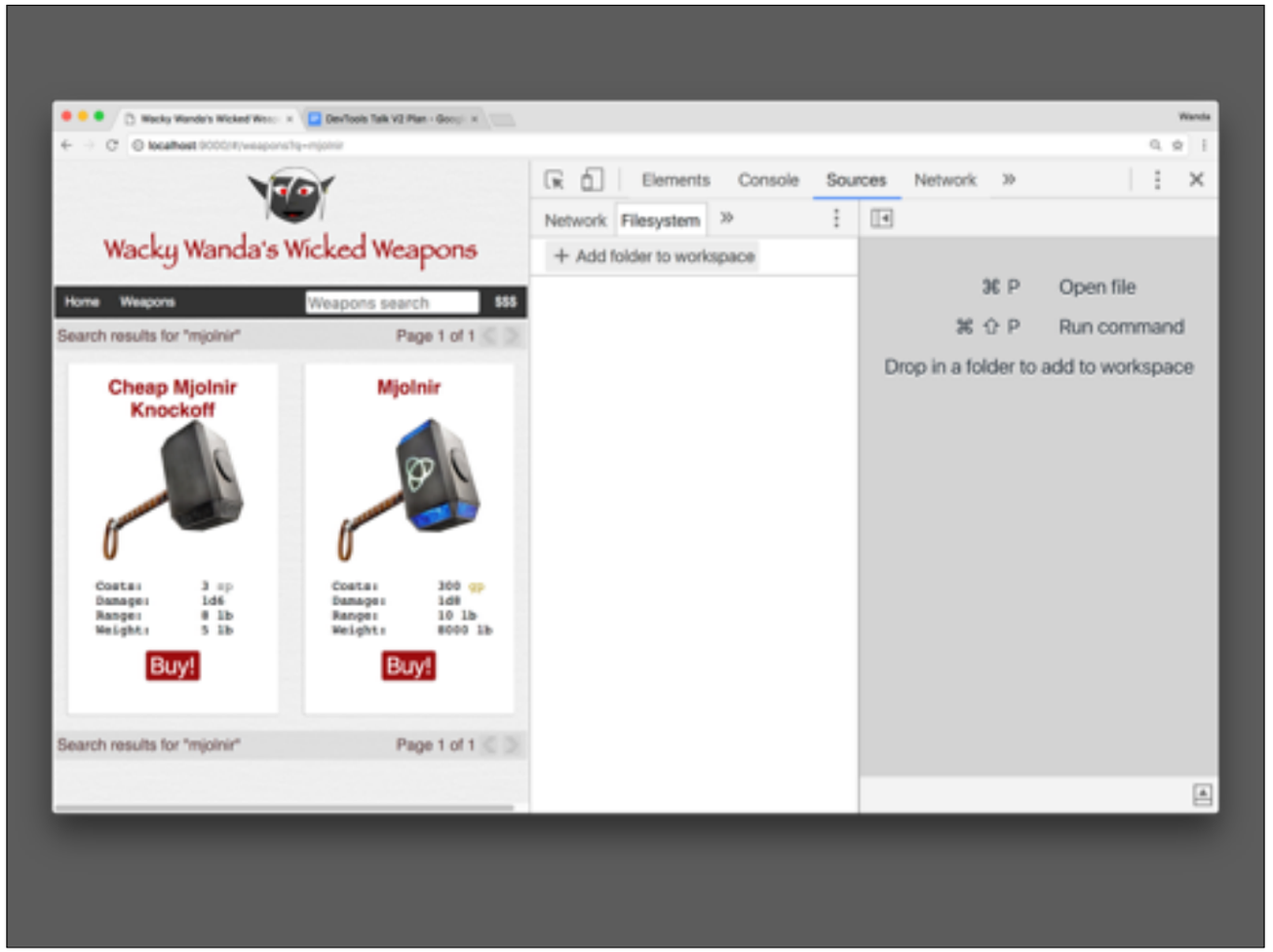
There are 3 tasks we need to preform to make this happen.



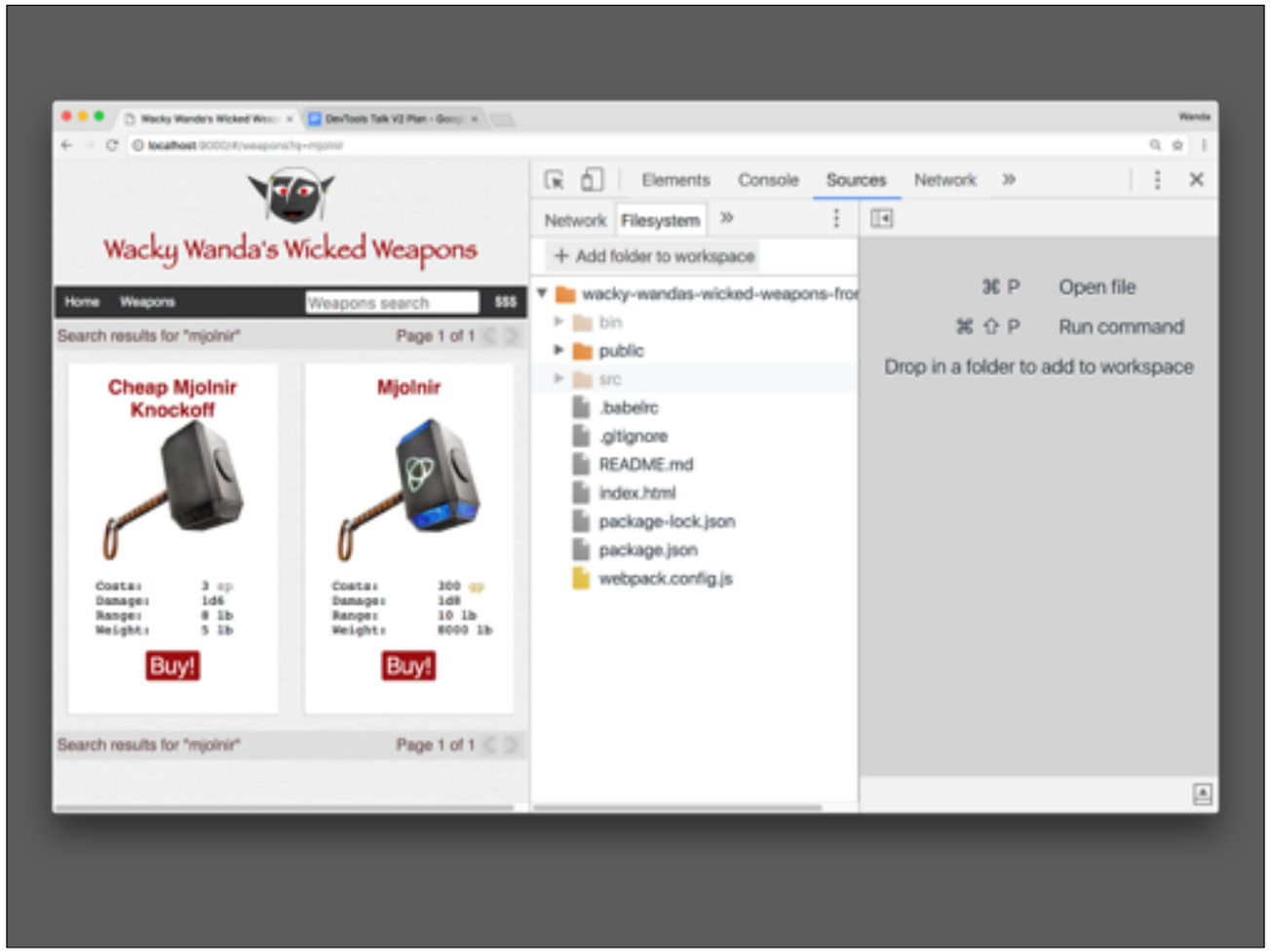
Before we get down to work let’s have take a look around the Sources Panel. We’ll start by checking out the Navigator sidebar on the left.



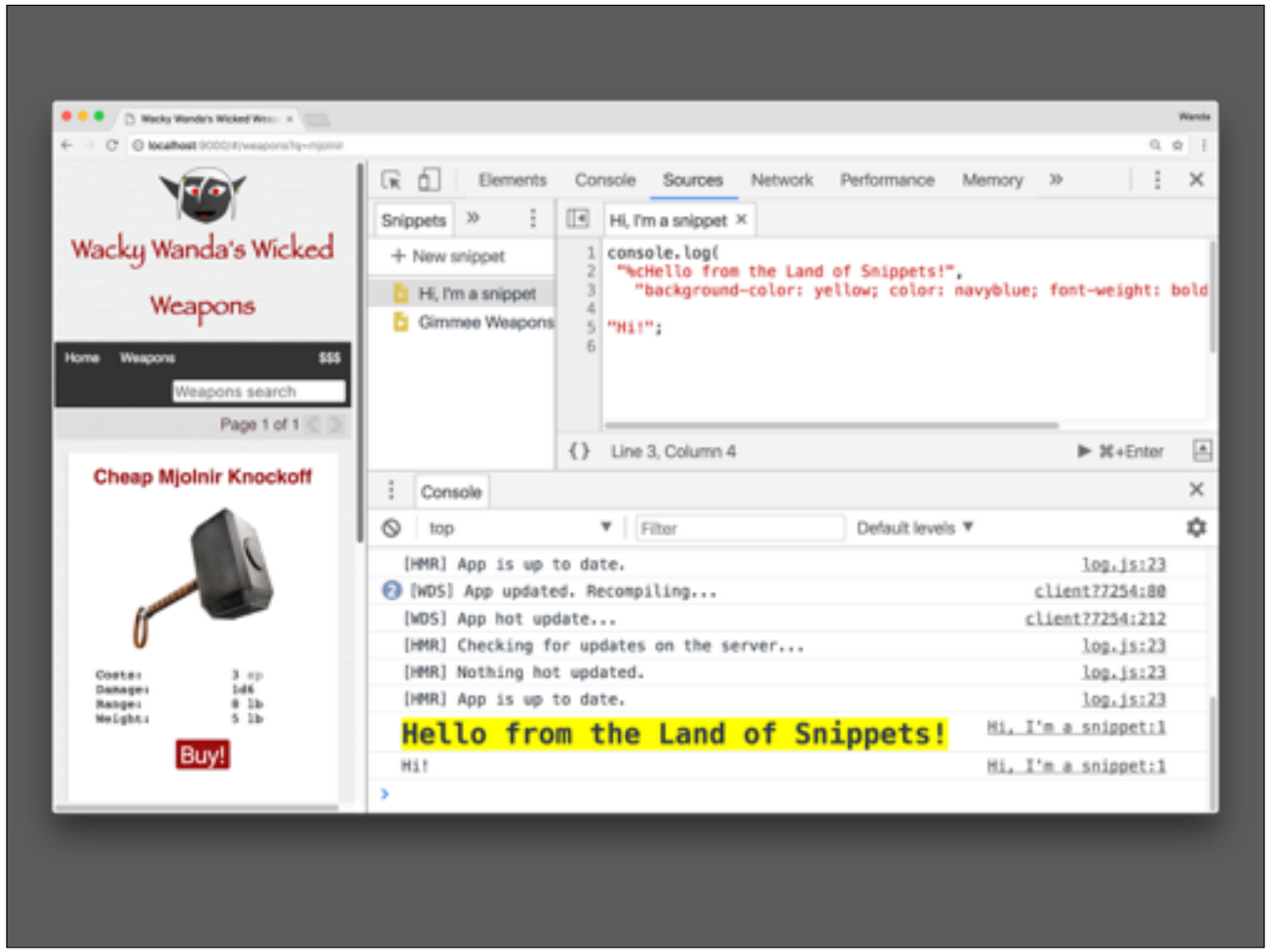
The Network tab shows all the Resources served up on the page. The main “localhost:9000” area represents the actual files for this project. There’s not much there because webpack is compacting all the javascript and styles. Under “wepack://” and “.” we have all the virtual files that webpack is working with. Webpack is mapping all the original files to the compiled resources using source maps.



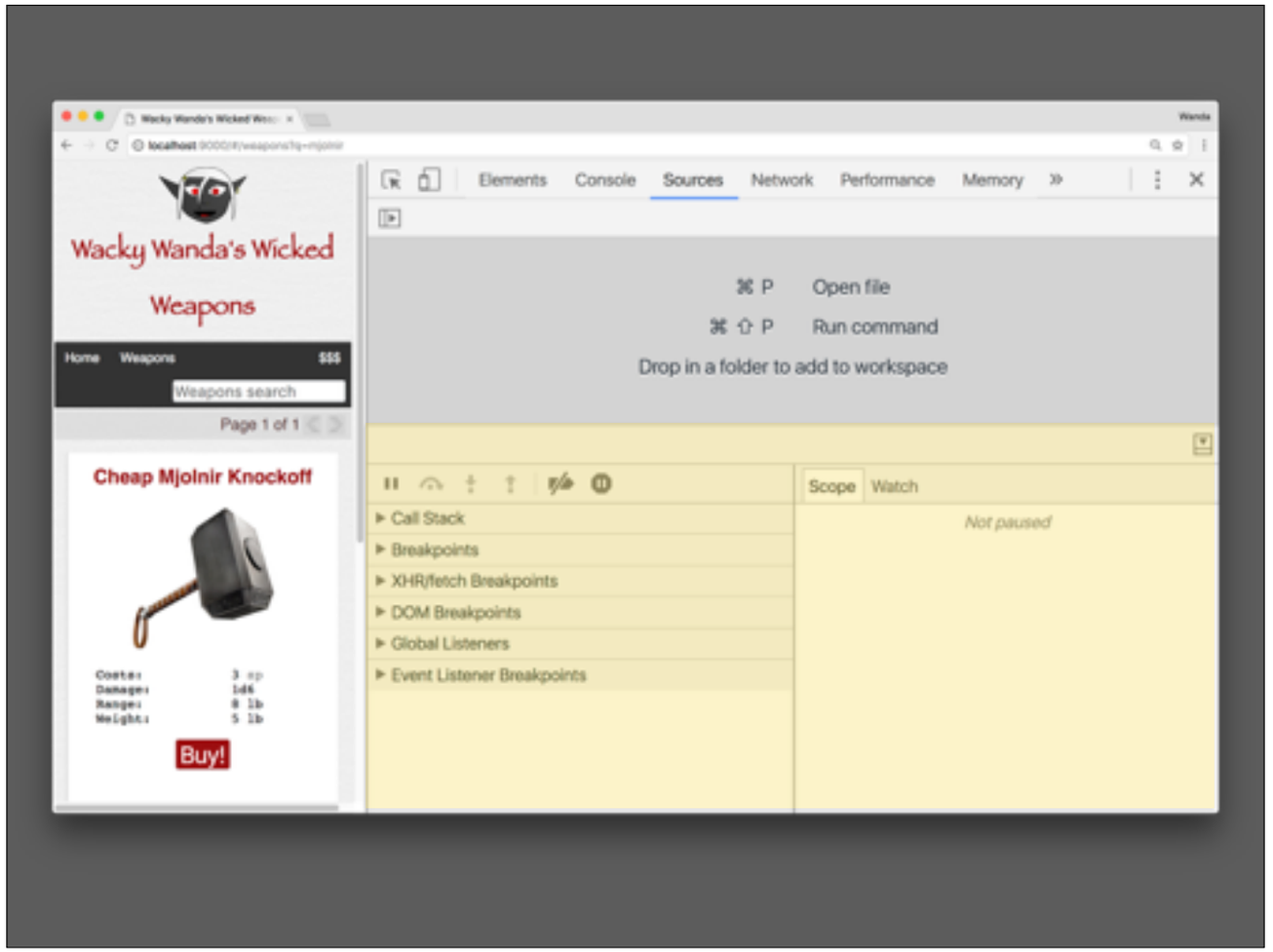
We can use Filesystem to tell Chrome where our original files are so it can attempted to update them as we change them. Because we're using source maps it doesn't do that great a job at the moment. We'll turn it on for working on the first task where we add a checkbox, but we'll turn the feature off again after that because it tends to mess up the breakpoints. I expect things will improve as bug fixes happen.



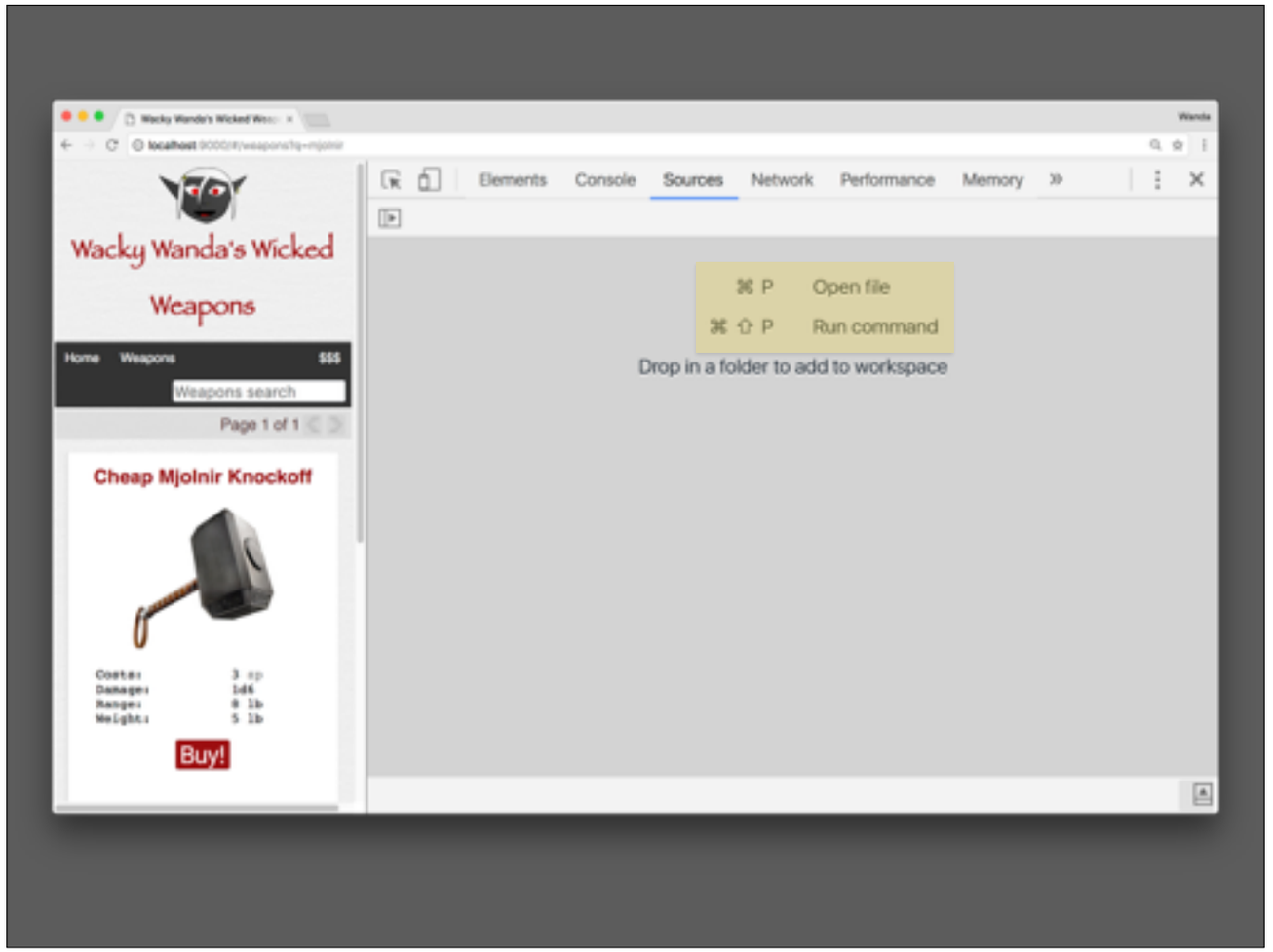
I went ahead and added the whole project folder. Chrome will attempt to map the files to the server hosted files as best as it can. Because of the source maps it will only have limited success for this project.



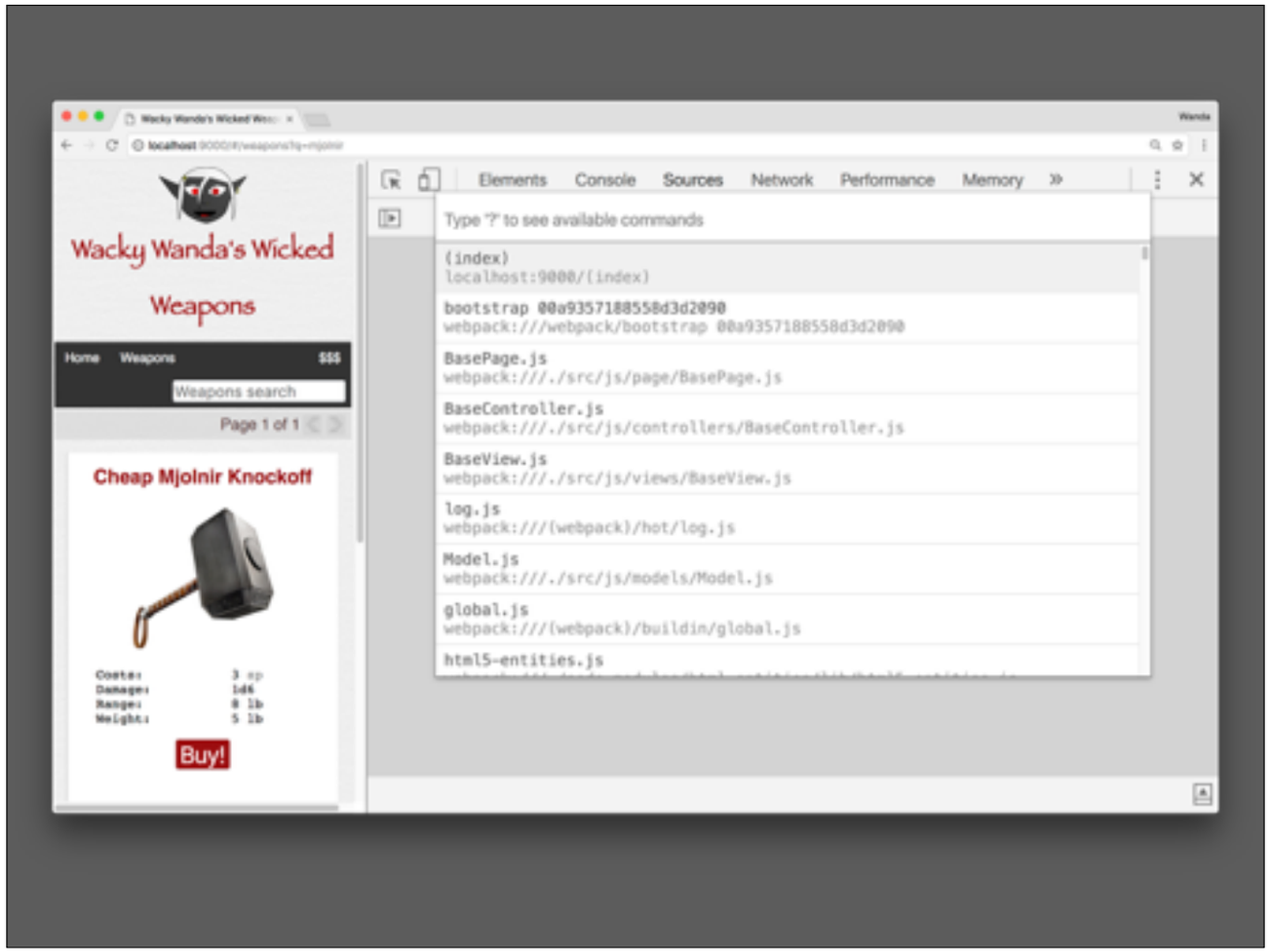
Snippets allow you to write macros which will work across multiple applications. For example I have a Snippet here that will print a glowing message. All that's happening code wise is I log with %c in the text, and add a second argument which is a mini stylesheet.



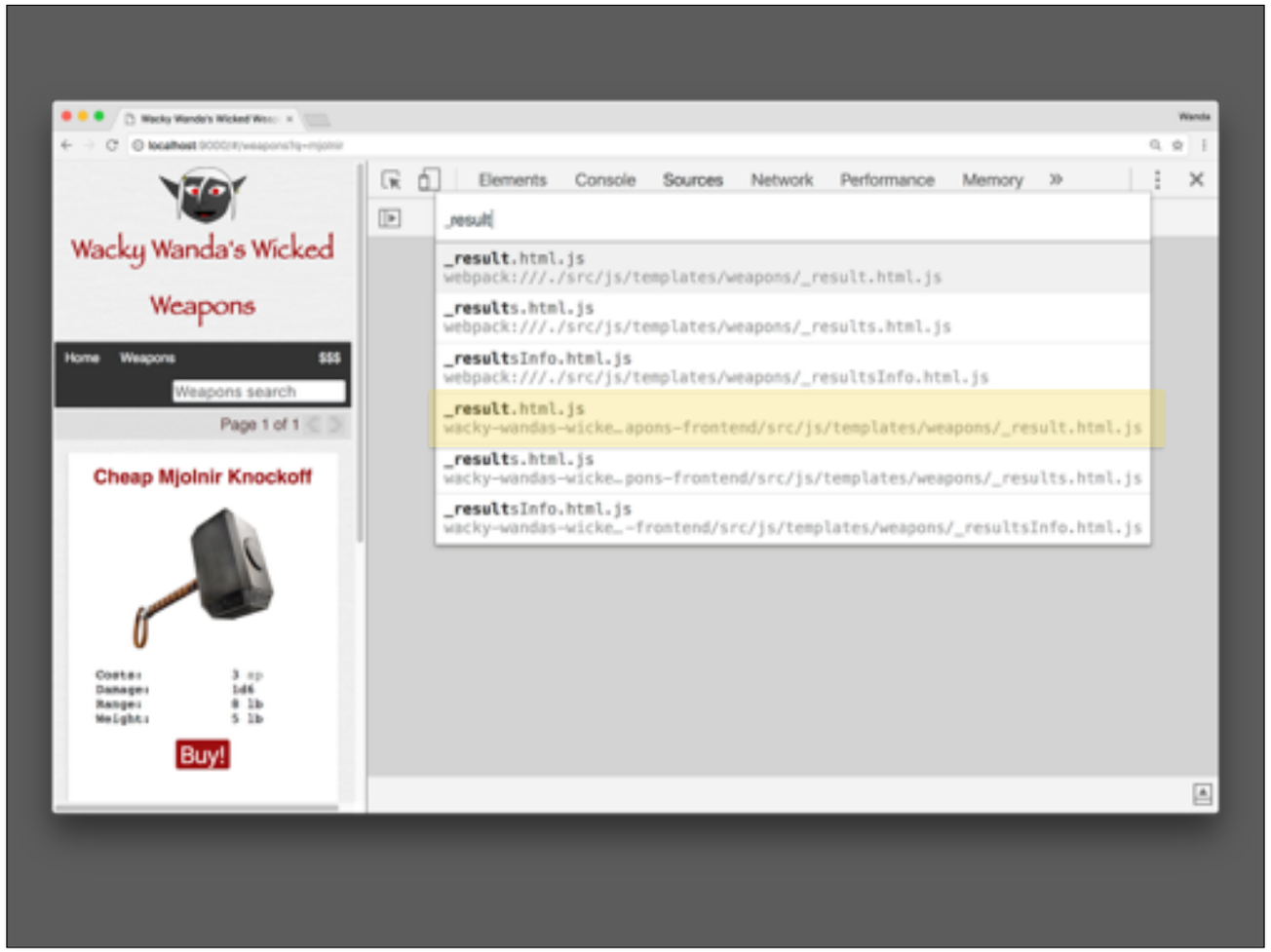
The other side side tab contains a whole bunch of debugging goodies. Mostly stuff you'll find in all good IDEs including the ability to step over code, view the call stack and work with breakpoints. Rather than dive into that right now we'll come back to it when we get around to some actual debugging.



