

Contemporary Web Development

Lesson 4

<https://www.yo>



InstaStalk solution

The Browser's developer tools



I would actually largely recommend using Firefox. The reason I am using Chrome for this course is:

A) It is easier to get it installed in the classroom.

B) Firefox has a [bug regarding Websockets monitoring](#) which I am teaching in this course.

Firefox developer edition has neat tools for HTML/CSS debugging such as Grid/Flexbox viewer.

The Console

Definitely the most important tool.

Log levels

Exposing Yukino

Using the window object to make our own object accessible in the console.

Responsive View (Example)

Elements Inspector

(What's wrong in this picture