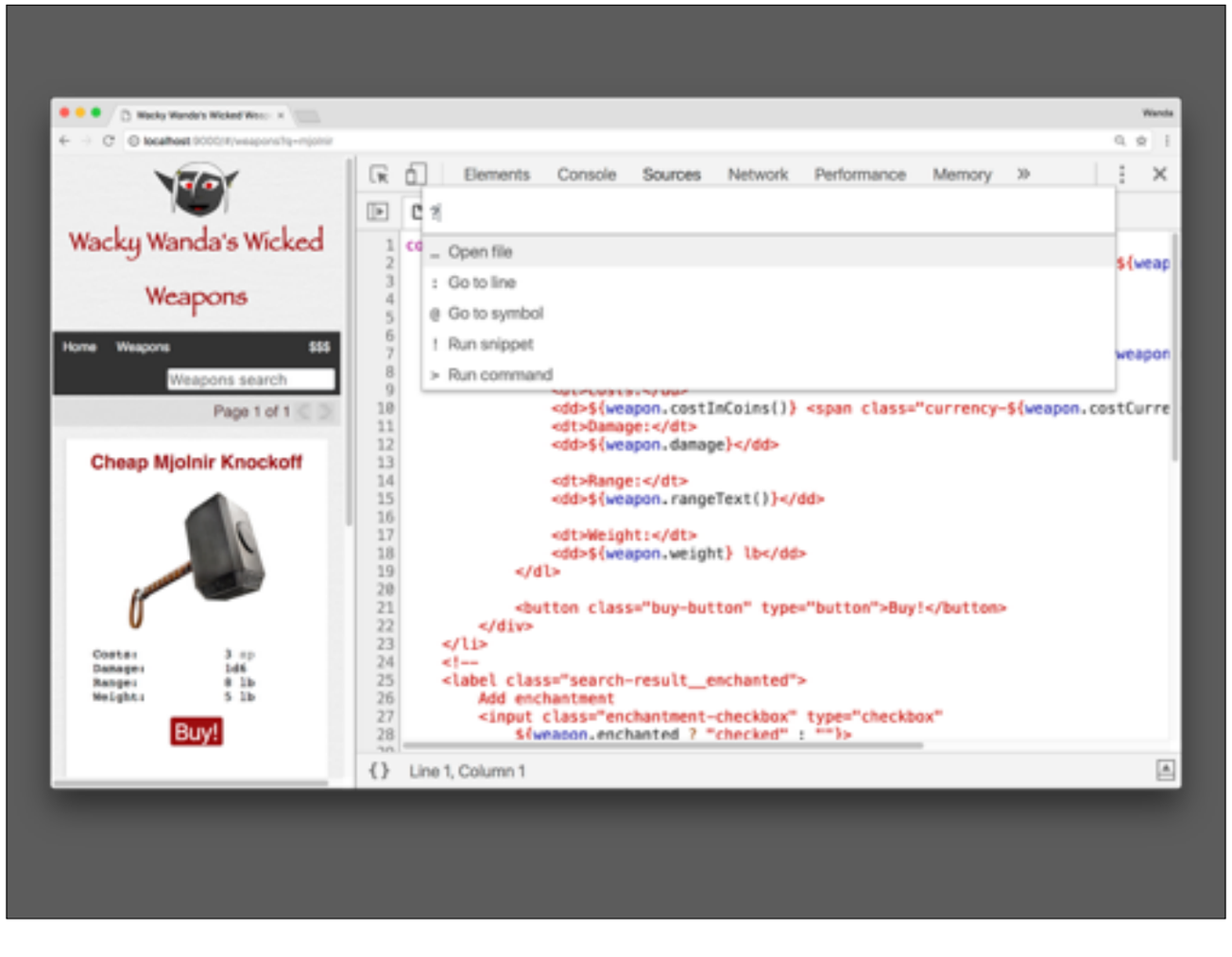
The central part of the panel if for displaying and debugging source. If no files are present it contains a rather mild mannered suggestion: “CMD+P Open File”. Let’s give that a whirl.



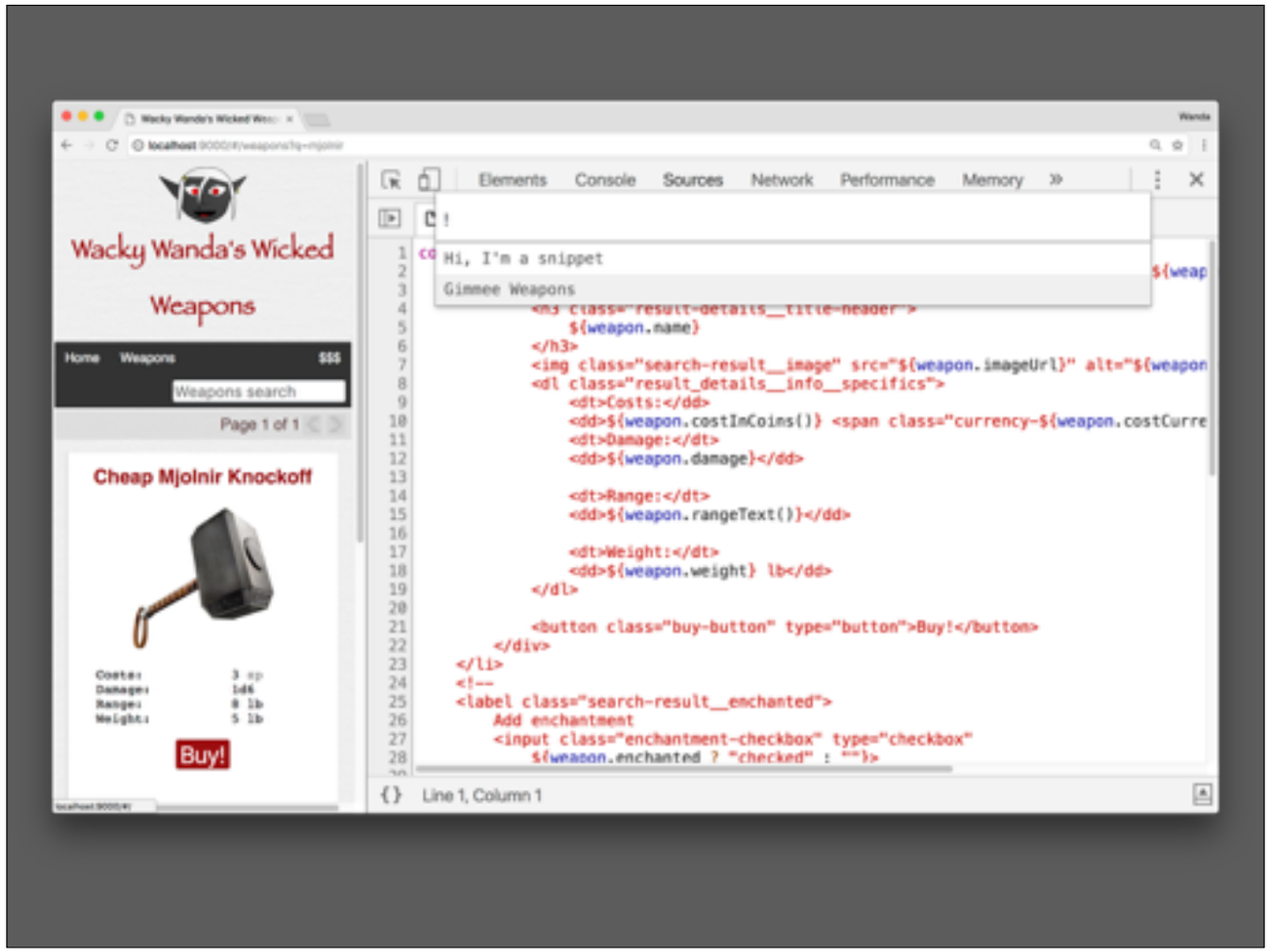
“Open File” means this really nifty Fuzzy Finder searcher utility.



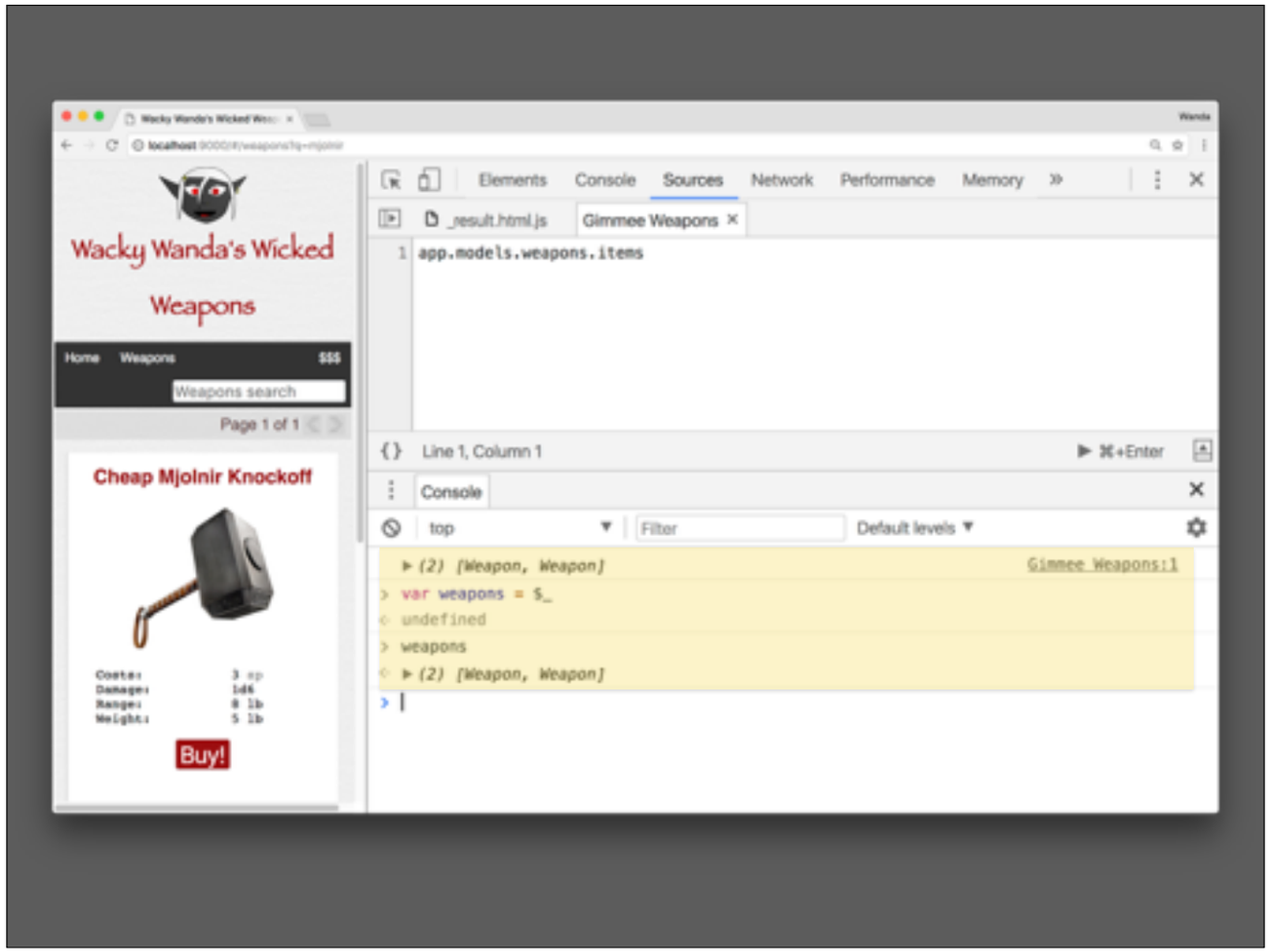
We'll try using it to open the javascript template where we need our checkbox to be, "_result.html.js". There are 2 versions of this file floating around, one is the webpack source map virtual file, and the other is the original source available through the Workspaces feature. We want this one for now.



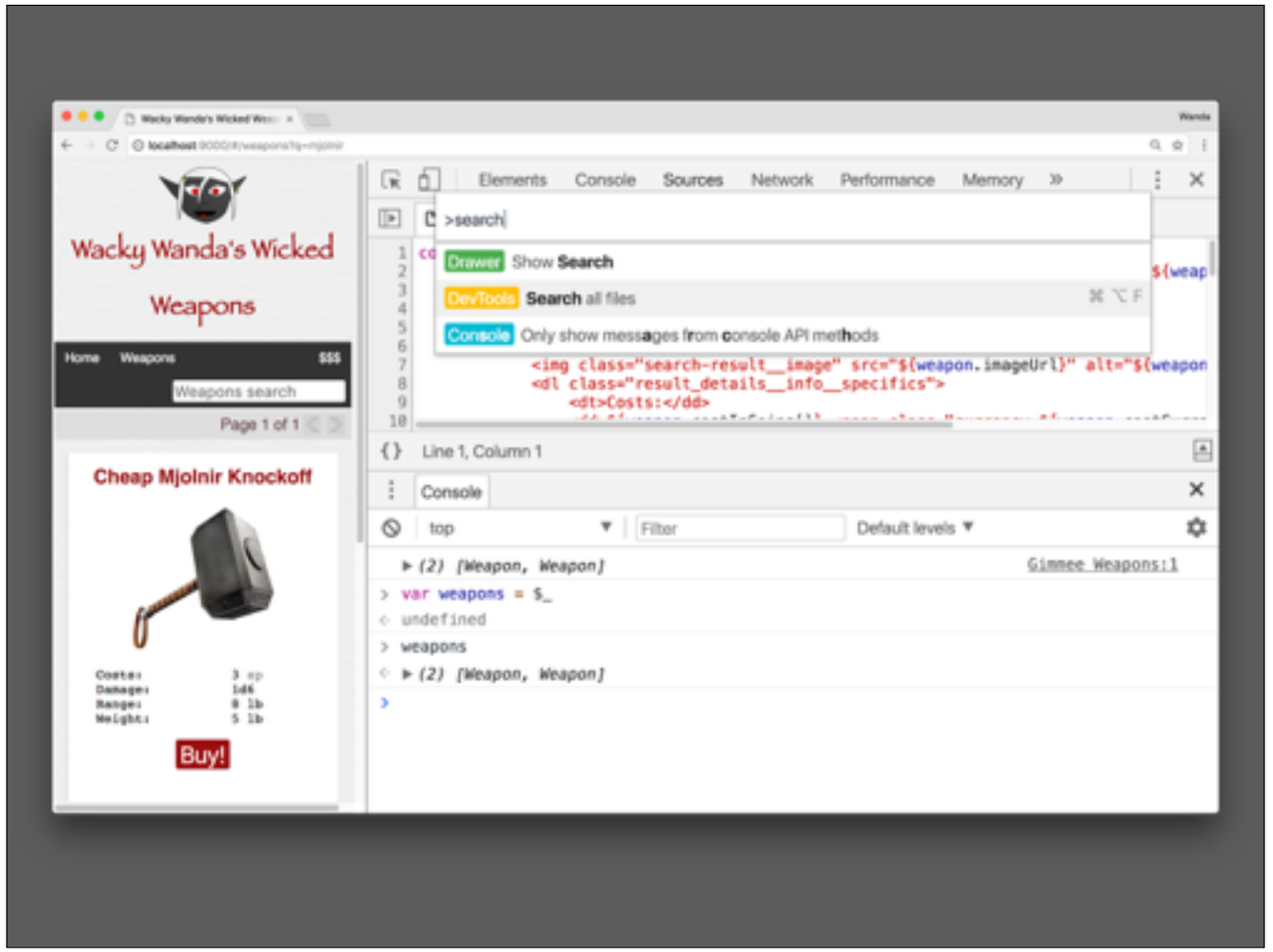
But there's more! If we CMD+P and choose "?" it looks like we can also use ":" to jump to a line, or "@" to jump to a function definition.



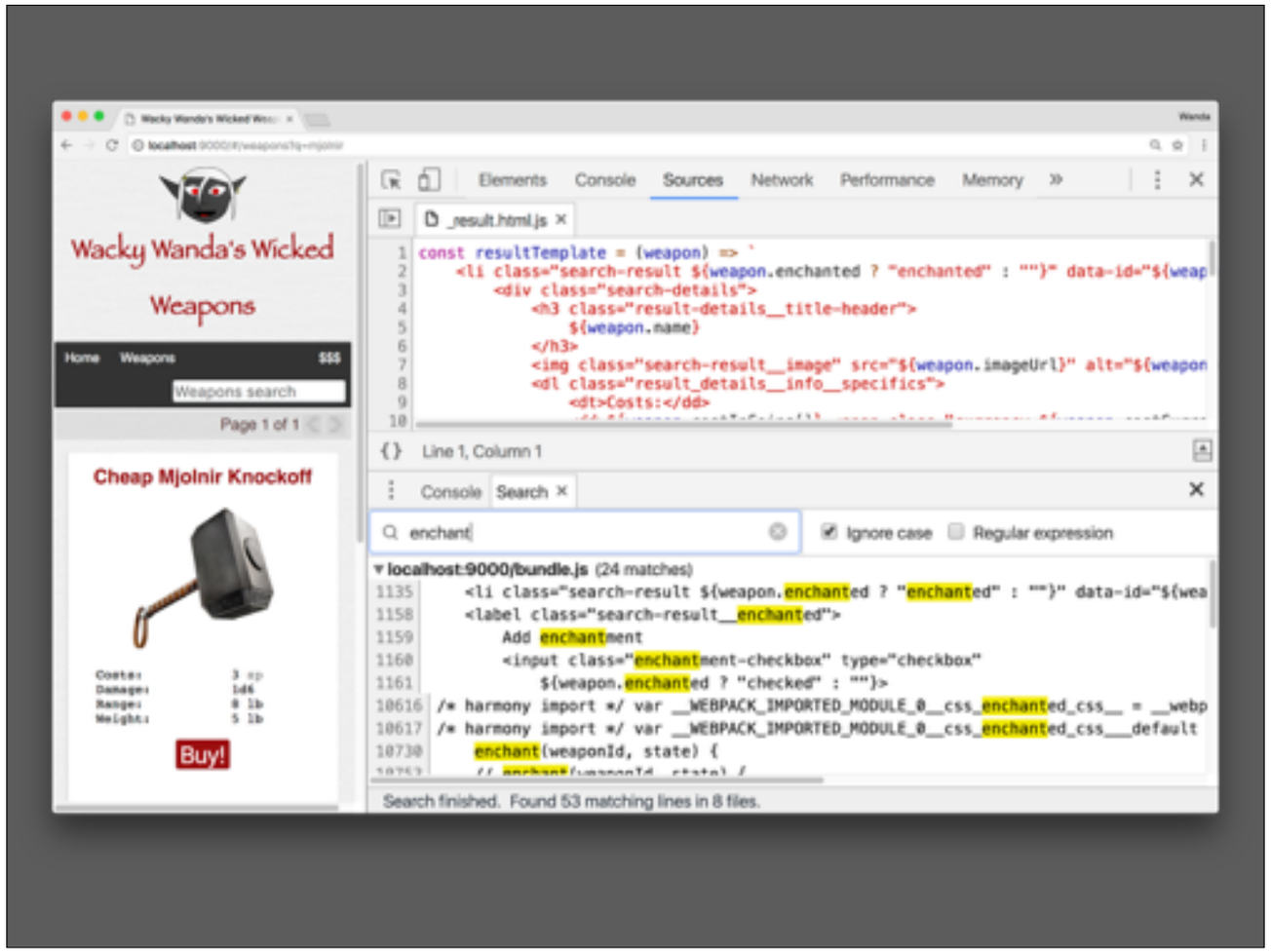
If we hit “!” we can run a snippet. Let’s try that with “Gimmee Weapons”.



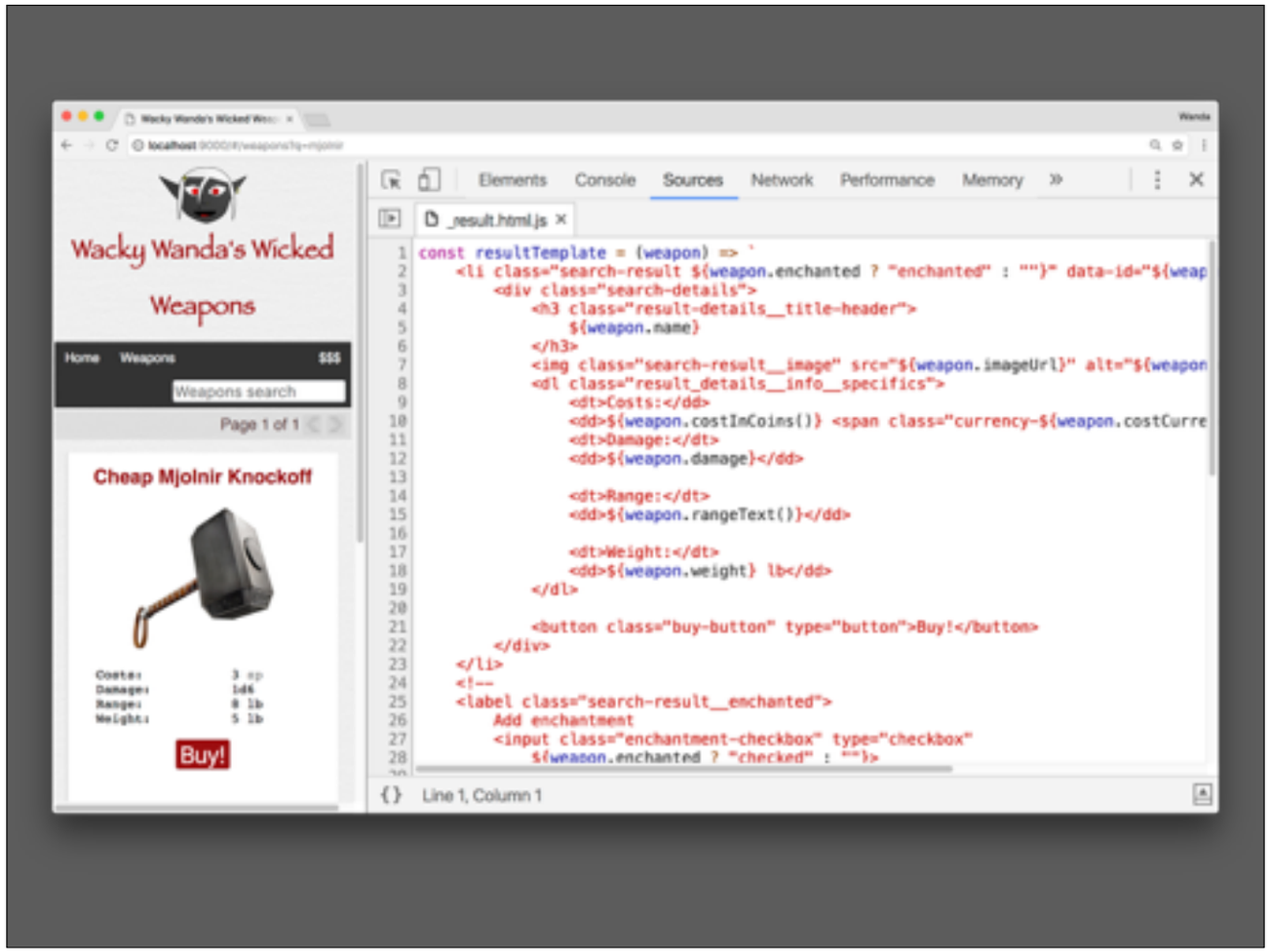
This works, I have the pages's weapons now inspected in the console. Only problem is we don't have them attached to a variable. Not a problem, we can use "$_" to attach it to a variable as it was the last evaluated expression.



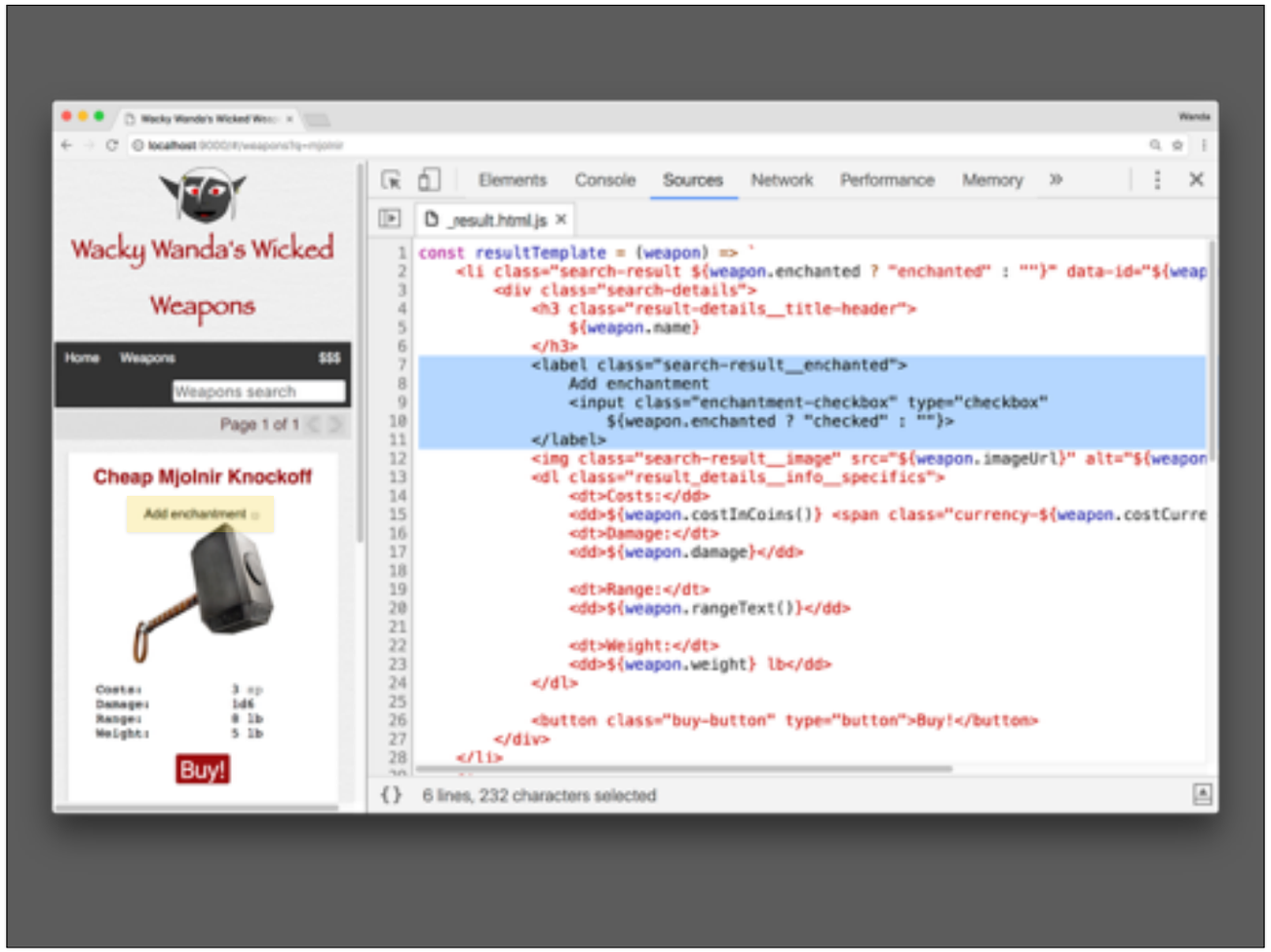
Last but not least we can use CMD+Shift+P to run almost any console command. This can be really handy when you can't remember how to reach a feature. For example I used to keep forgetting how to get to **Search in Files.** If I use this Run Command feature I just type **Search** and there it is!



Search in Files is one of the Drawer based features. For example I can do a search on “enchant” and it’ll show me all the sources on the site that has this keyword.

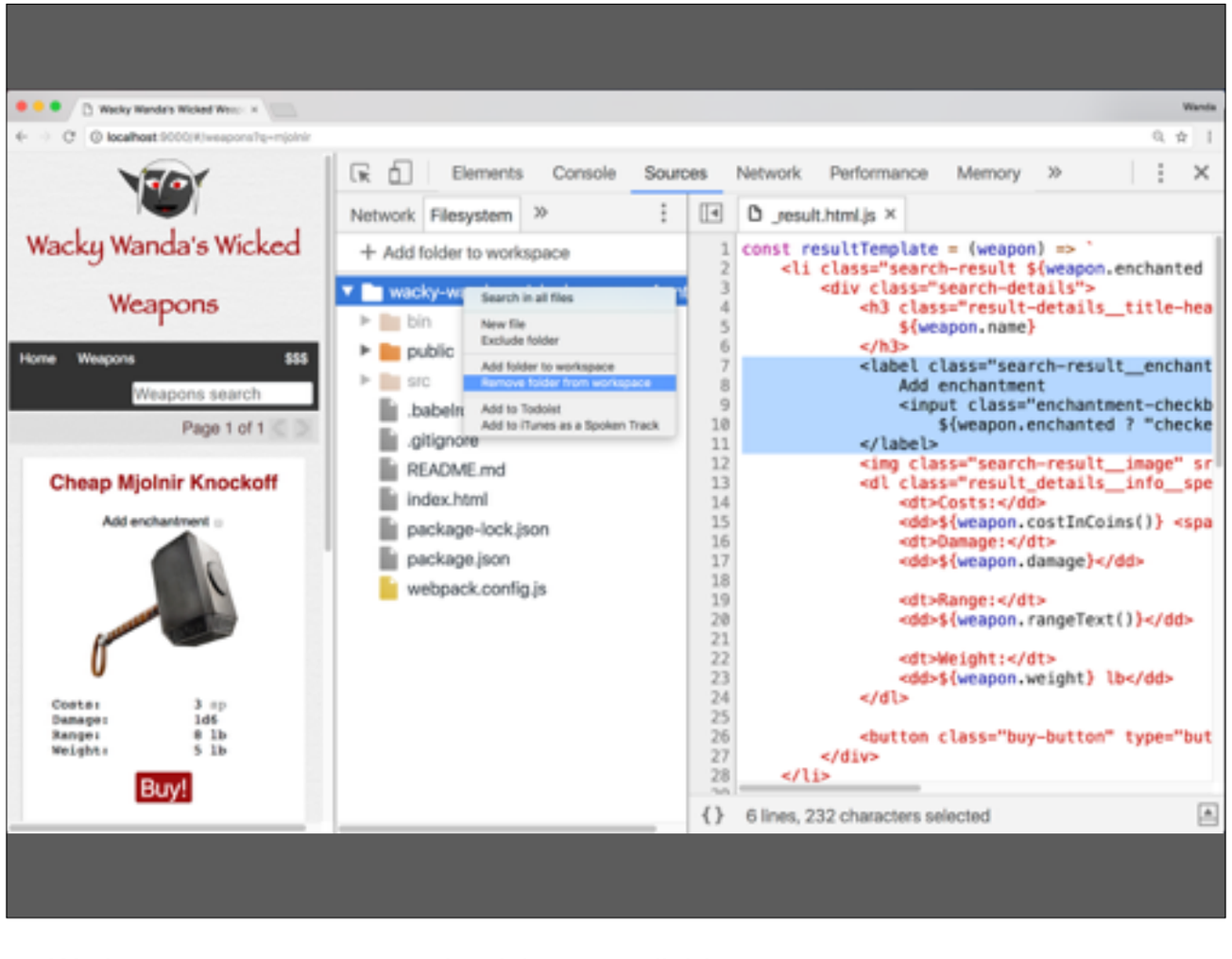


Let’s get started! We have already opened the template we need to modify. Templates in this projects are just ES6 functions that return a multiline string. With ES6 it’s very easy to inject small expressions as interpolated values. We need to go ahead and add the checkbox just above the image.

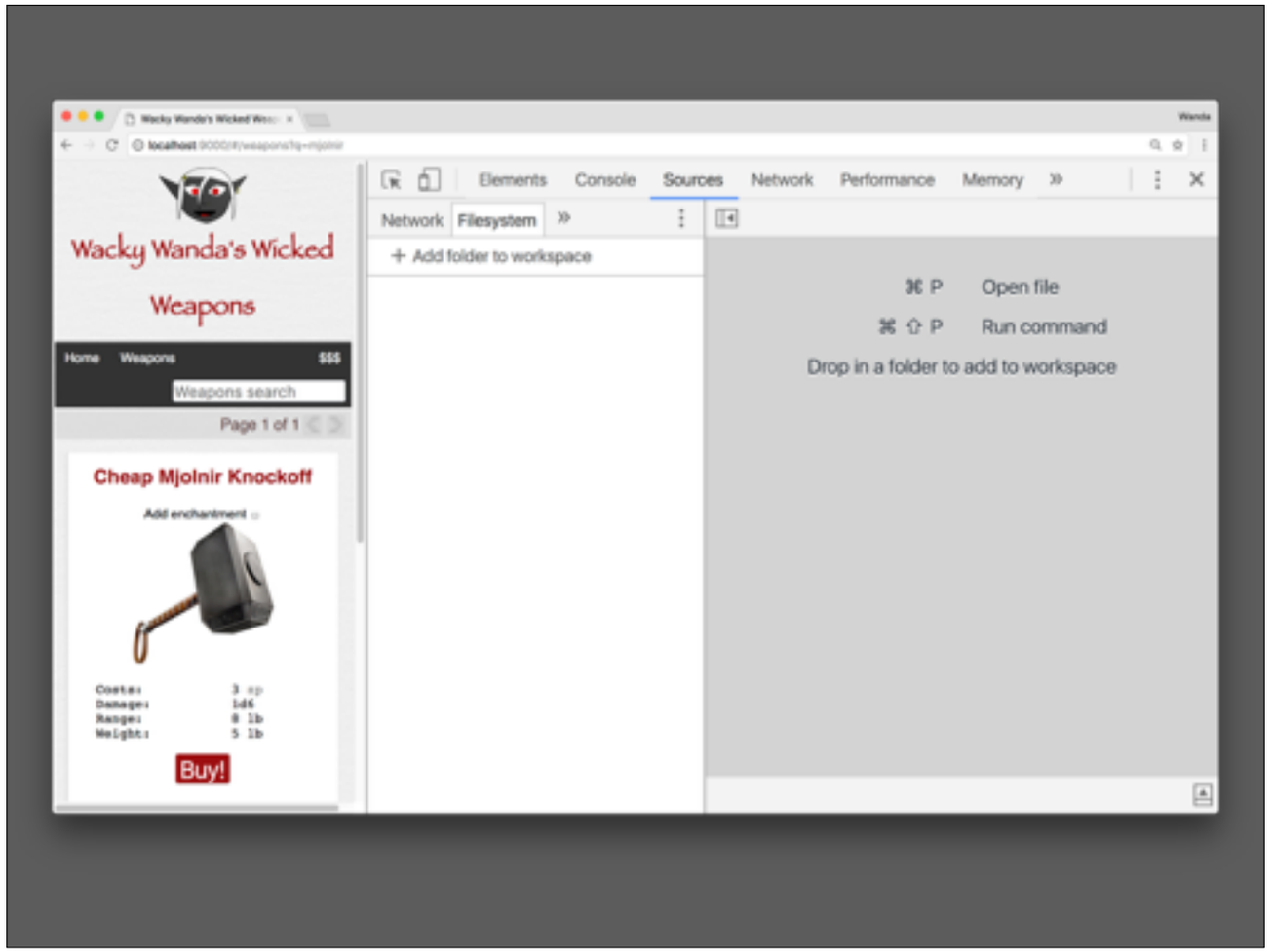


We need a label and a checkbox input. We can use the weapon model's **enchanted** attribute to decided if **"checked"** is applied using a trinary expression. While making the code change a star will appear next to the filename at the top of the page. When we save with CMD + S the original source file will update. This will cause web pack to refresh the page, and there's our new checkbox!

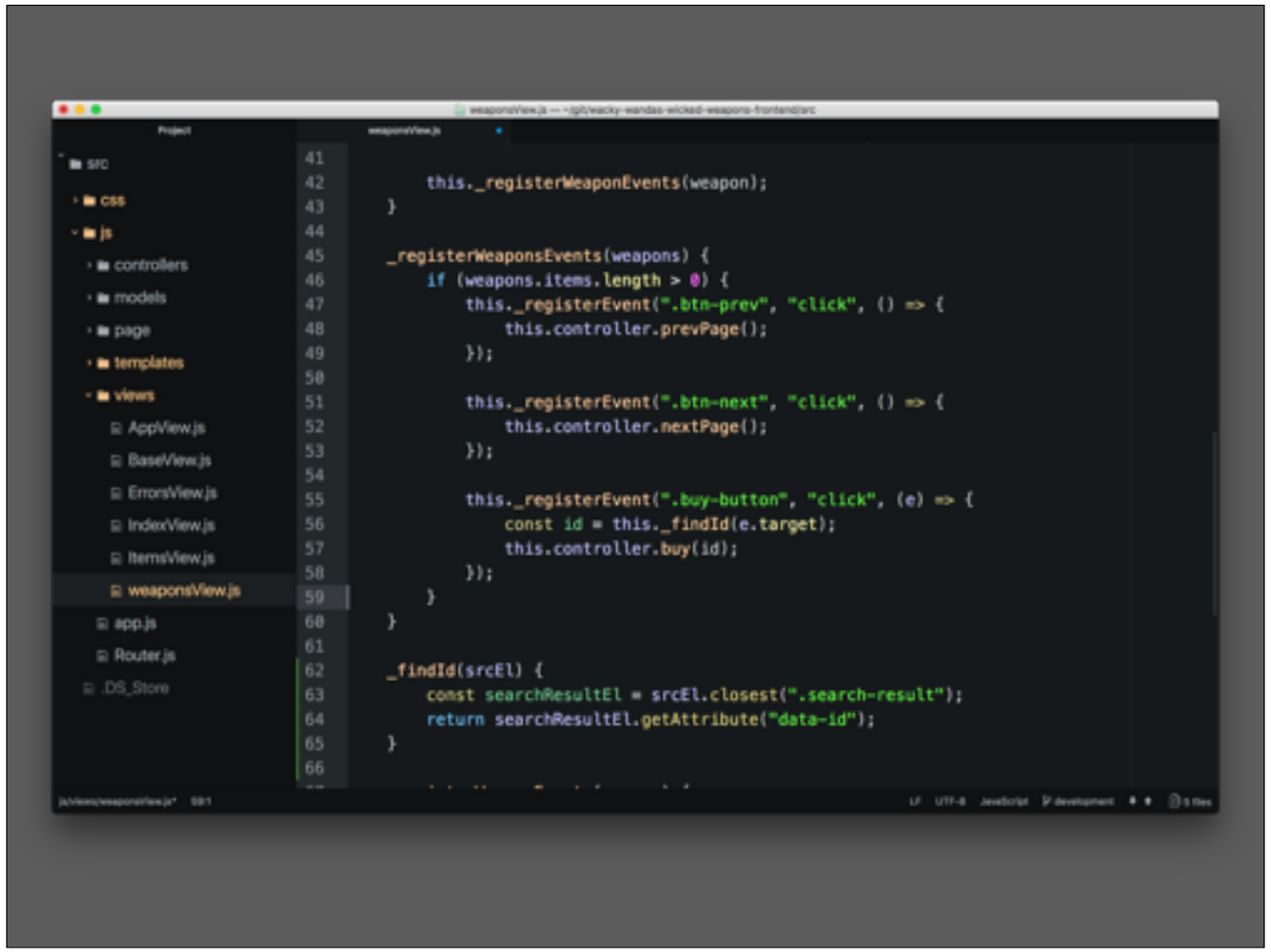
It doesnt' actually do anything yet though. Next we need to register an event from the Weapon View.



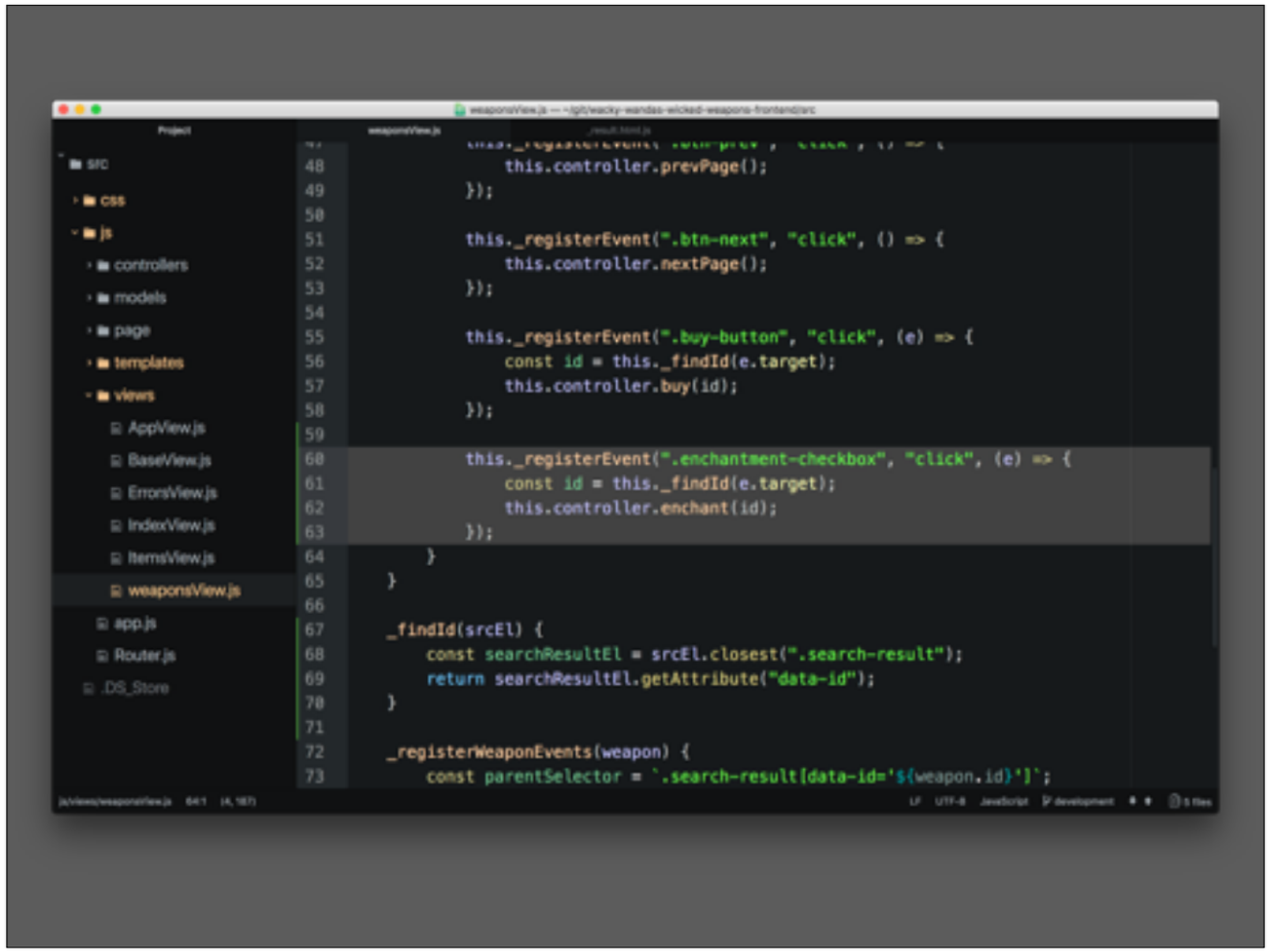
At this point we're going to say goodbye to Workspaces so we can use the debugger reliably.



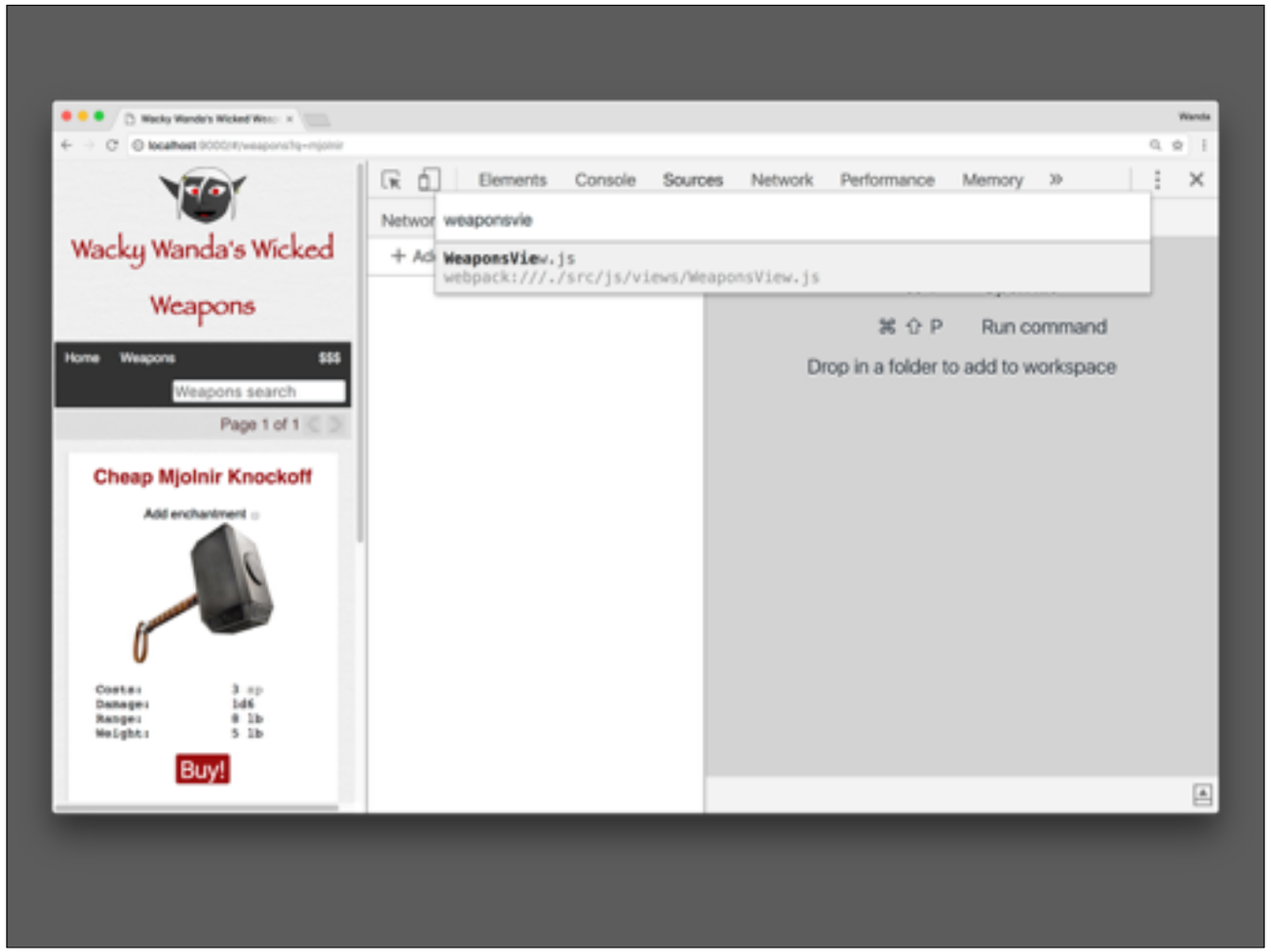
From here on in we'll rely on Atom for code changes instead. We need to go open WeaponsView.



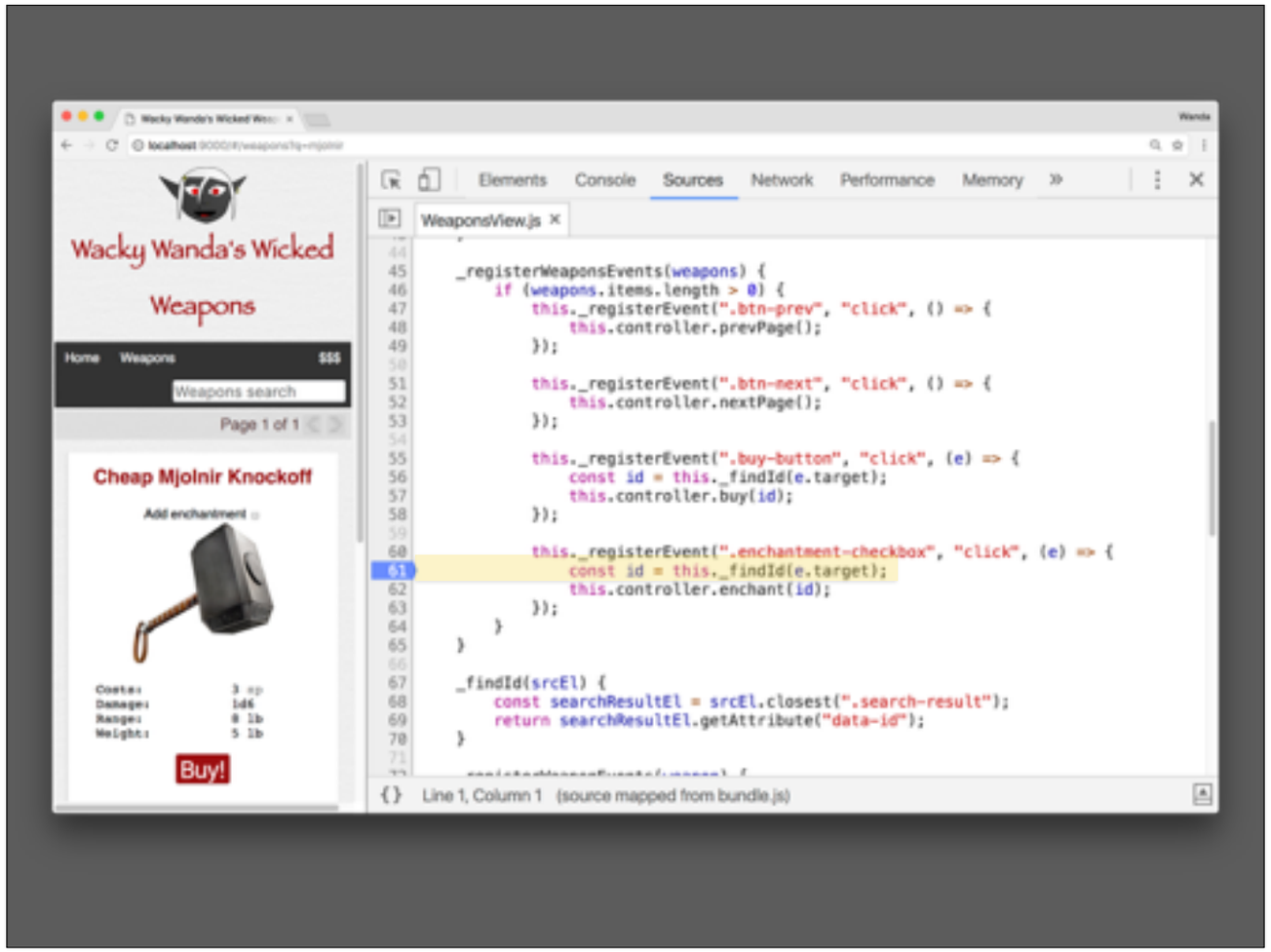
This looks promising, we likely need to add an event to **_registerWeaponsEvents()**. The functionality should be pretty similar to the the code for the Buy button.



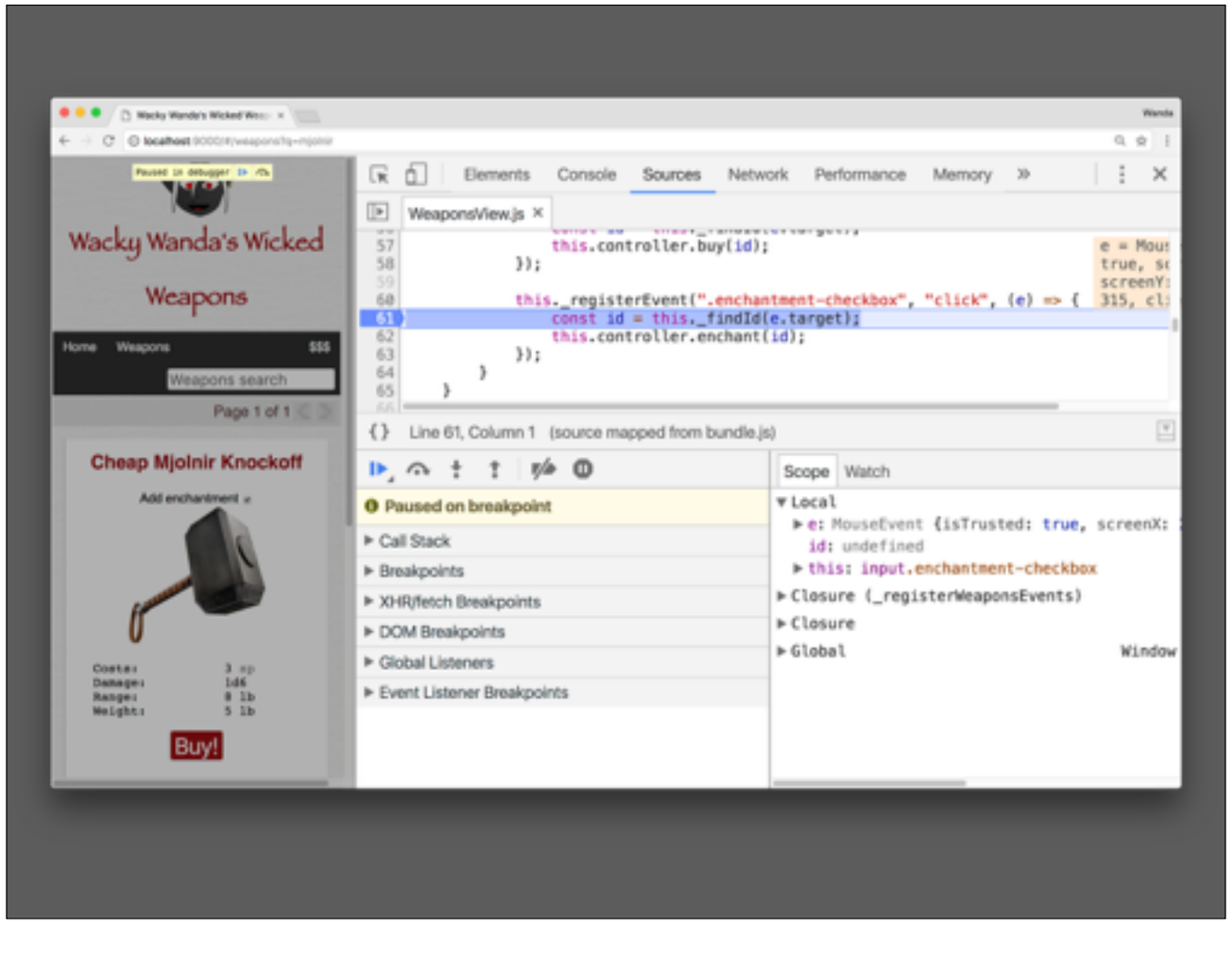
Most of this function is easy to figure out but I can’t remember the exact place to get the checked value. So I’m going to load this code in Chrome and set a breakpoint.



I can use CMD+P to open weapsonView.js in chrome.

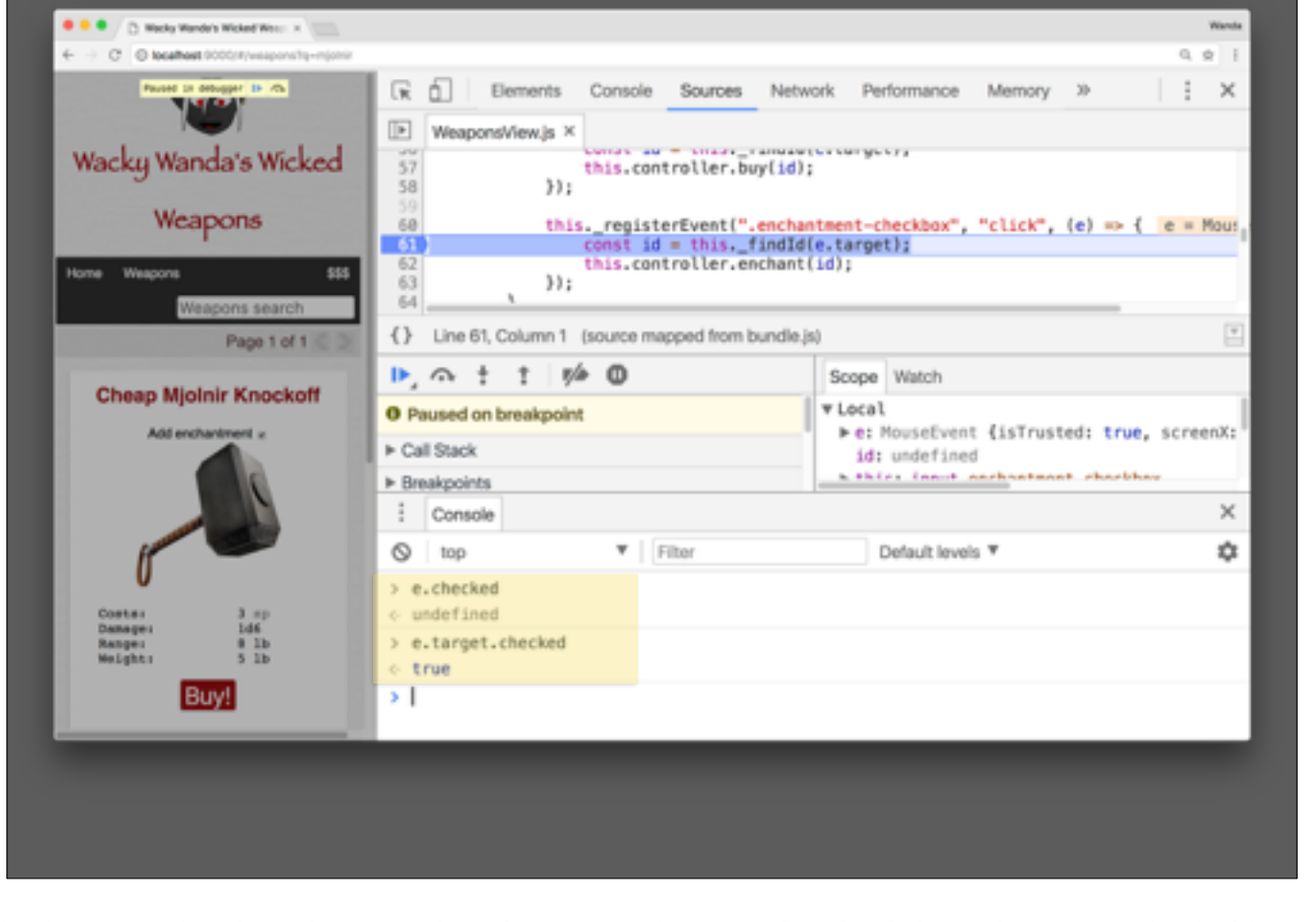


I'll place a breakpoint by clicking the margin at the beginning of the function.

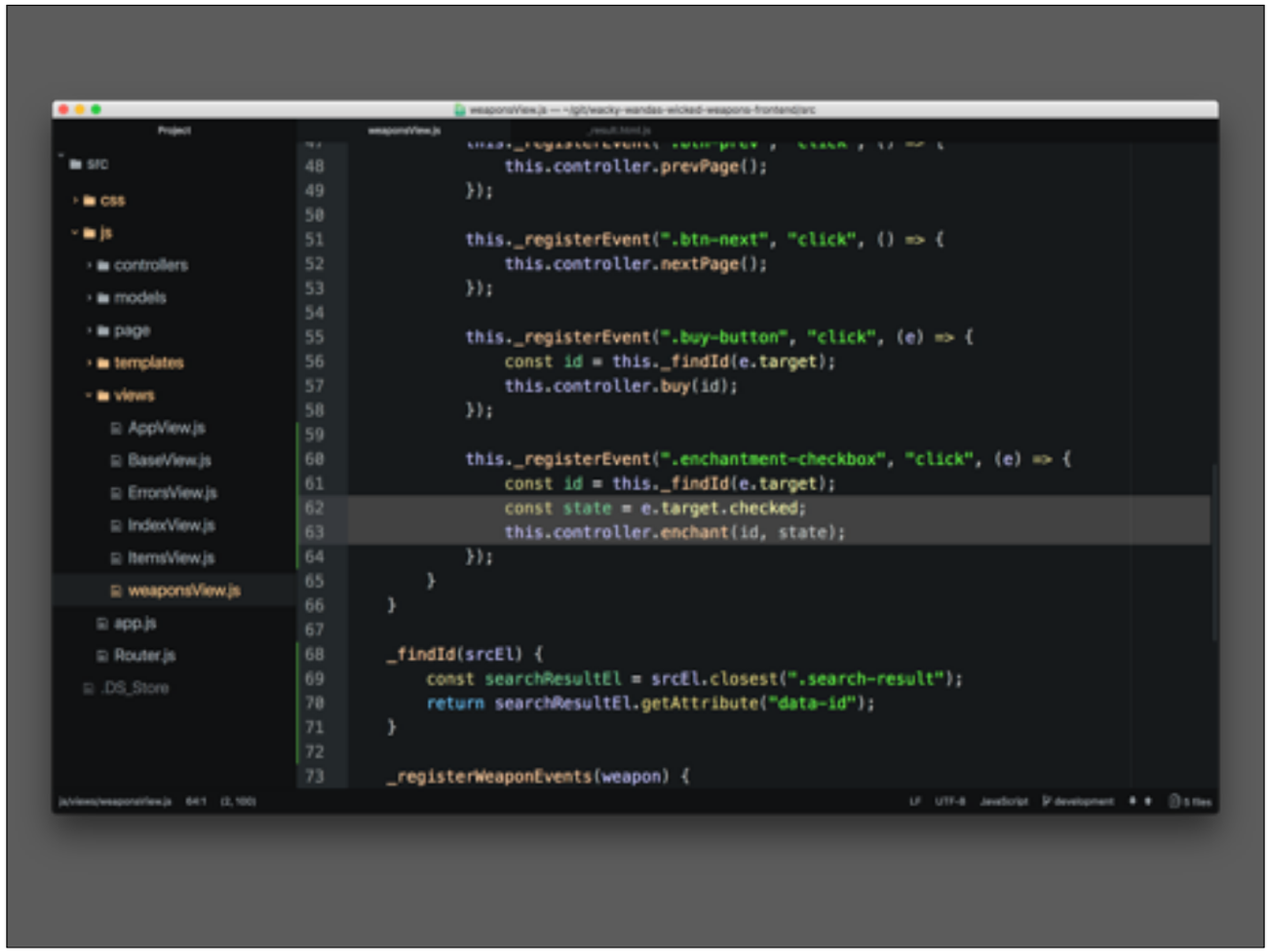


When I click on the checkbox we hit the breakpoint. Which means we've registered the event correctly.

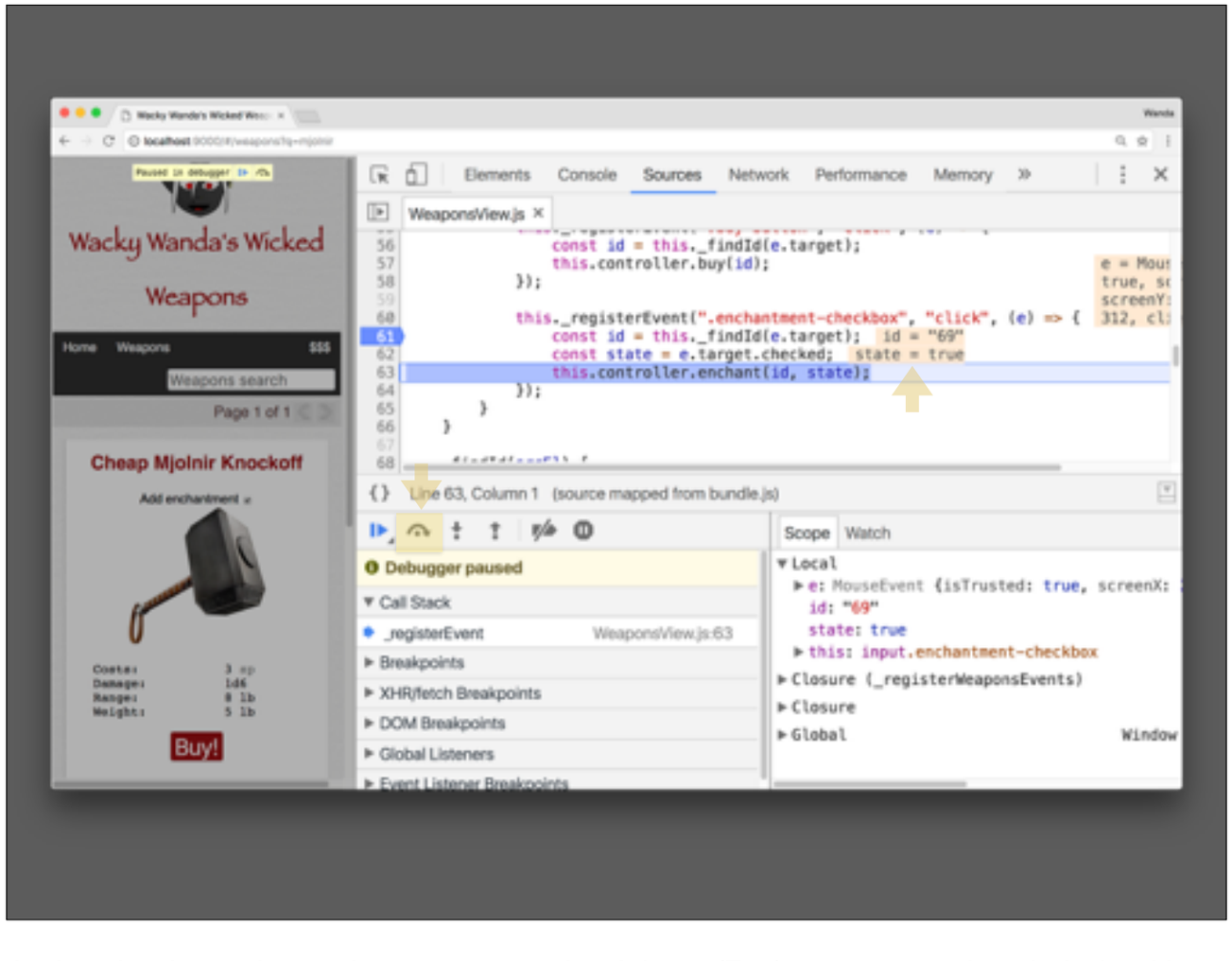
We still need to to figure out where the checked state is though.



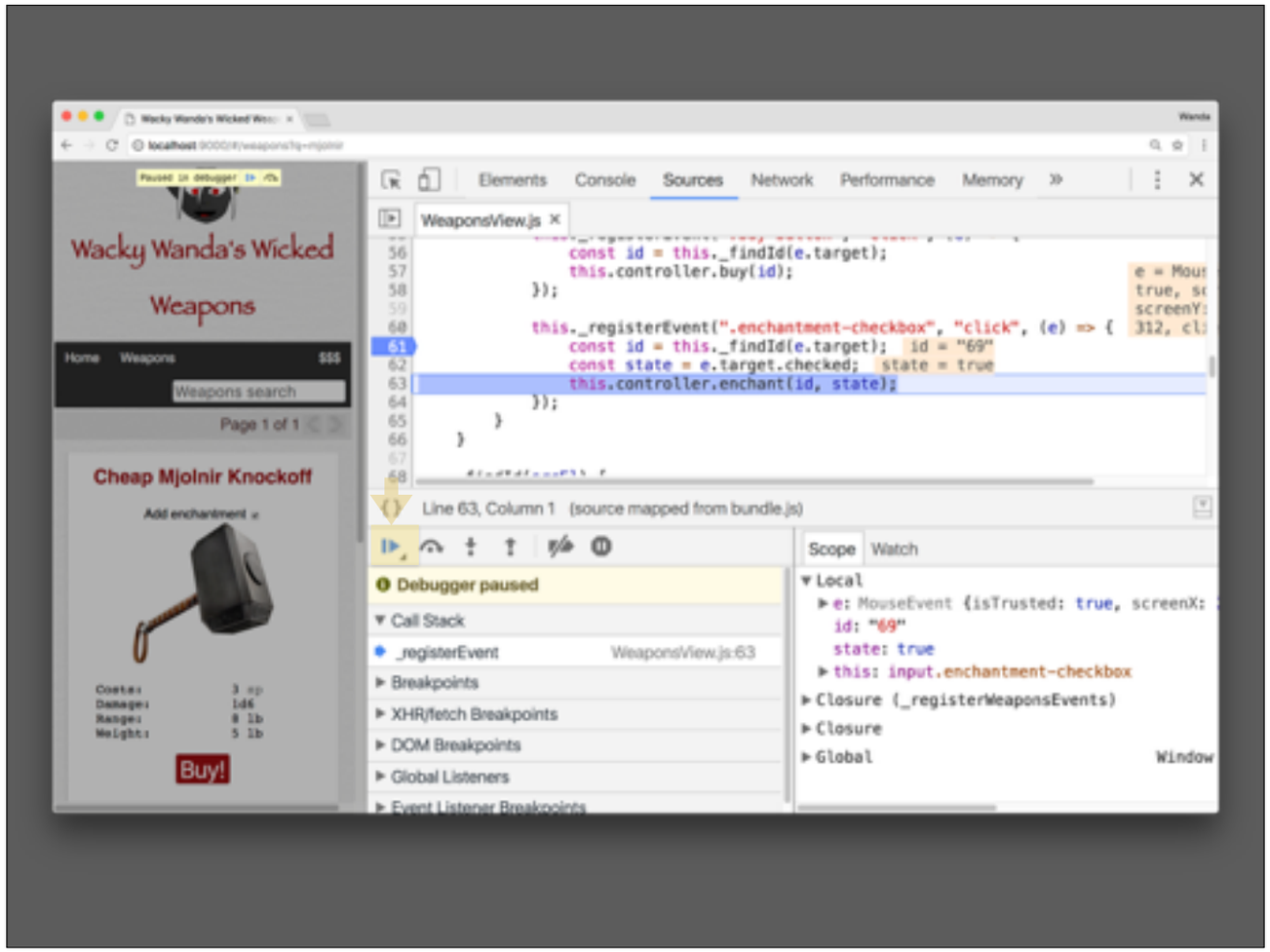
I know the checked state is somewhere in the event but I can’t remember the exact place. **e.checked** doesn’t seem to be it. I have better luck with **e.target.checked**.



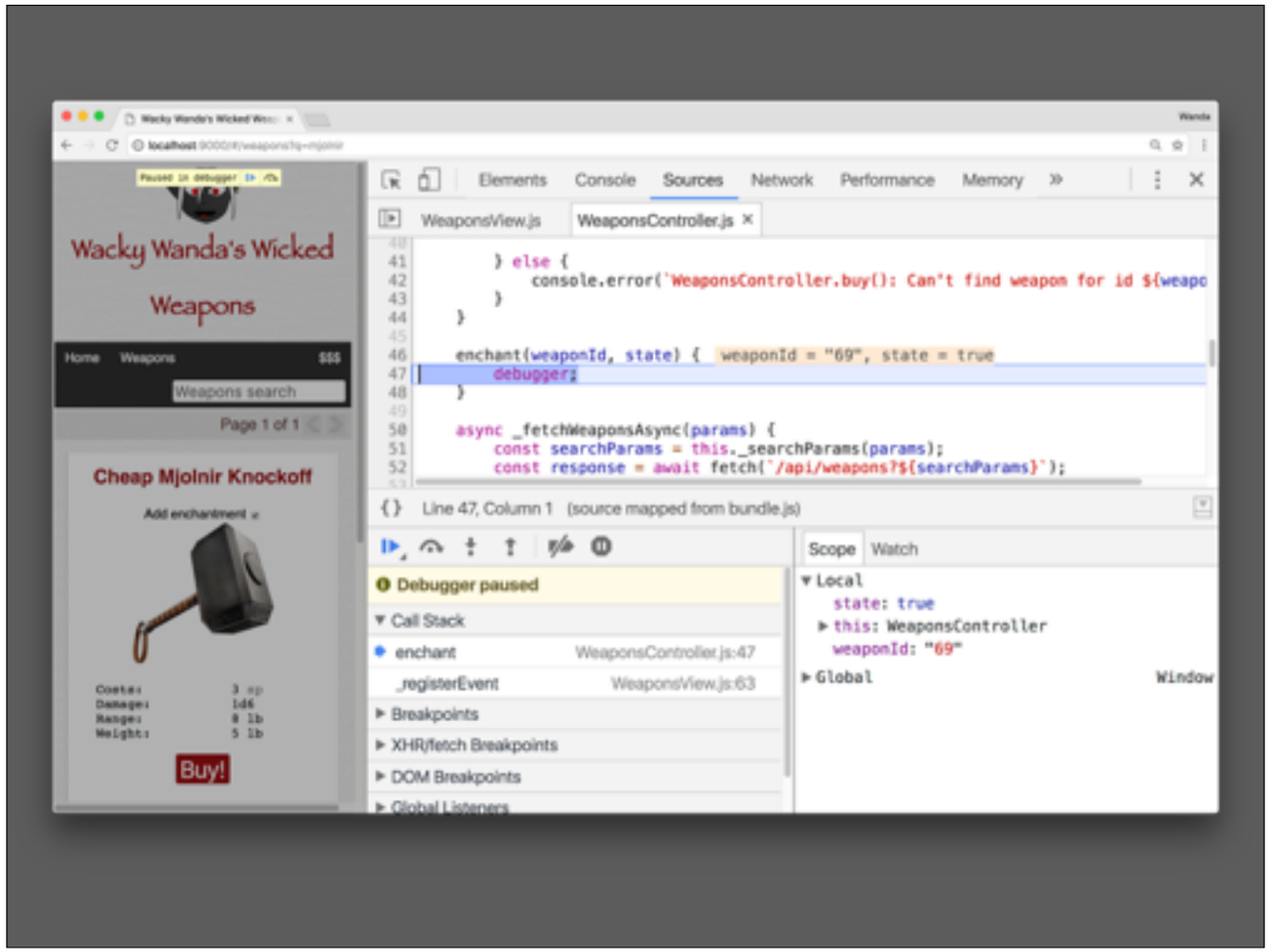
If we add a state variable that uses the **e.target.checked** state and feeds it to the controller we should have everything we need.



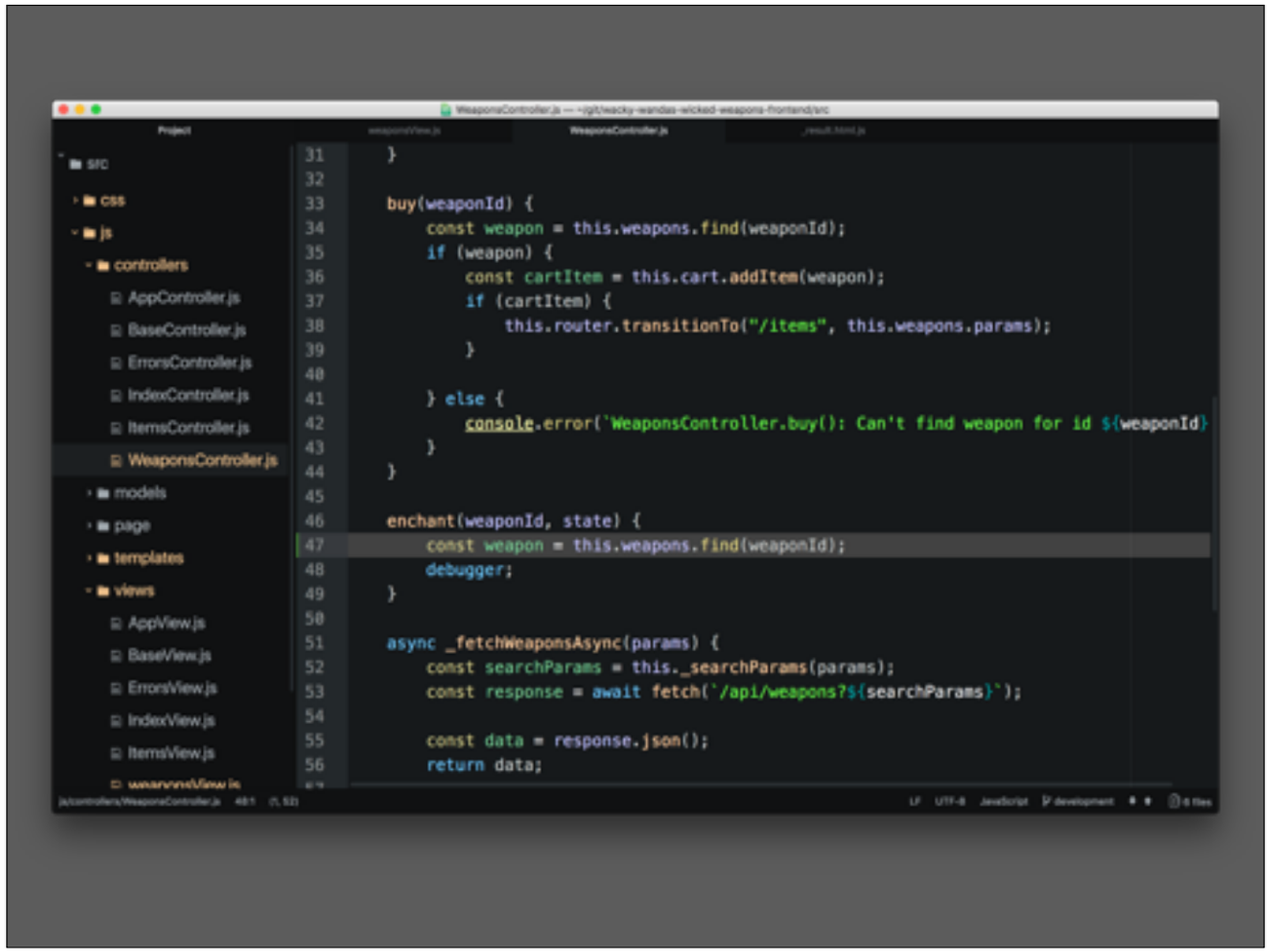
Testing again from Chrome. After hitting the breakpoint and stepping over a couple of times (F10) we can see that do indeed have a valid **state** variable.



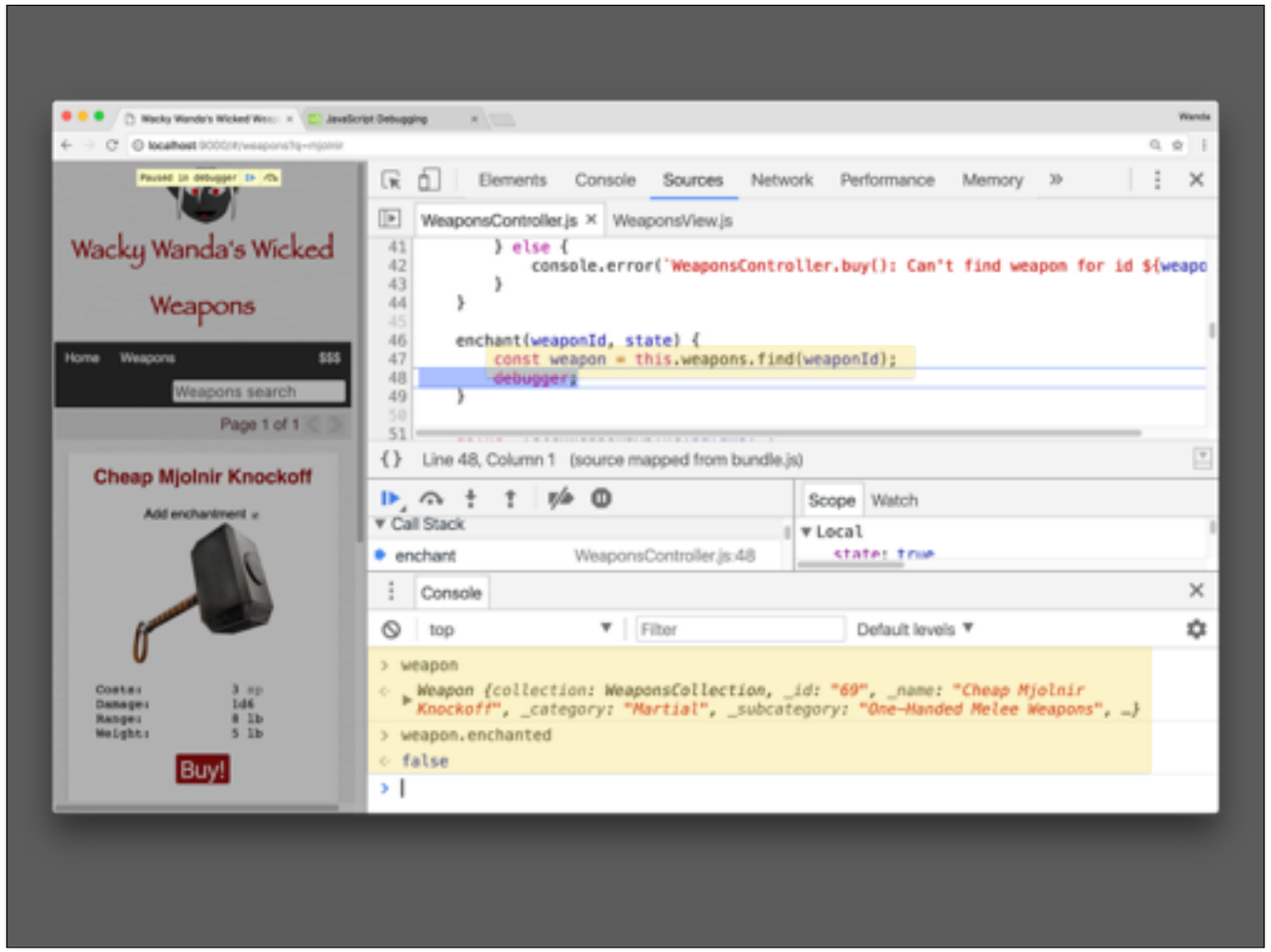
We can press F8 to continue but we haven't written the controller action for "enchant" yet so we'll probably hit an exception.



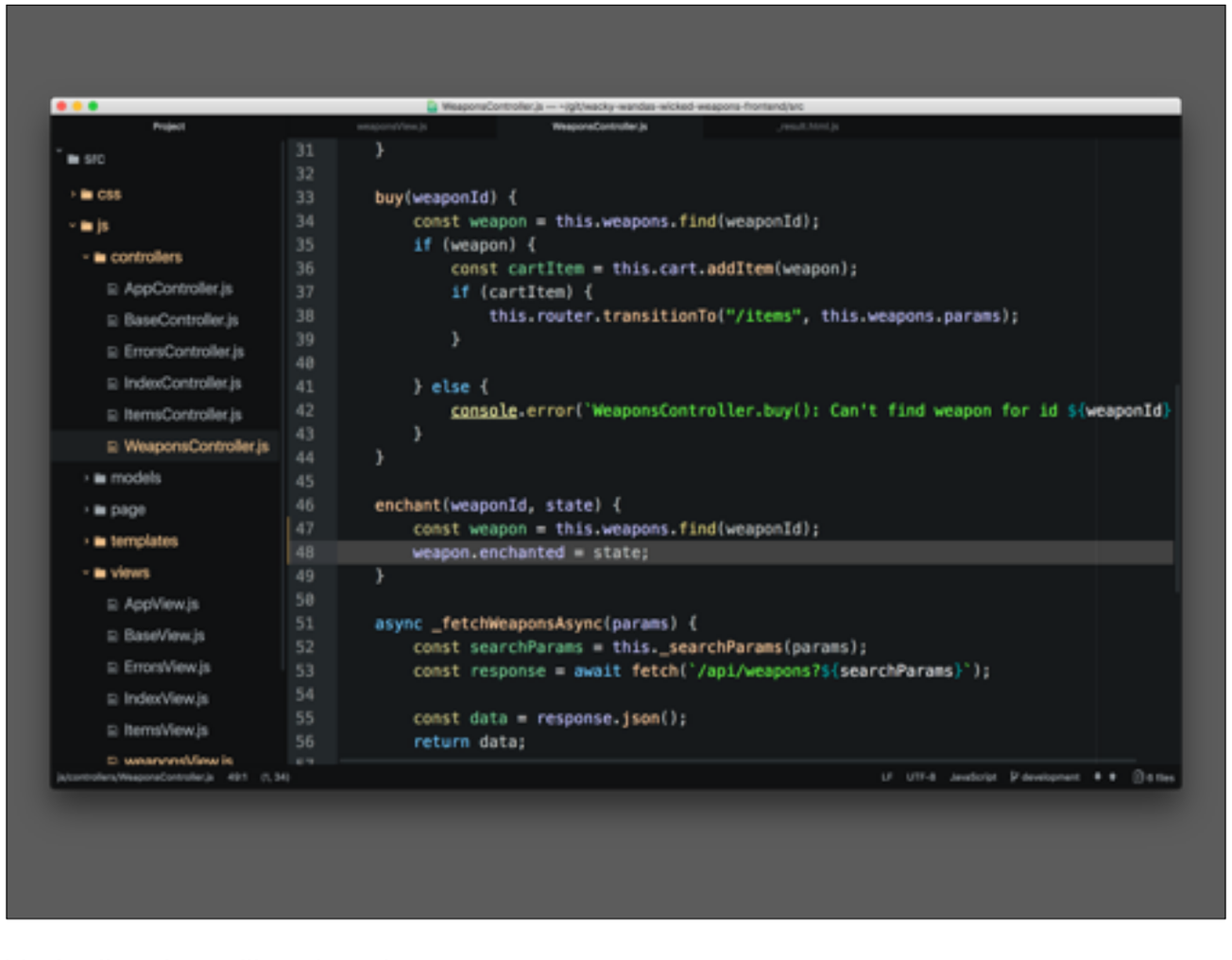
… or not! Looks like Jaspar actually got as far starting this controller action but left the keyword **debugger** here. Any time the word debugger is executed while the Dev Tools are open it is interpreted as an actually breakpoint. This works in all modern browsers. This is very handy when you're working outside of Chrome but want to open the code you're working on in Chrome.



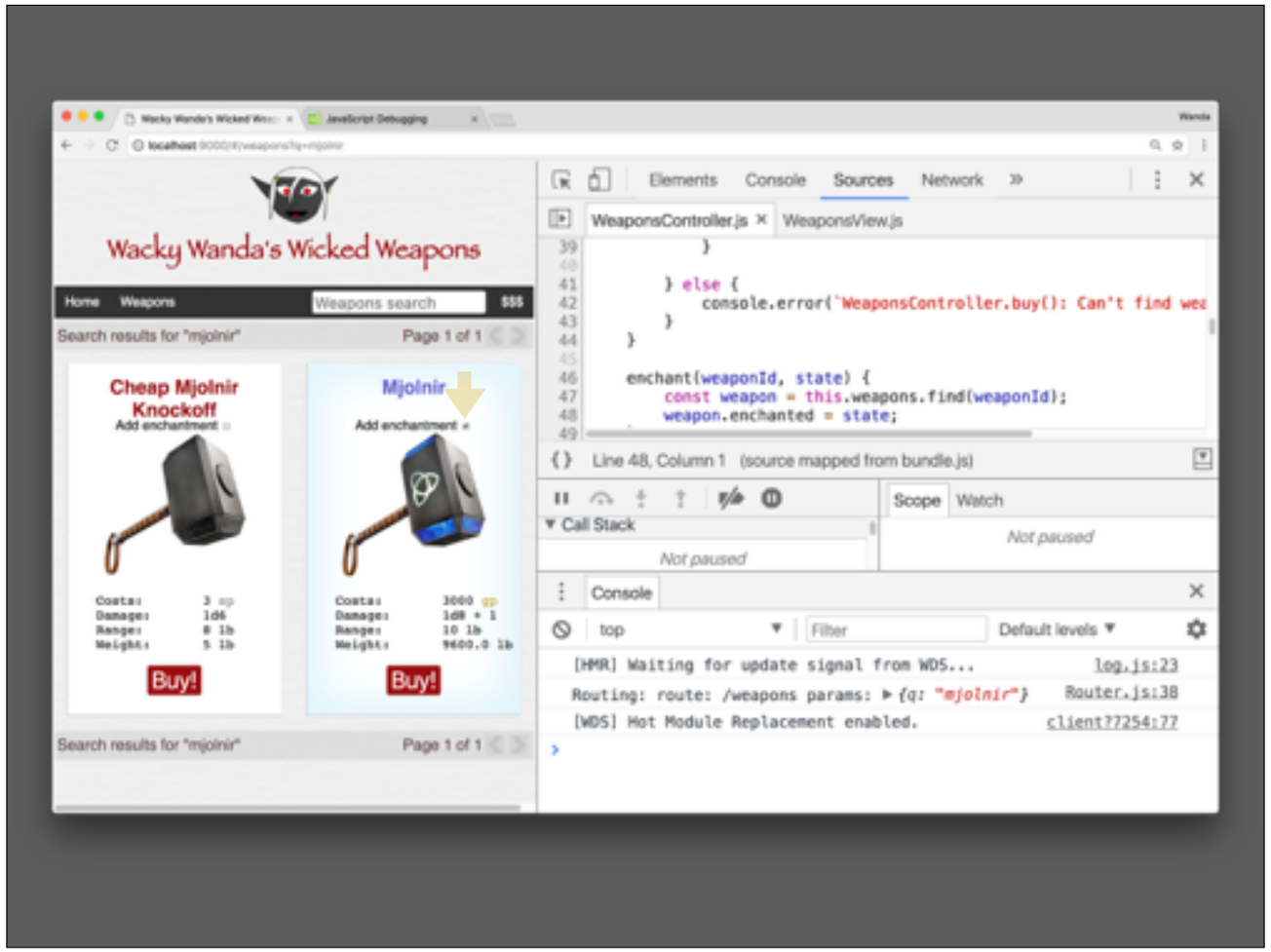
The enchant action is again behaves similar to the buy action. I should be able to steal that line for looking up the individual weapon.



Now when we hit the debugger we have a weapon at our disposal. This is the same weapon model we worked with earlier. I confirm this by querying the **enchanted** property



Let’s replace that debugger statement with the line that will update the weapon.



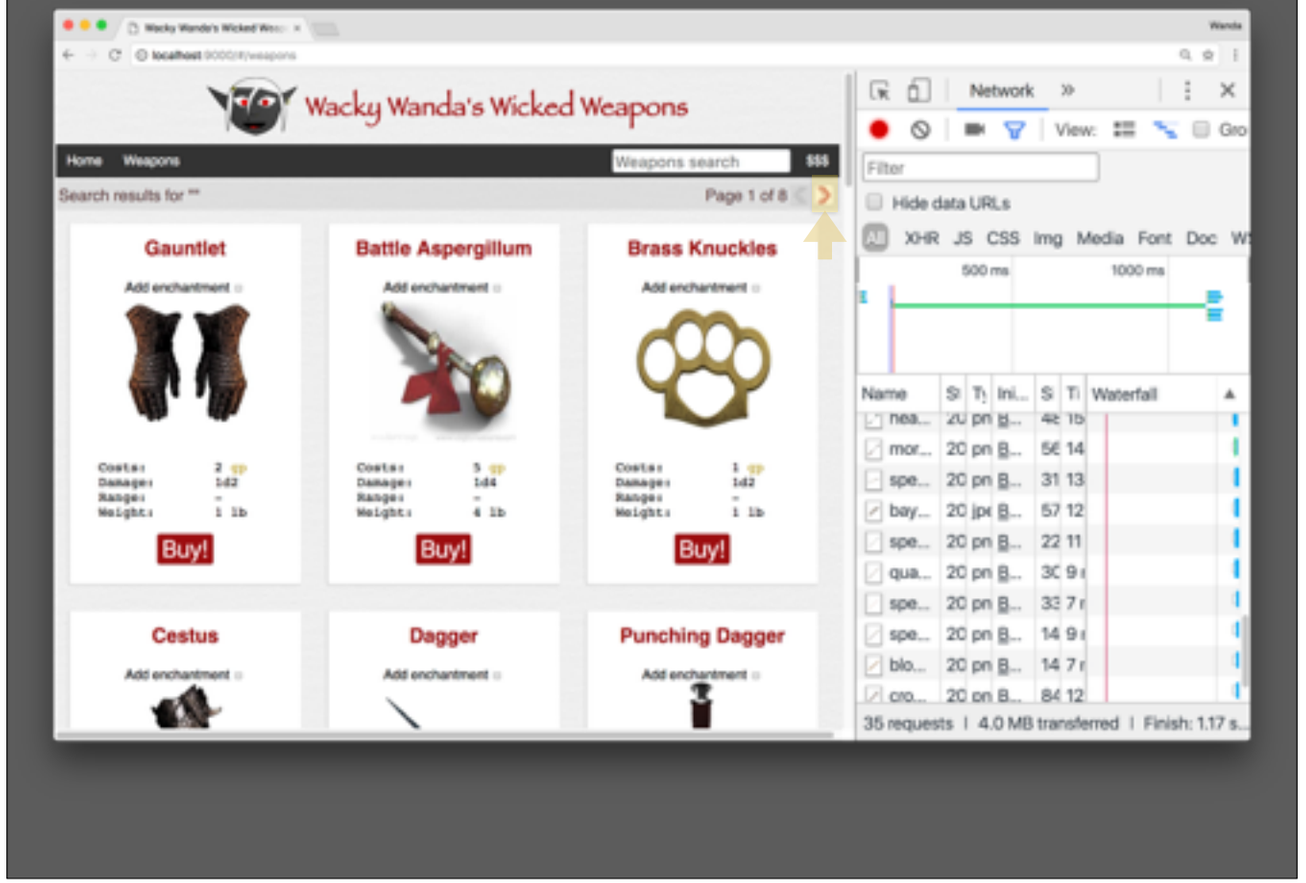
And if we test it…

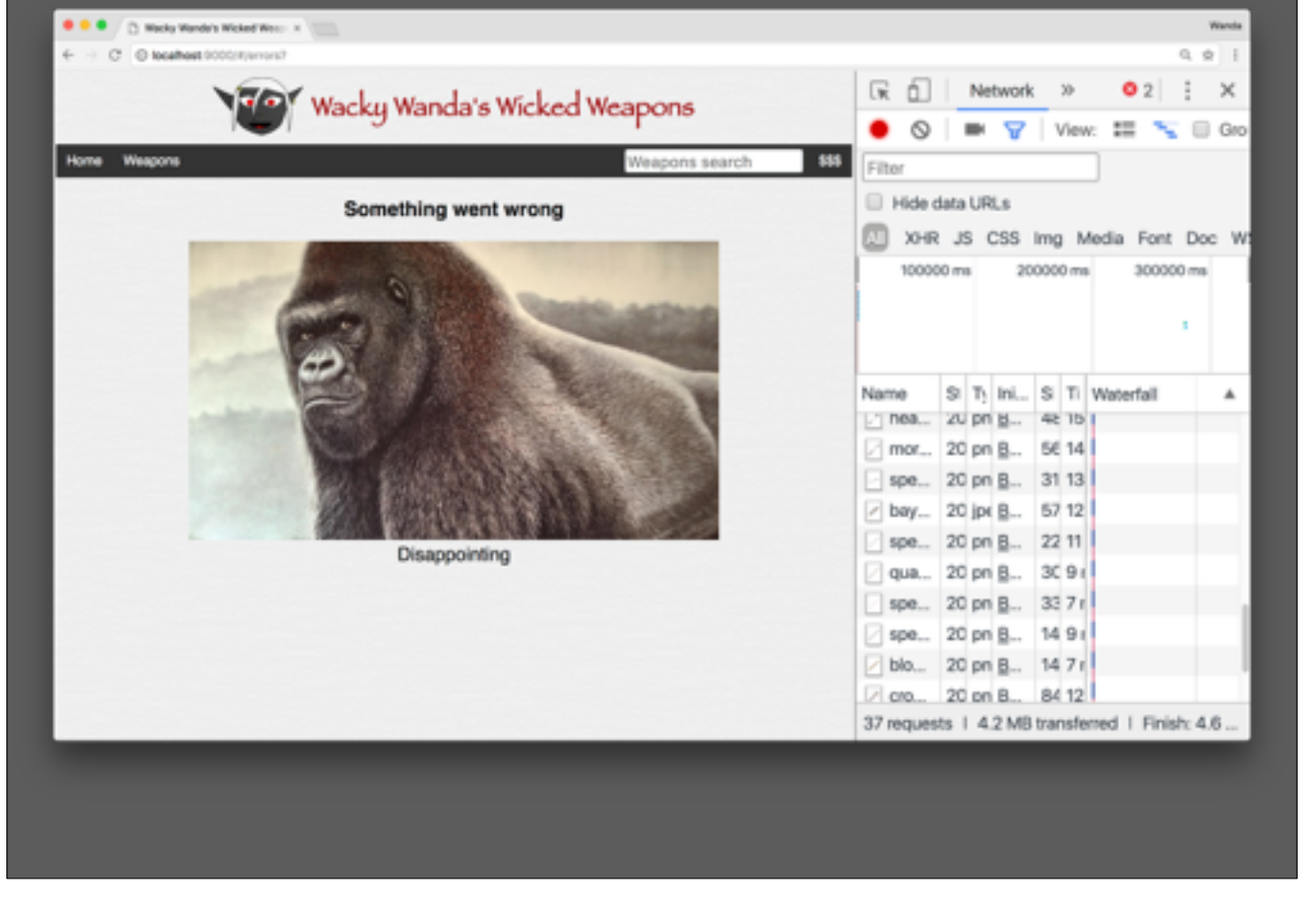
It works!

Network Panel

Jacked Ajax

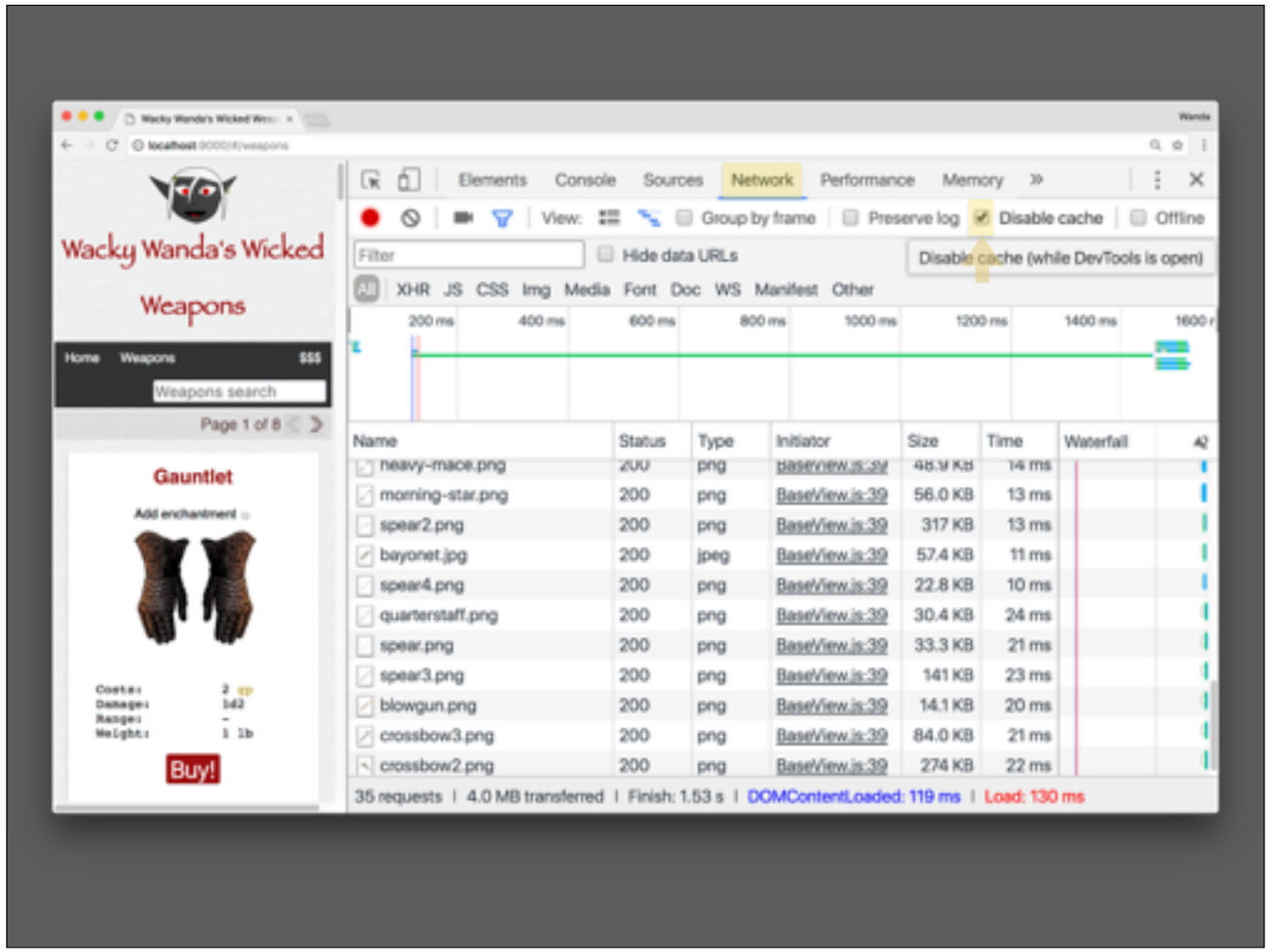
We have one more problem to resolve, fixing the paging bug.


Wacky Wanda's Wicked Weapons
Home
Weapons
Weapons search
$$$
Search results for ""
Page 1 of 8
Gauntlet
Add enchantment
Cost: 2 gp
Damage: 1d2
Range: -
Weight: 1 lb
Buy!
Battle Aspergillum
Add enchantment
Cost: 5 gp
Damage: 1d4
Range: -
Weight: 4 lb
Buy!
Brass Knuckles
Add enchantment
Cost: 1 gp
Damage: 1d2
Range: -
Weight: 1 lb
Buy!
Cestus
Add enchantment
Dagger
Add enchantment
Punching Dagger
Add enchantment
Network
Filter
Hide data URLs
XHR JS CSS Img Media Font Doc
500 ms
1000 ms
Name
Waterfall
35 requests | 4.0 MB transferred | Finish: 1.17 s

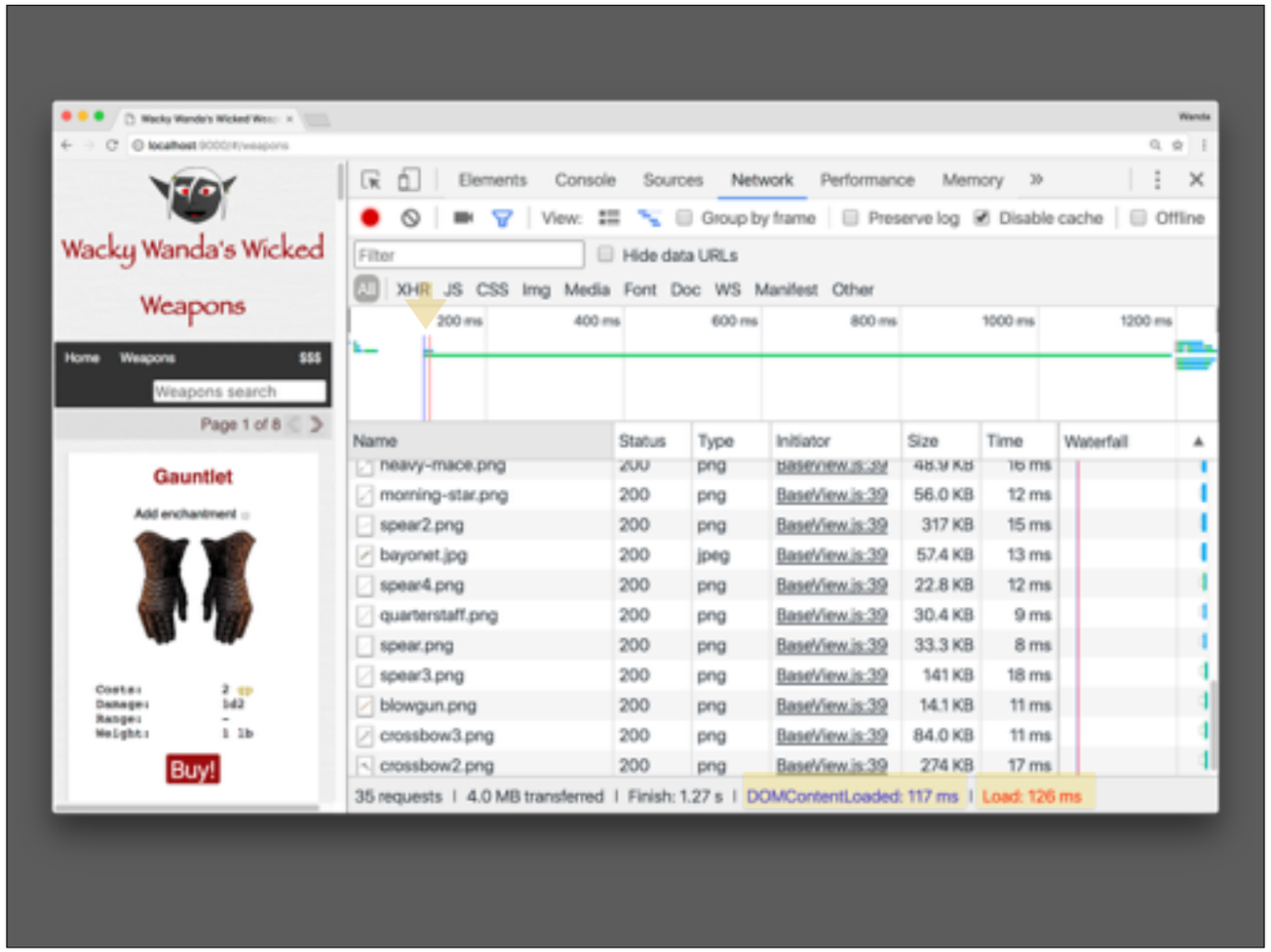


… this happens.

It’s probably an ajax problem so we’ll go fix it from the Network Panel. We’ll also take a look at Page Load Performance while we’re there.



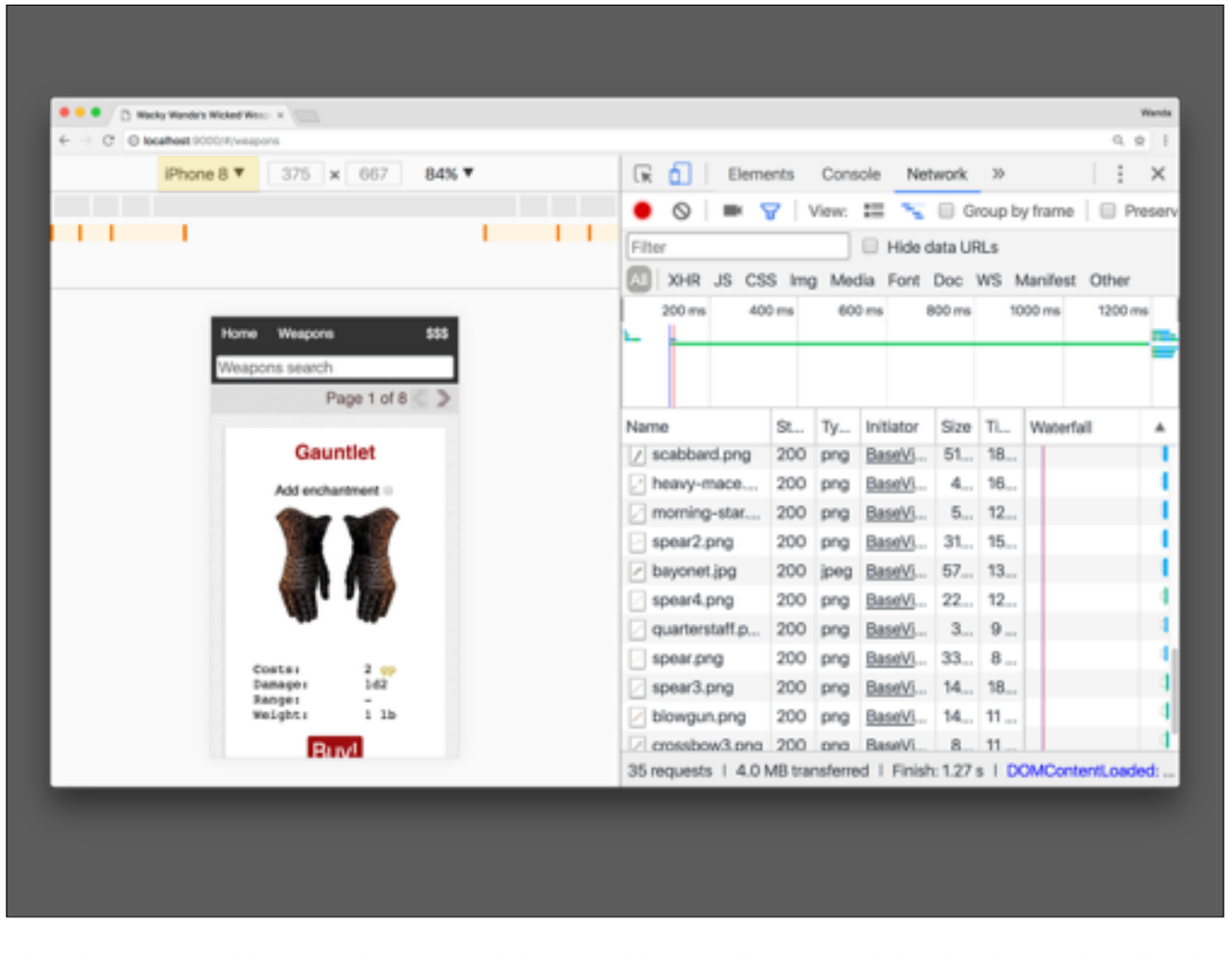
Welcome to the Network Panel. Before we get started, notice that I have **Disable Cache** turned on. This should always be turned on. It only take effect when the Dev Tools are open. If it's not set the browser will cache the page causing changes you make not to show up straight away.



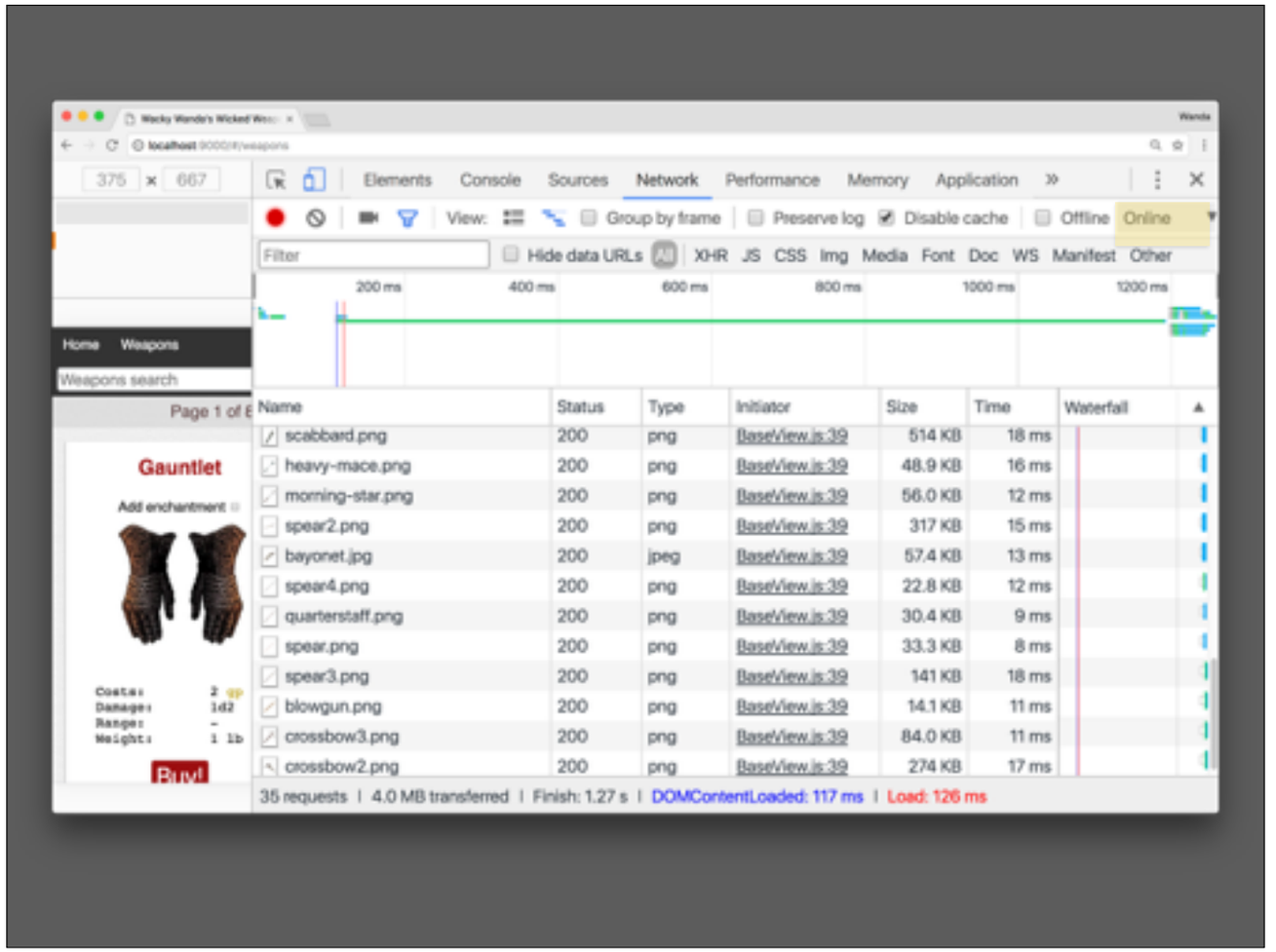
The waterfall view is useful for analyzing Page Load Performance. If we look at the status bar there are 2 very important metrics on the page. There's the **DomContentLoaded** in blue and the **Load** marked in red. They are also show in the timeline overview above as the blue line and the red line.

The DomContentLoaded time how long it takes for the document to load. At this point assets such as css, images and subframes may still be downloading, but the page events are now starting to happen. The full Load is for when those dependencies have also downloaded.

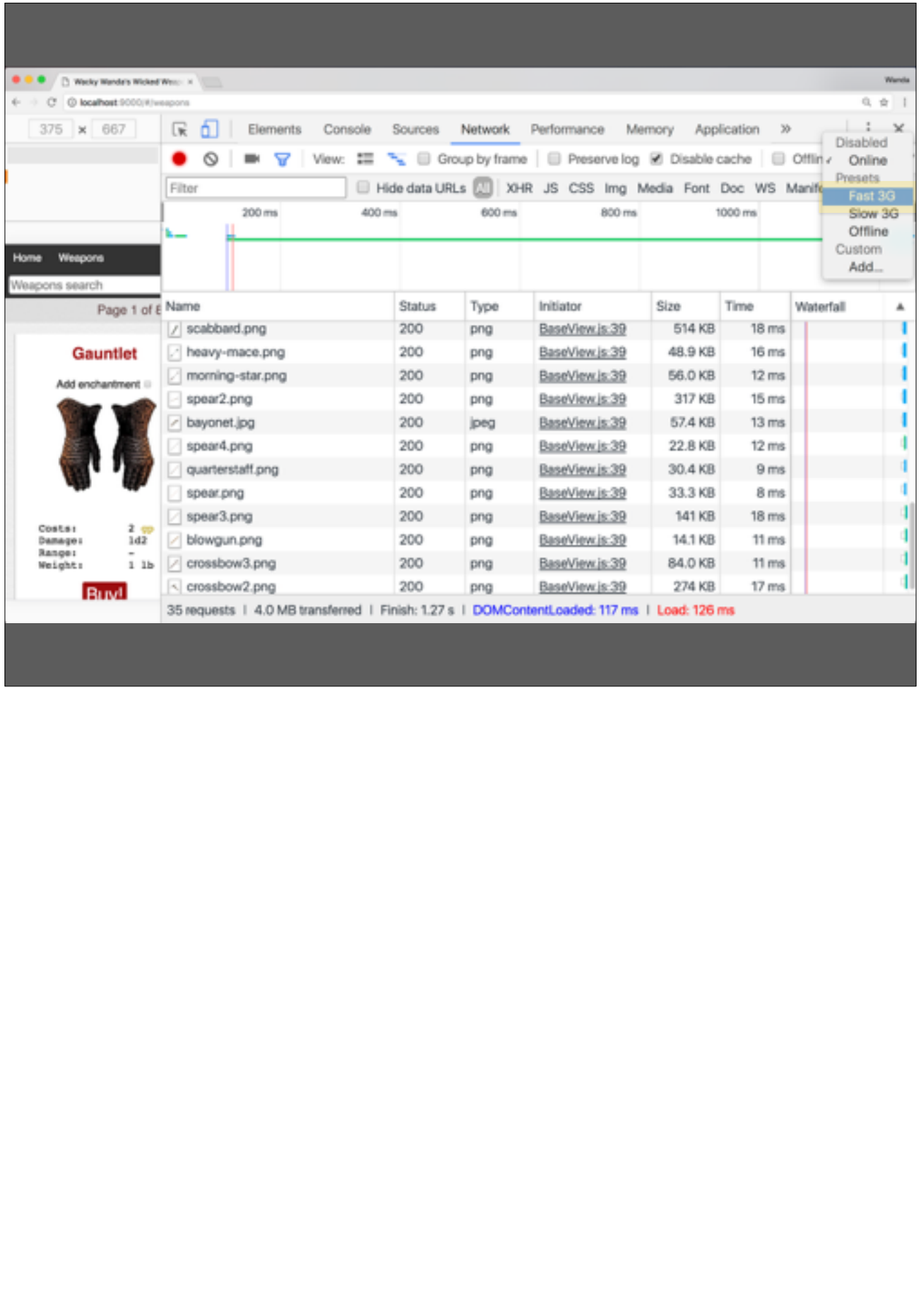
Which one is more important? It depends on your website. The thing that really matters here is how long does your site appear to be loading? If the time is less than a second the visitor is hardly going to notice. If it takes longer than 5 seconds you are definitely going to start losing visitors. Our site apparently was fully ready in 126ms! Not bad!

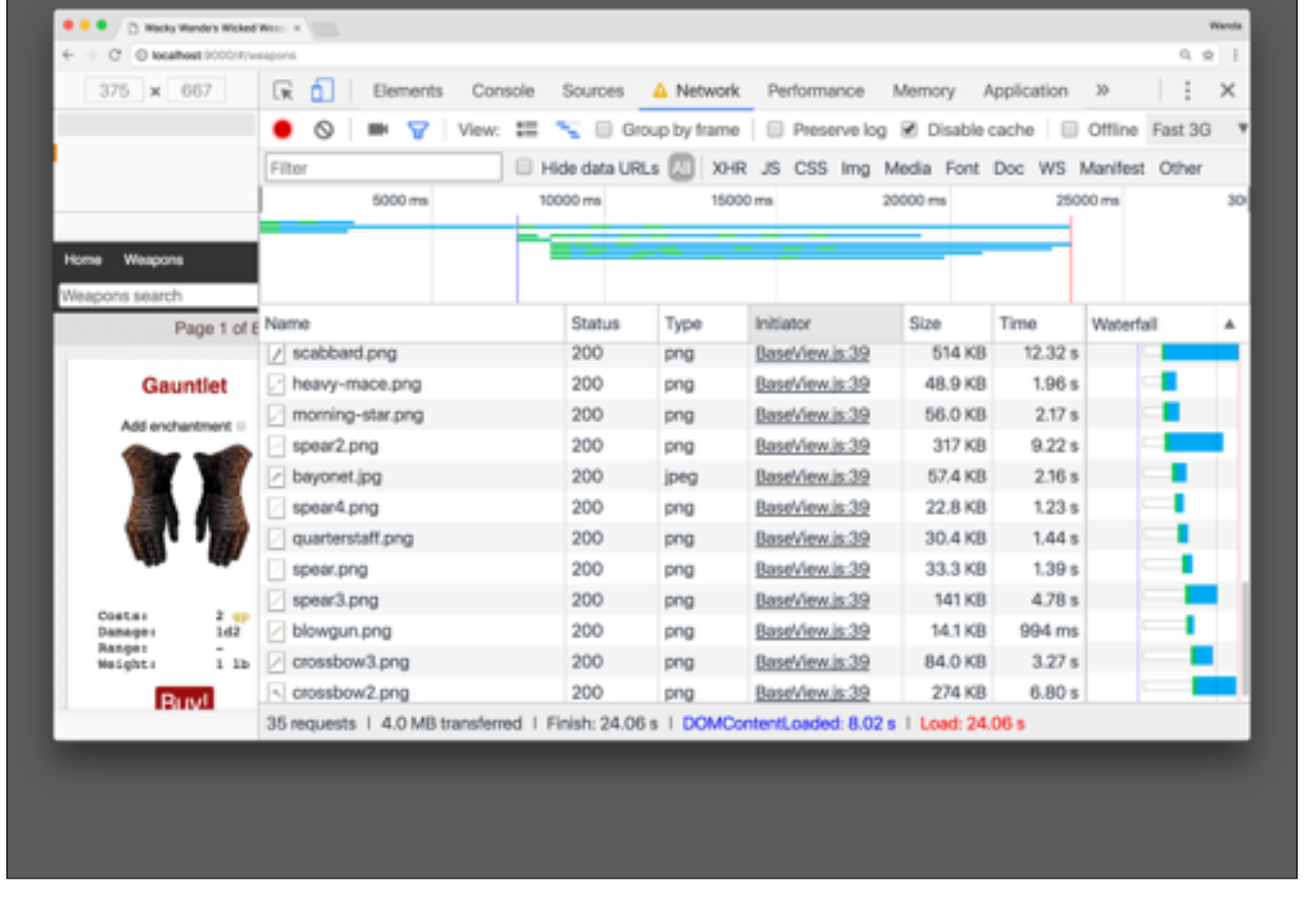


… except that wasn't a very realistic test. Firstly we should be testing for mobile first. It's usually the mobile site that will suffer the worst performance problems. I'll go with iPhone8.



Secondly we should turn on network throttling to mimic a mobile network more realistically. I'll throttle to **Fast 3G**.


Elements
Console
Sources
Network
Performance
Memory
Application
Disabled
Online
Presets
Fast 3G
Slow 3G
Offline
Custom
Add...
Home
Weapons
Weapons search
Gauntlet
Add enchantment
Costs: 2
Damage: 1d2
Range: -
Weight: 1 lb
Buy!
Name
Status
Type
Initiator
Size
Time
Waterfall
scabbard.png 200 png BaseView.js:39 514 KB 18 ms
heavy-mace.png 200 png BaseView.js:39 48.9 KB 16 ms
morning-star.png 200 png BaseView.js:39 56.0 KB 12 ms
spear2.png 200 png BaseView.js:39 317 KB 15 ms
bayonet.jpg 200 jpeg BaseView.js:39 57.4 KB 13 ms
spear4.png 200 png BaseView.js:39 22.8 KB 12 ms
quarterstaff.png 200 png BaseView.js:39 30.4 KB 9 ms
spear.png 200 png BaseView.js:39 33.3 KB 8 ms
spear3.png 200 png BaseView.js:39 141 KB 18 ms
blowgun.png 200 png BaseView.js:39 14.1 KB 11 ms
crossbow3.png 200 png BaseView.js:39 84.0 KB 11 ms
crossbow2.png 200 png BaseView.js:39 274 KB 17 ms
35 requests | 4.0 MB transferred | Finish: 1.27 s | DOMContentLoaded: 117 ms | Load: 126 ms

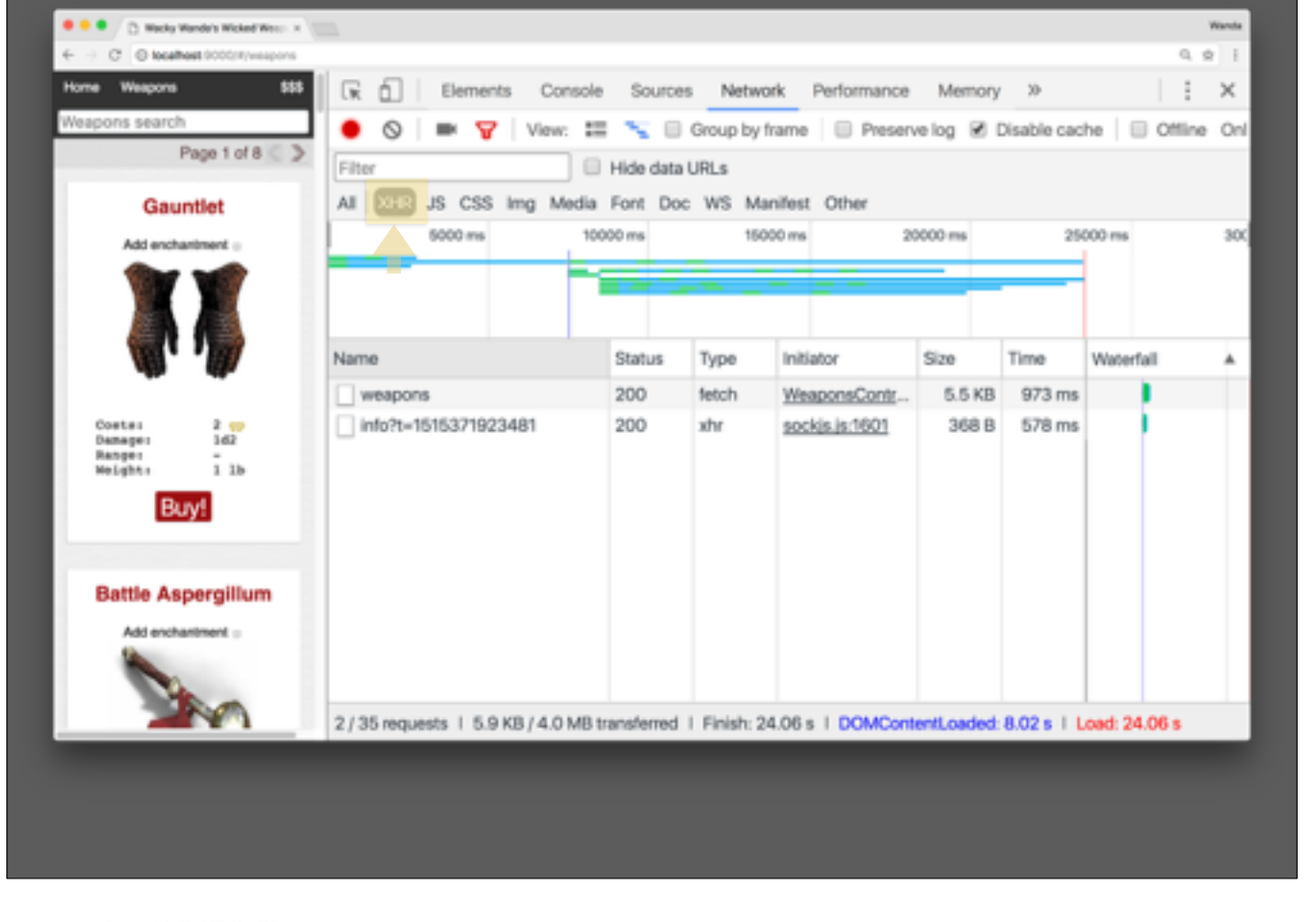


And refresh.

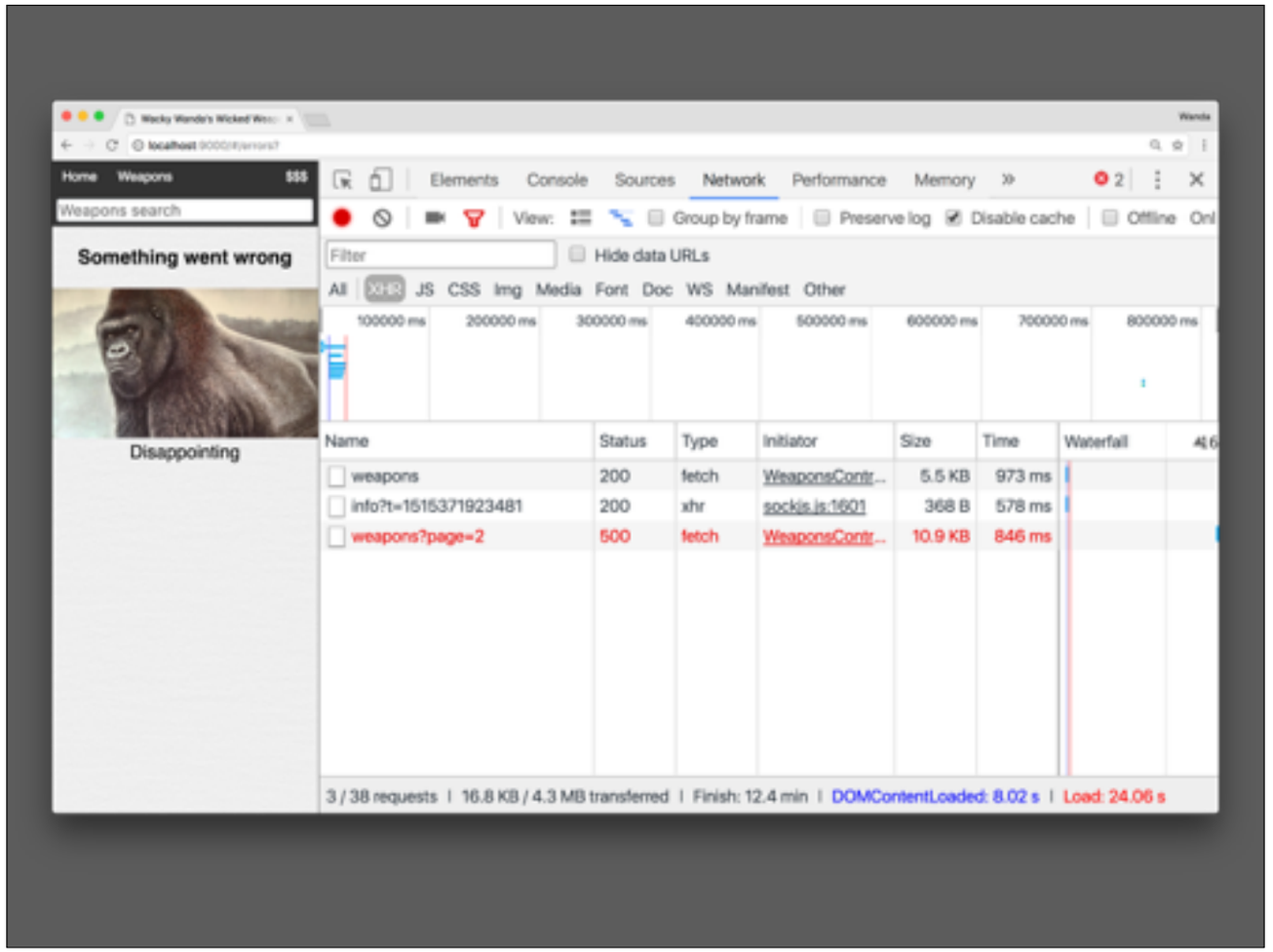
This time not so fast! DOMContentLoaded: 8.02s, Load 24.06s!

The problem is clearly with how we're managing all the images. They're way larger than they need to be and they probably don't all need to load straight away during page load.

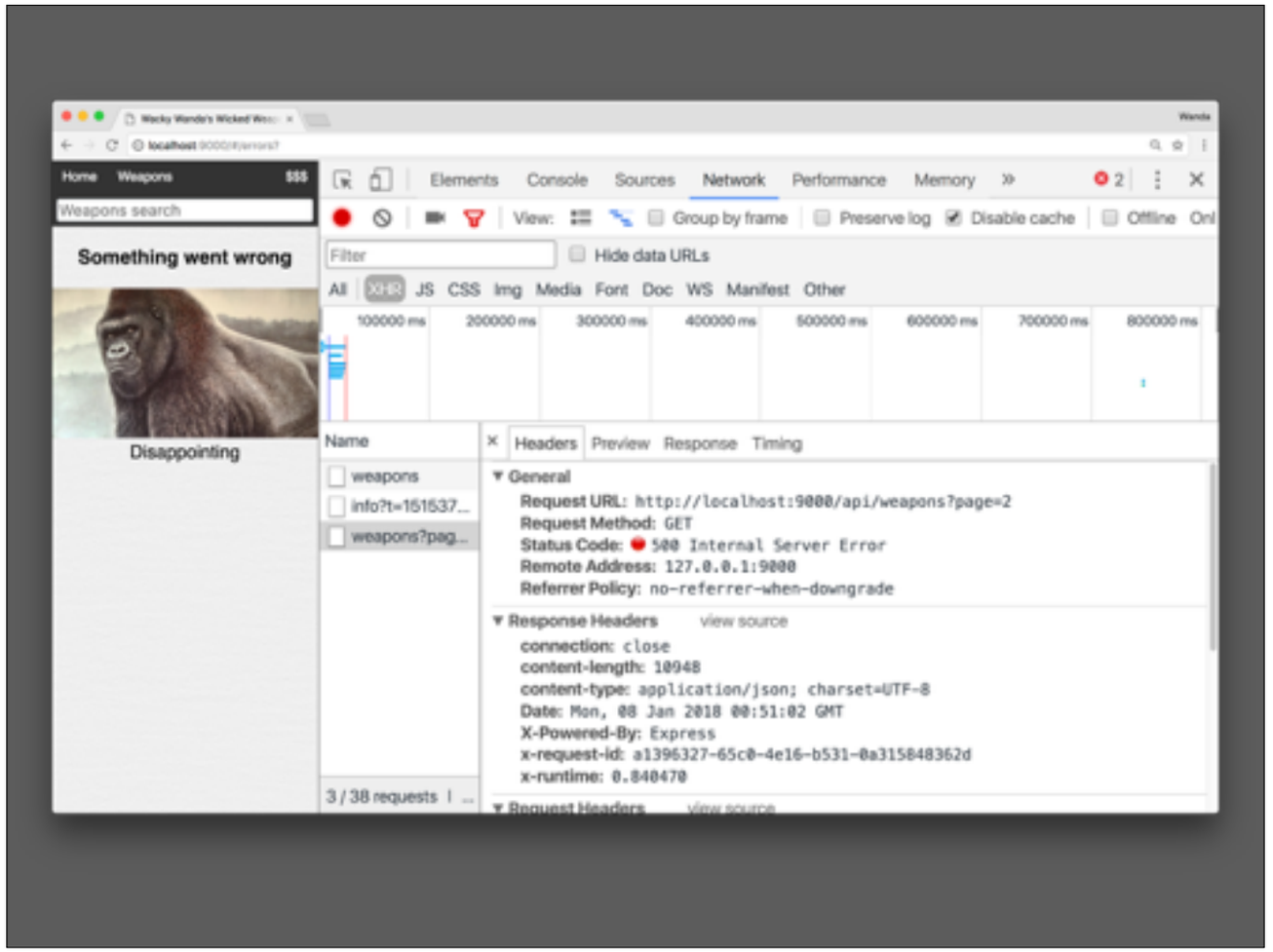
But while this is interesting, we're mostly here for the AJAX problem. So I'm going to switch the throttling and device toolbar back off.



As we're focussing on ajax, we should turn on the "XHR" filter.



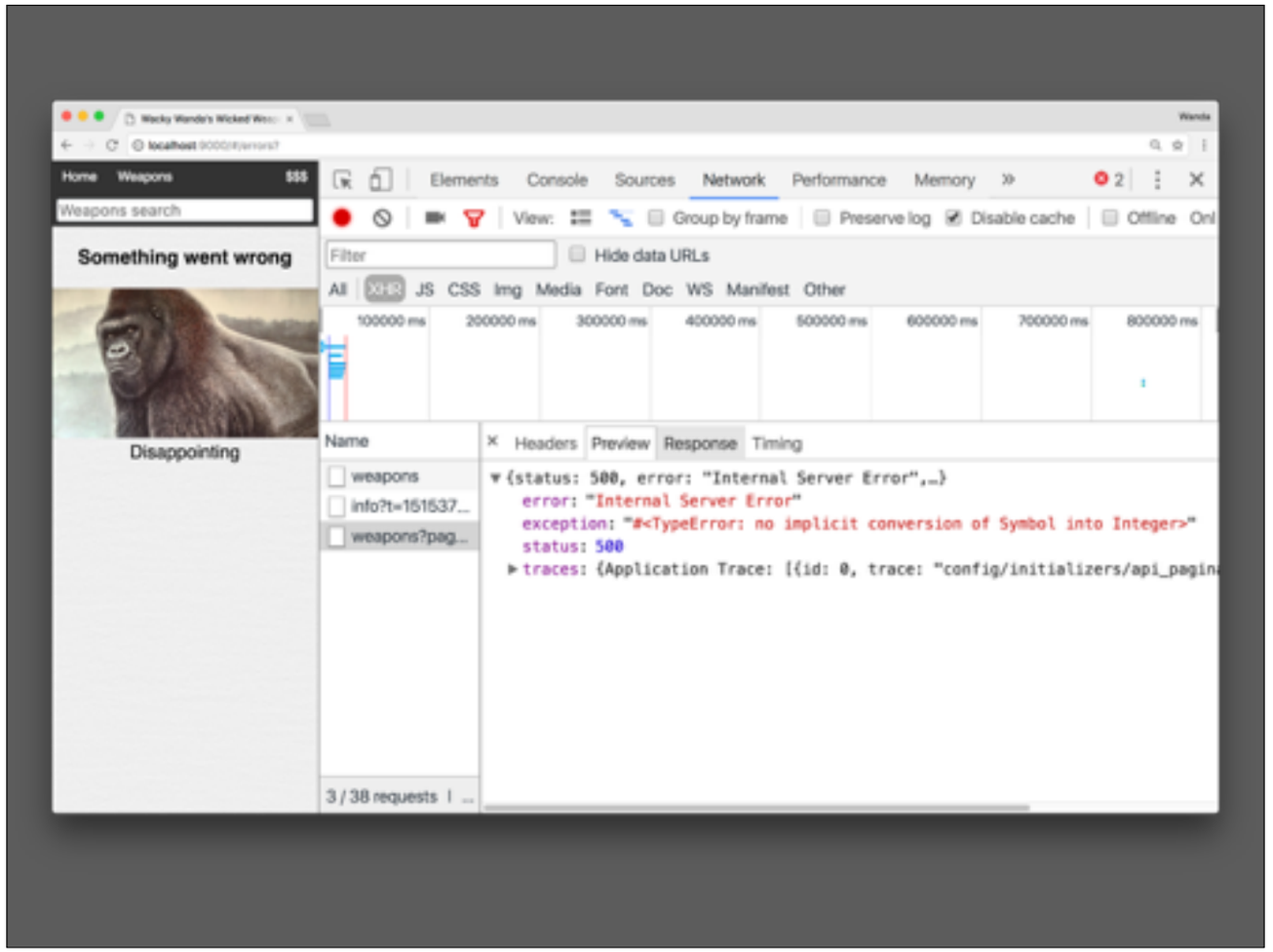
Let’s try going to the next page. We have a red ajax request. That can’t be good!



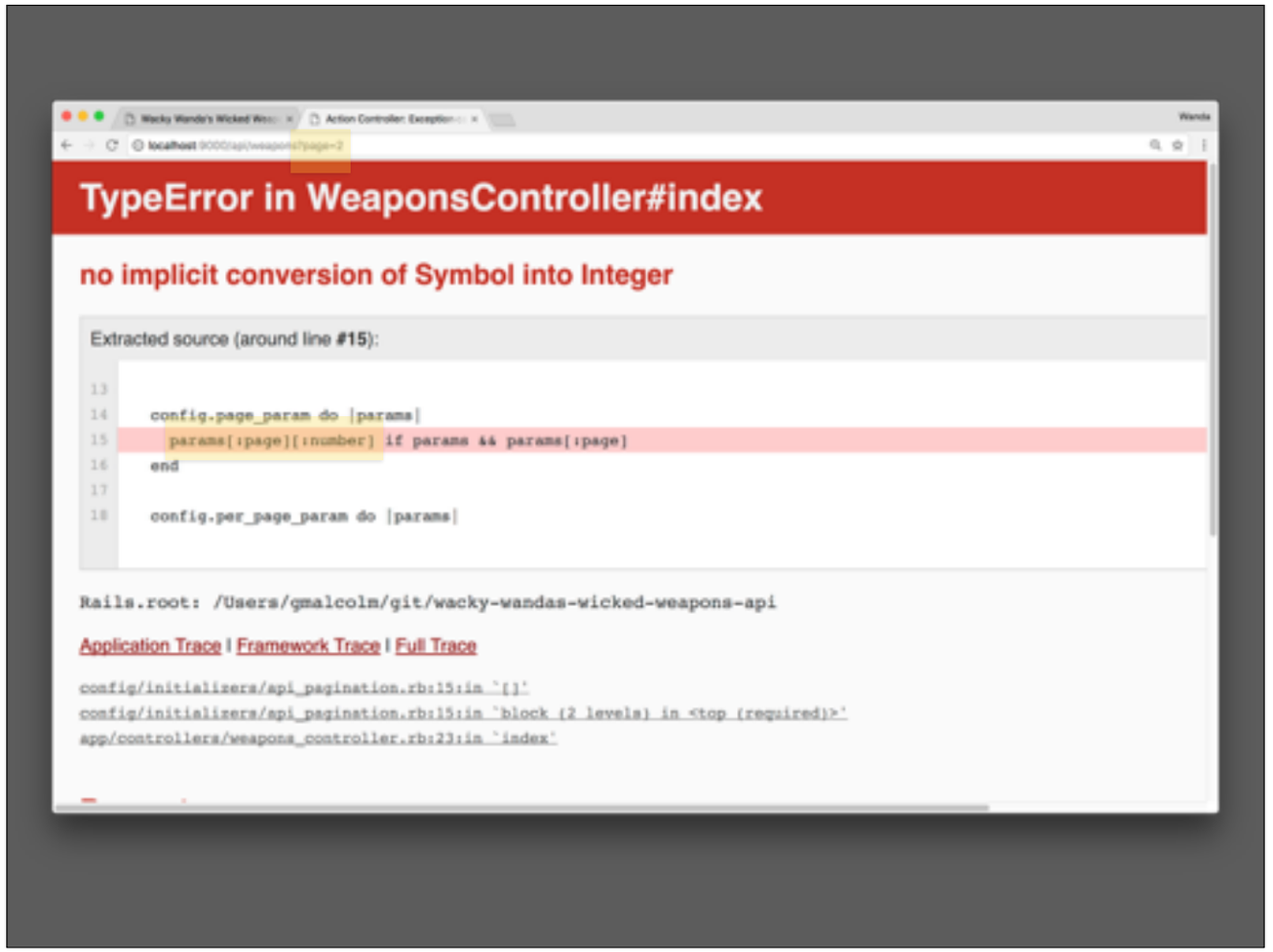
If we click on it we can see it was trying to hit the API endpoint using this url:

http://localhost:9000/api/weapons?page=2

Looks like it failed on 500 Internal Server Error. So probably not our fault!



The preview shows some extra exception information. That means we must be connecting to some kind of DEV Server with additional troubleshooting information.

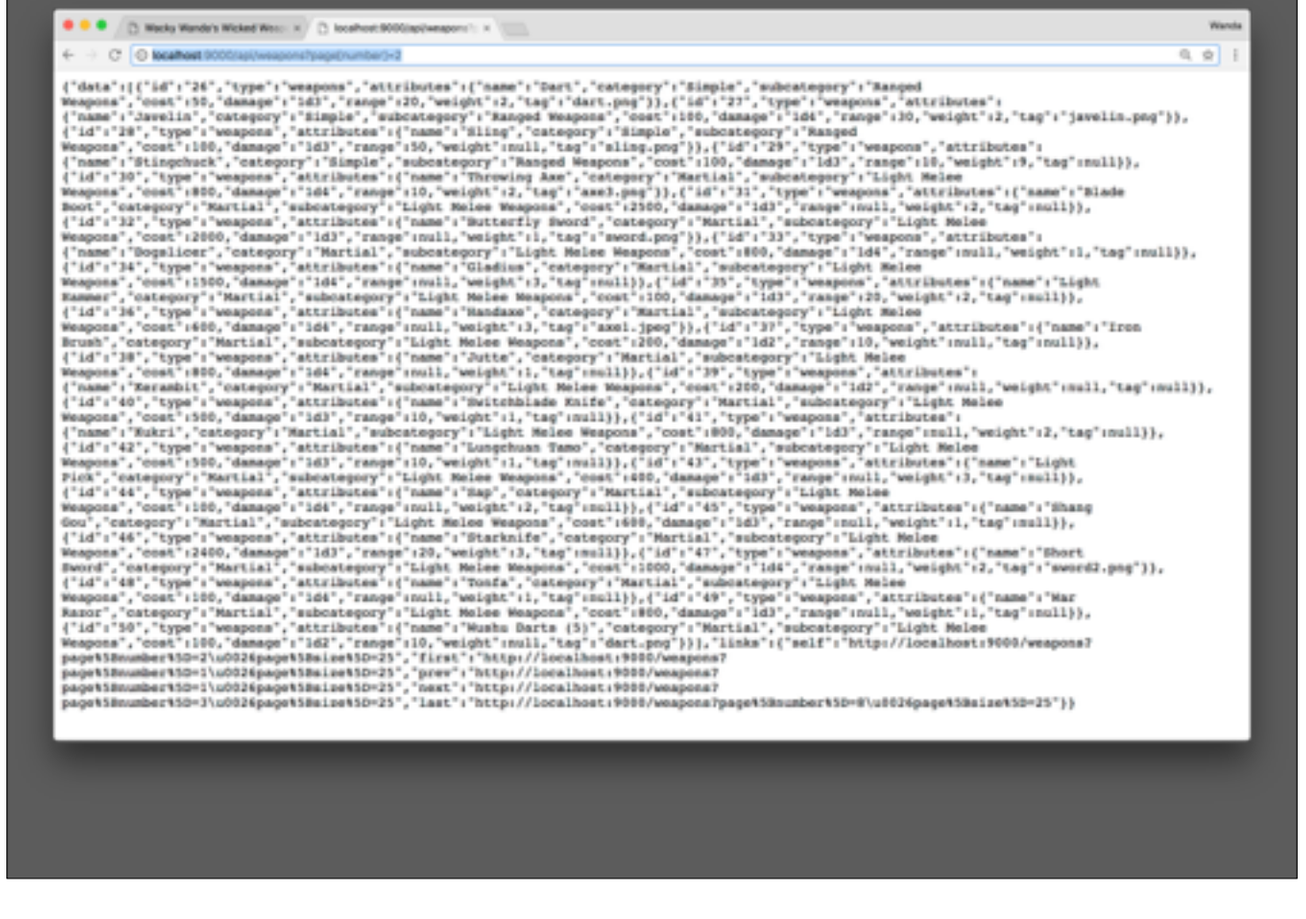


Let’s follow the API url we found. Looks like some kind of Ruby on Rails server.

Even if I didn’t know ruby, that params error is telling. It looks like it’s trying to reach **page[number]** as a query param. The url shows plain “**page**”.

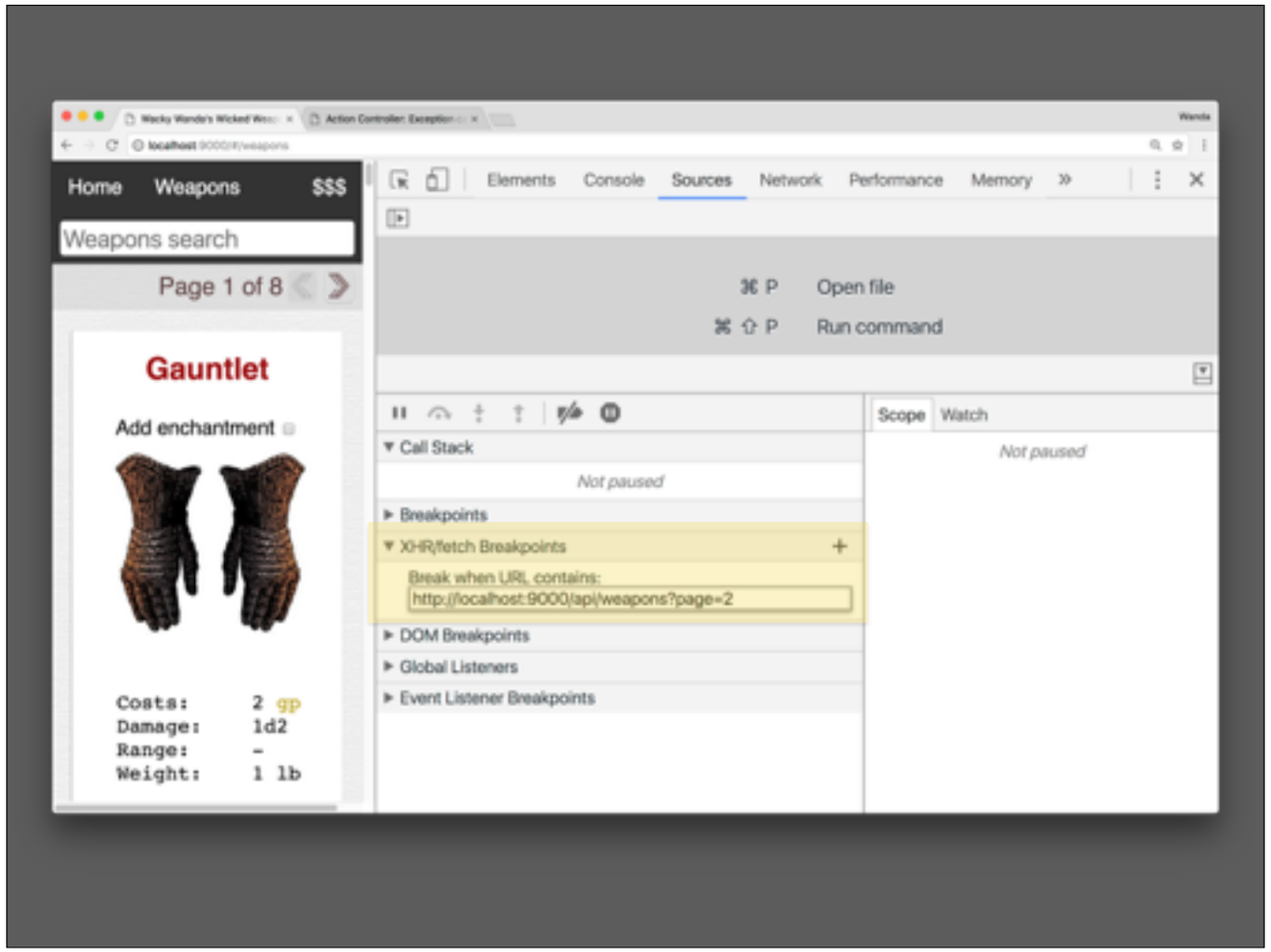
I wonder what would happen if we corrected the url to this?

http://localhost:9000/api/weapons?page[number]=2

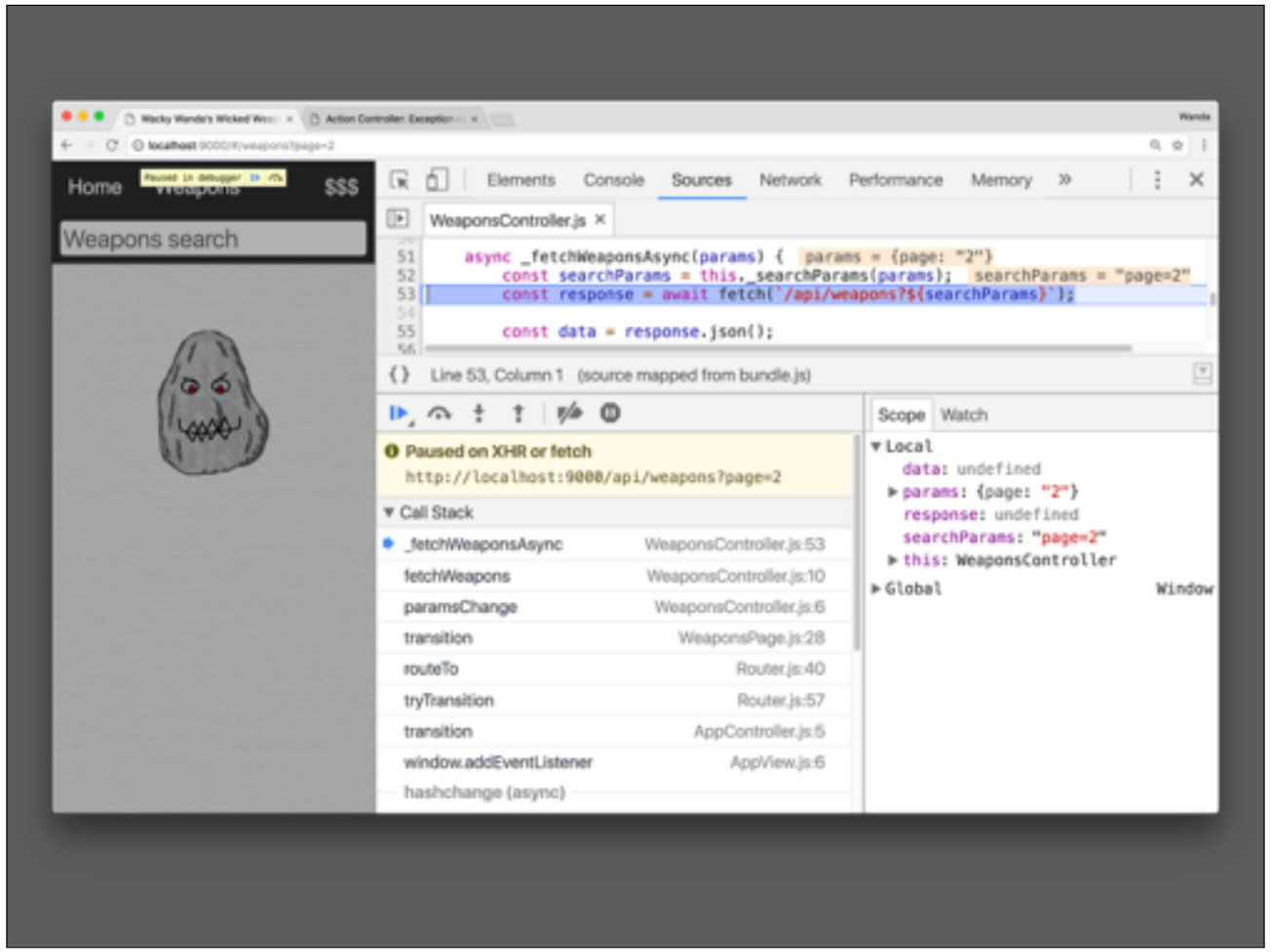


{"data":[{"id":"26","type":"weapons","attributes":{"name":"Dart","category":"Simple","subcategory":"Ranged
Weapons","cost":50,"damage":"1d3","range":20,"weight":2,"tag":"dart.png"}},{"id":"27","type":"weapons","attributes":
{"name":"Javelin","category":"Simple","subcategory":"Ranged Weapons","cost":150,"damage":"1d4","range":30,"weight":2,"tag":"javelin.png"}},
{"id":"28","type":"weapons","attributes":{"name":"Sling","category":"Simple","subcategory":"Ranged
Weapons","cost":100,"damage":"1d3","range":50,"weight":null,"tag":"sling.png"}},{"id":"29","type":"weapons","attributes":
{"name":"Stingchuck","category":"Simple","subcategory":"Ranged Weapons","cost":100,"damage":"1d3","range":10,"weight":9,"tag":null}},
{"id":"30","type":"weapons","attributes":{"name":"Throwing Axe","category":"Martial","subcategory":"Light Melee
Weapons","cost":800,"damage":"1d4","range":10,"weight":2,"tag":"axe3.png"}},{"id":"31","type":"weapons","attributes":{"name":"Blade
Boot","category":"Martial","subcategory":"Light Melee Weapons","cost":2500,"damage":"1d3","range":null,"weight":2,"tag":null}},
{"id":"32","type":"weapons","attributes":{"name":"Butterfly Sword","category":"Martial","subcategory":"Light Melee
Weapons","cost":2000,"damage":"1d3","range":null,"weight":1,"tag":"sword.png"}},{"id":"33","type":"weapons","attributes":
{"name":"Dogslicer","category":"Martial","subcategory":"Light Melee Weapons","cost":800,"damage":"1d4","range":null,"weight":1,"tag":null}},
{"id":"34","type":"weapons","attributes":{"name":"Gladius","category":"Martial","subcategory":"Light Melee
Weapons","cost":1500,"damage":"1d6","range":null,"weight":3,"tag":null}},{"id":"35","type":"weapons","attributes":{"name":"Light
Hammer","category":"Martial","subcategory":"Light Melee Weapons","cost":100,"damage":"1d3","range":20,"weight":2,"tag":null}},
{"id":"36","type":"weapons","attributes":{"name":"Handaxe","category":"Martial","subcategory":"Light Melee
Weapons","cost":600,"damage":"1d6","range":null,"weight":3,"tag":"axe1.jpeg"}},{"id":"37","type":"weapons","attributes":{"name":"Iron
Brush","category":"Martial","subcategory":"Light Melee Weapons","cost":200,"damage":"1d2","range":10,"weight":null,"tag":null}},
{"id":"38","type":"weapons","attributes":{"name":"Jutte","category":"Martial","subcategory":"Light Melee
Weapons","cost":800,"damage":"1d4","range":null,"weight":1,"tag":null}},{"id":"39","type":"weapons","attributes":
{"name":"Kerambit","category":"Martial","subcategory":"Light Melee Weapons","cost":200,"damage":"1d2","range":null,"weight":null,"tag":null}},
{"id":"40","type":"weapons","attributes":{"name":"Switchblade Knife","category":"Martial","subcategory":"Light Melee
Weapons","cost":500,"damage":"1d3","range":10,"weight":1,"tag":null}},{"id":"41","type":"weapons","attributes":
{"name":"Kukri","category":"Martial","subcategory":"Light Melee Weapons","cost":800,"damage":"1d3","range":null,"weight":2,"tag":null}},
{"id":"42","type":"weapons","attributes":{"name":"Lungchuan Tamo","category":"Martial","subcategory":"Light Melee
Weapons","cost":500,"damage":"1d3","range":10,"weight":1,"tag":null}},{"id":"43","type":"weapons","attributes":{"name":"Light
Pick","category":"Martial","subcategory":"Light Melee Weapons","cost":400,"damage":"1d3","range":null,"weight":3,"tag":null}},
{"id":"44","type":"weapons","attributes":{"name":"Nap","category":"Martial","subcategory":"Light Melee
Weapons","cost":100,"damage":"1d4","range":null,"weight":2,"tag":null}},{"id":"45","type":"weapons","attributes":{"name":"Shang
Gou","category":"Martial","subcategory":"Light Melee Weapons","cost":600,"damage":"1d3","range":null,"weight":1,"tag":null}},
{"id":"46","type":"weapons","attributes":{"name":"Starknife","category":"Martial","subcategory":"Light Melee
Weapons","cost":2400,"damage":"1d3","range":20,"weight":3,"tag":null}},{"id":"47","type":"weapons","attributes":{"name":"Short
Sword","category":"Martial","subcategory":"Light Melee Weapons","cost":1000,"damage":"1d4","range":null,"weight":2,"tag":"sword2.png"}},
{"id":"48","type":"weapons","attributes":{"name":"Tonfa","category":"Martial","subcategory":"Light Melee
Weapons","cost":100,"damage":"1d4","range":null,"weight":1,"tag":null}},{"id":"49","type":"weapons","attributes":{"name":"War
Razor","category":"Martial","subcategory":"Light Melee Weapons","cost":800,"damage":"1d3","range":null,"weight":1,"tag":null}},
{"id":"50","type":"weapons","attributes":{"name":"Wushu Darts (5)","category":"Martial","subcategory":"Light Melee
Weapons","cost":100,"damage":"1d2","range":10,"weight":null,"tag":"dart.png"}}],"links":{"self":"http://localhost:9000/weapons?
page%5Bnumber%5D=2\u0026page%5Bsize%5D=25","first":"http://localhost:9000/weapons?
page%5Bnumber%5D=1\u0026page%5Bsize%5D=25","prev":"http://localhost:9000/weapons?
page%5Bnumber%5D=1\u0026page%5Bsize%5D=25","next":"http://localhost:9000/weapons?
page%5Bnumber%5D=3\u0026page%5Bsize%5D=25","last":"http://localhost:9000/weapons?page%5Bnumber%5D=8\u0026page%5Bsize%5D=25"}}

Yep, the corrected url works. Guess the problem is on our end. Next we need to find the client code that makes the ajax call. I know of 2 ways to set a breakpoint an ajax call. Let’s go with the more “normal” way first.

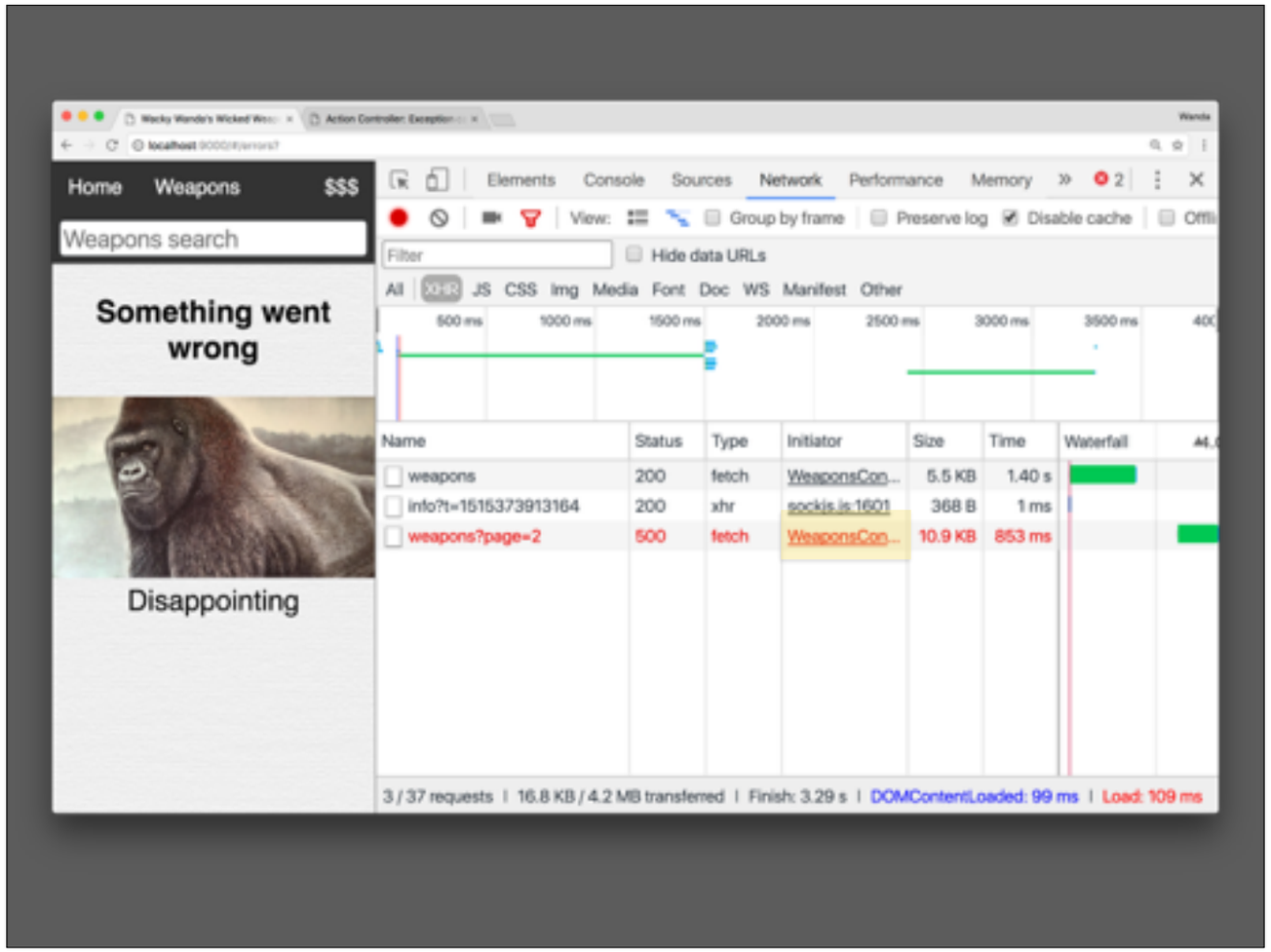


From the Sources Panel we can set an XHR/fetch breakpoint as one the debugging options. Just give it a whole or partial url and it'll break on each matching request with that url fragment.

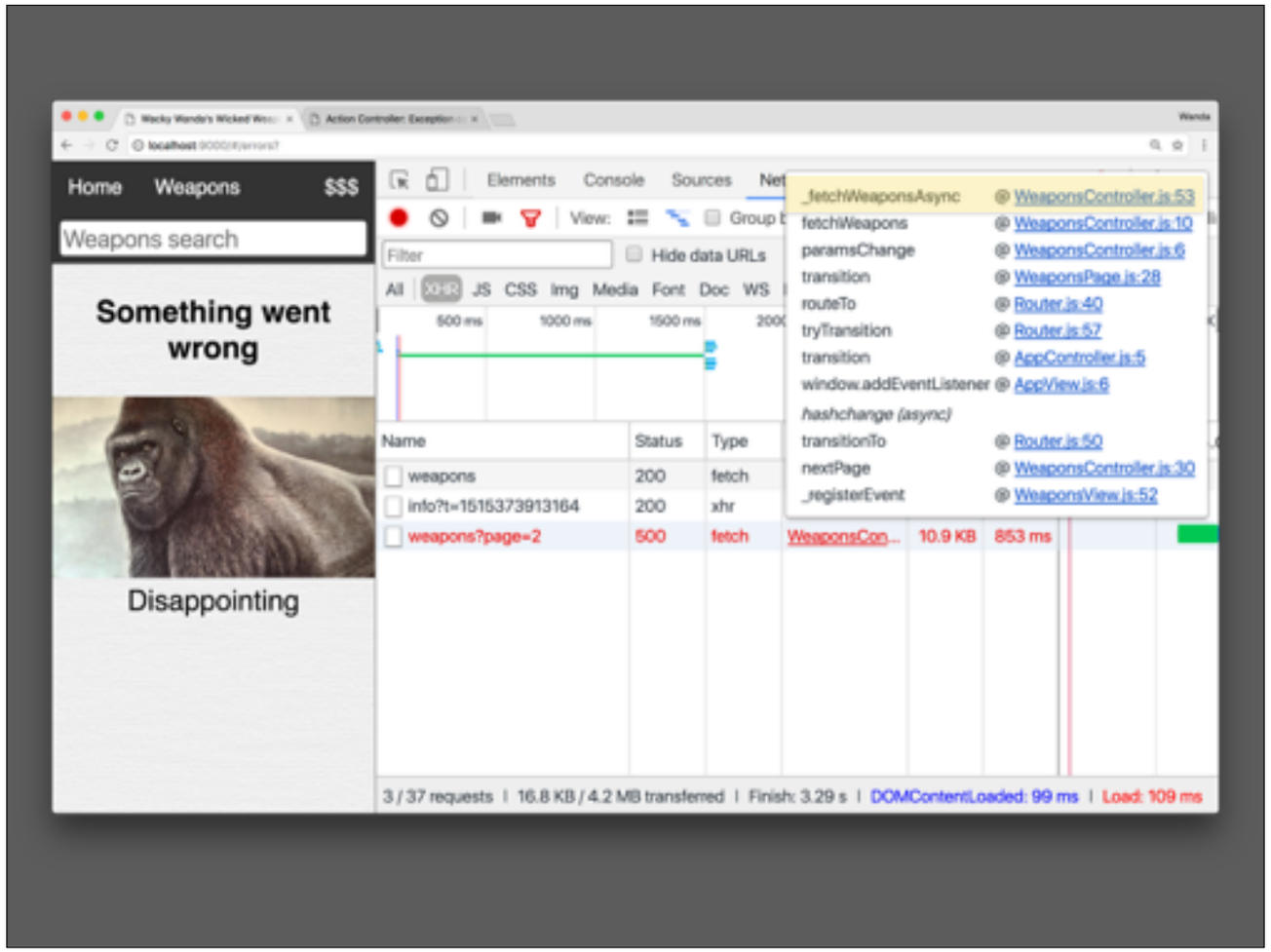


Yep, that worked just fine!

But theres another way which I think is even easier. So I’m going to clear away that breakpoint and try that way instead.

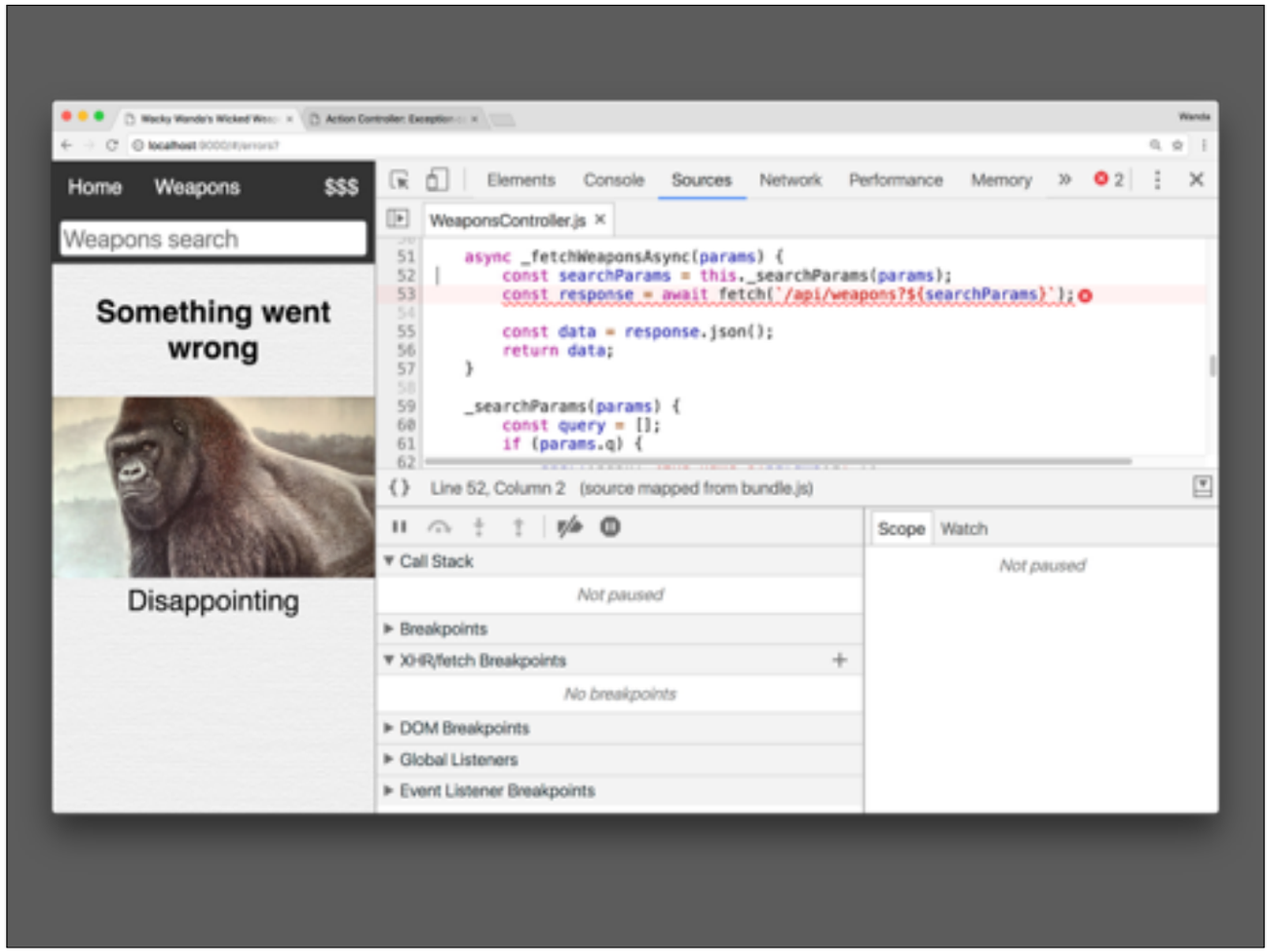


Here we are back at the Network Panel with the failed ajax call. If we don’t just click on the ajax line theres some columns showing. Notice the Initiator column? Let’s hover over the the contents of that.

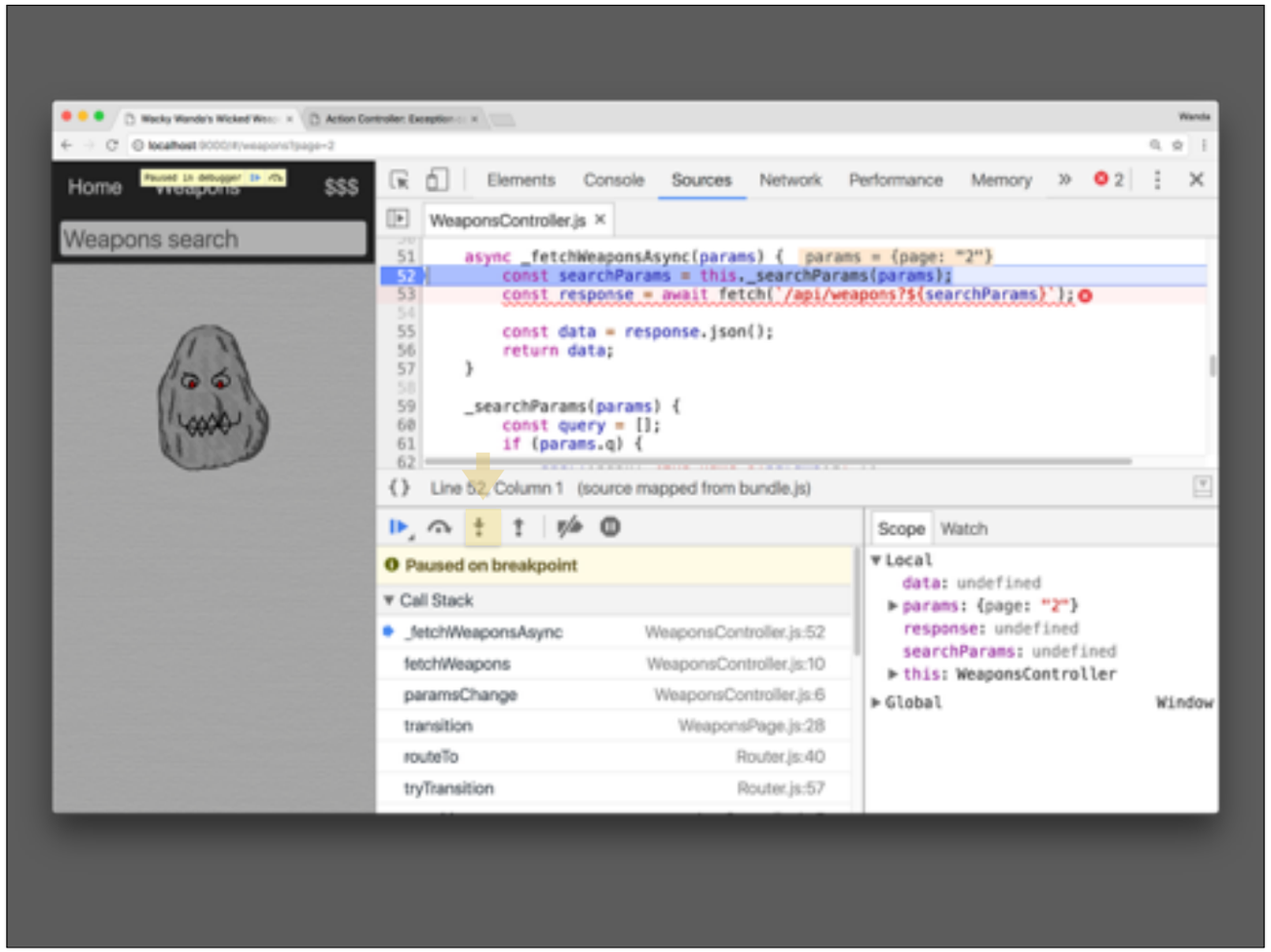


Look! It’s a stack trace!

I think _fetchWeaponsAsync looks the most promising point to break on. Let’s click there.

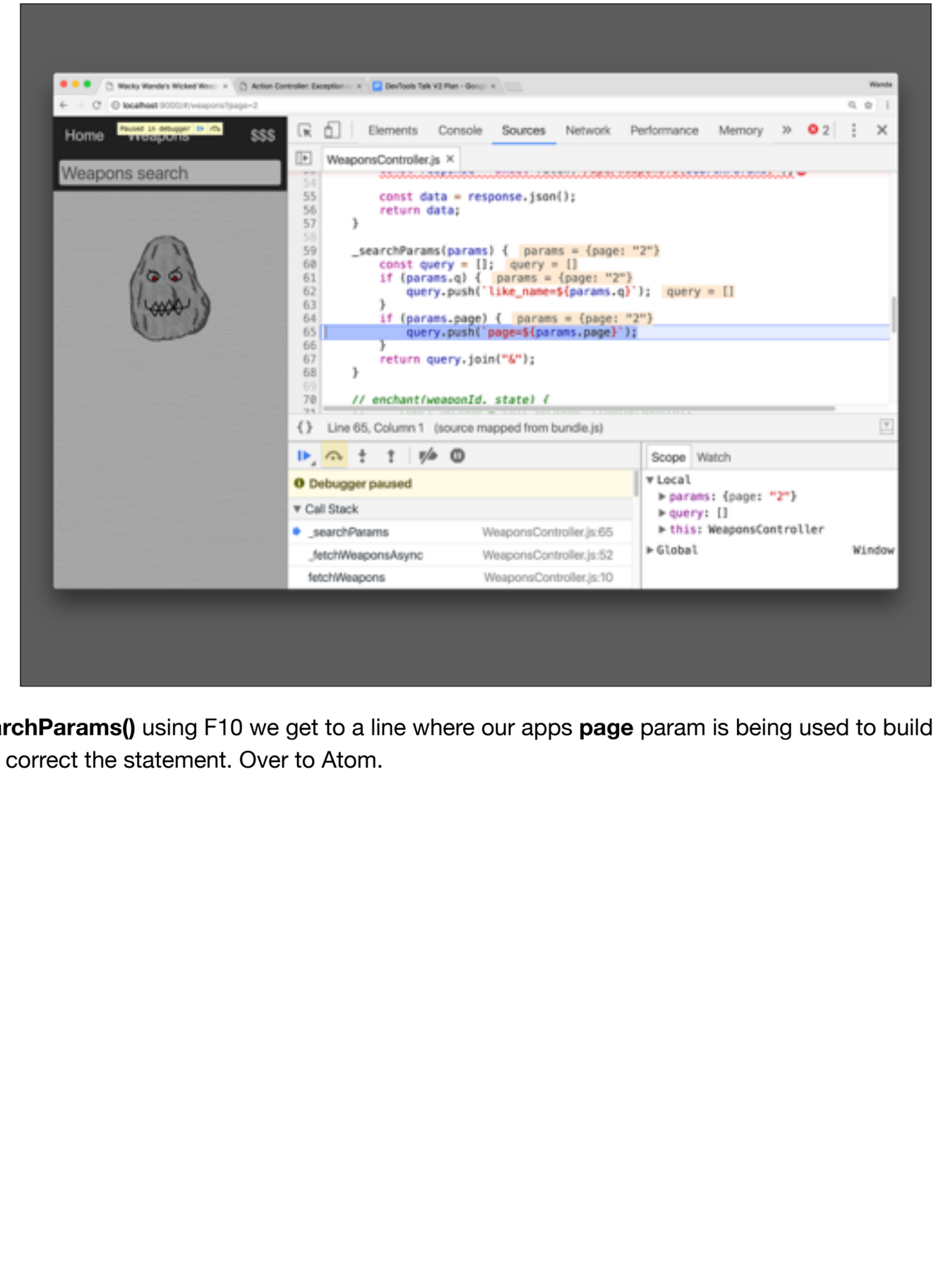


And here we are again! This time I'll need to chose where the breakpoint goes. I'm thinking that first line.

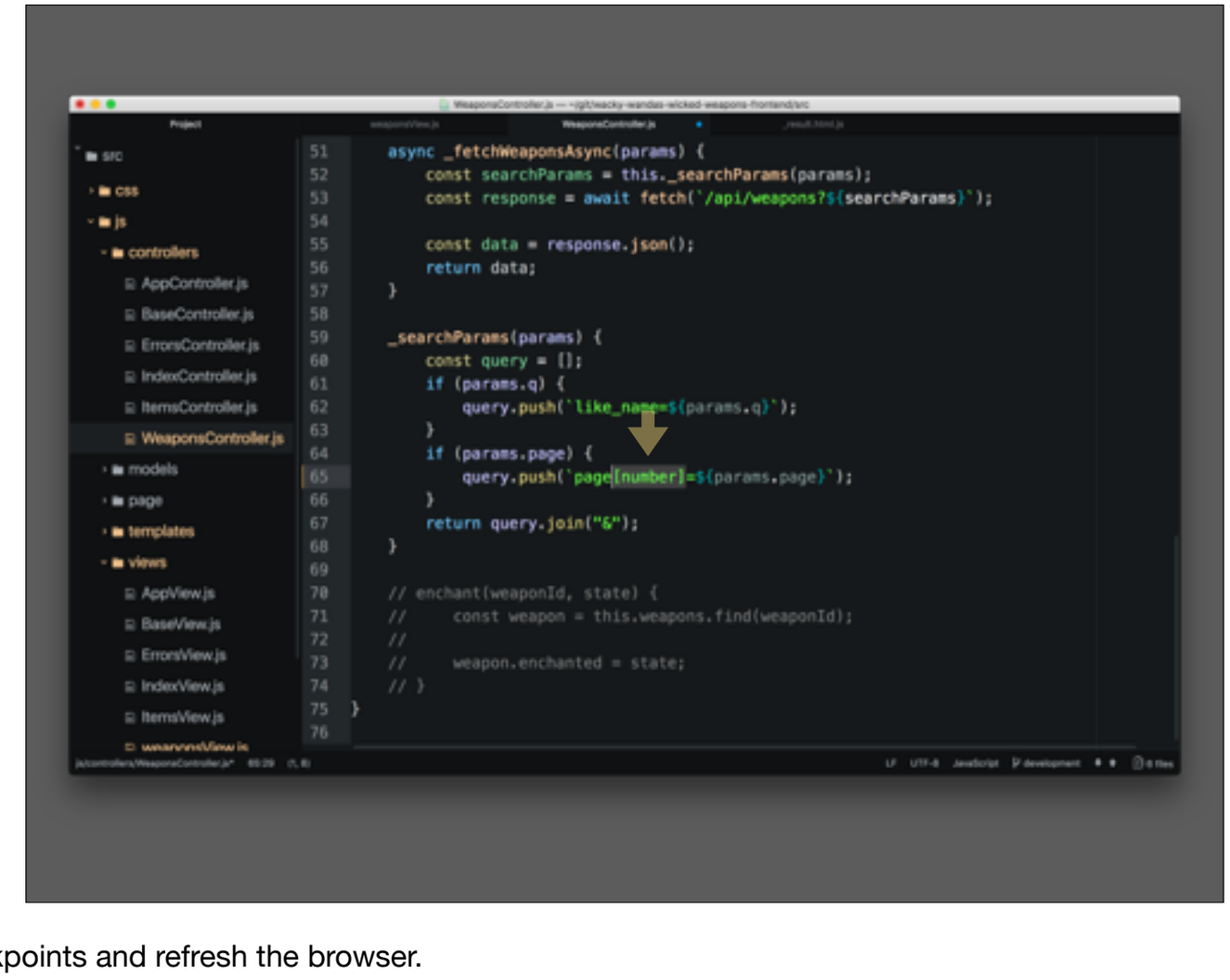


OK, let’s go retrigger the issue…

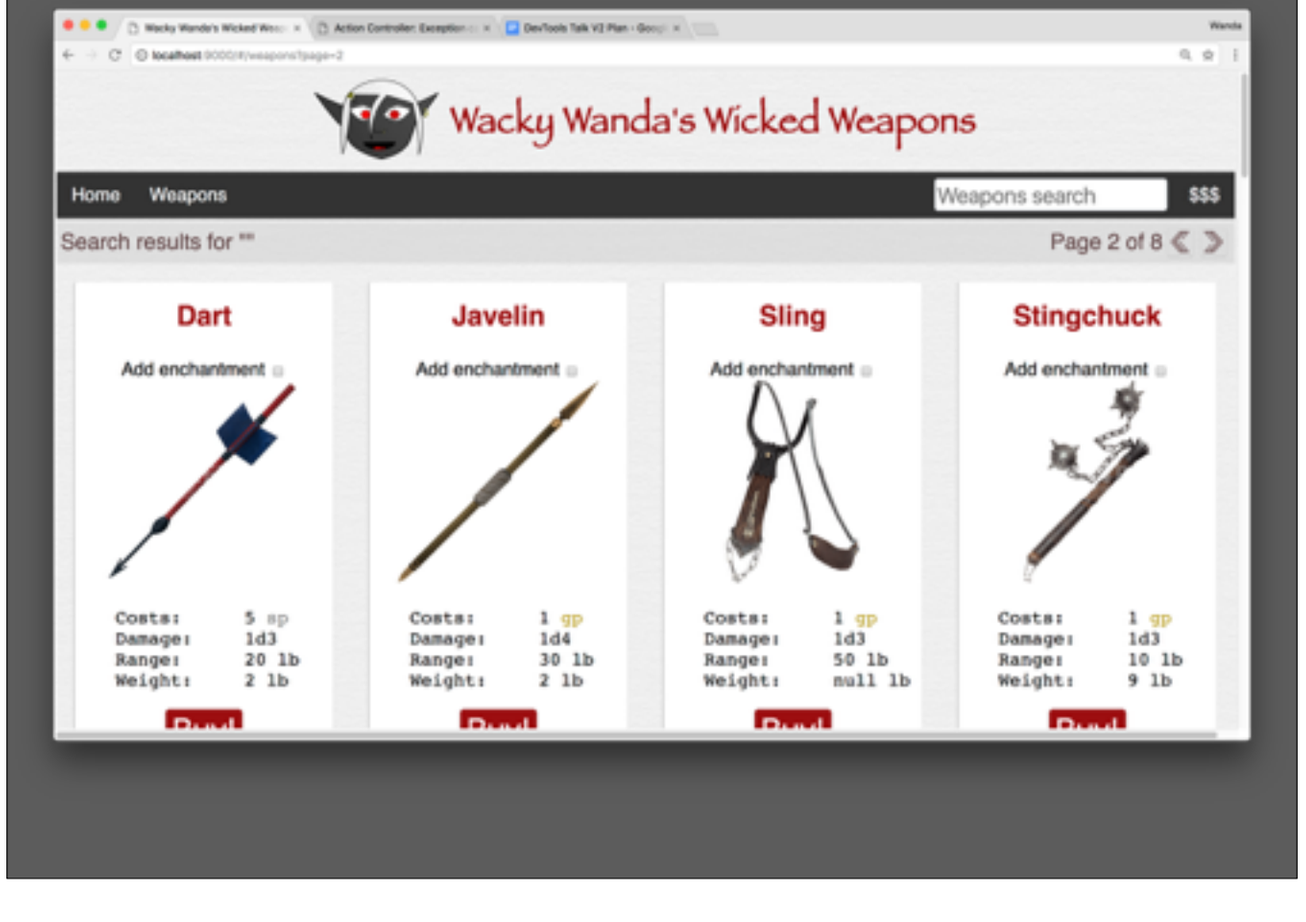
I want to take a look in _searchParams. I’ll need to click on Step In or press F11 for that.



If we step over a few times inside _**searchParams()** using F10 we get to a line where our apps **page** param is being used to build the api requests page param. This is clearly where the problem lies. Time to correct the statement. Over to Atom.



Insert “[number]”. Save. Remove breakpoints and refresh the browser.



Yep, looking good! Paging is now working

Resources

Learn Dev Tools:

https://developers.google.com/web/tools/chrome-devtools/

(or go to Dev Tools Menu -> Help -> Documentation)

Presentation Materials:

http://bit.ly/wickedweapons

or

https://speakerdeck.com/gregmalcolm/chrome-dev-tools

https://github.com/gregmalcolm/wacky-wandas-wicked-weapons-frontend

https://github.com/gregmalcolm/wacky-wandas-wicked-weapons-api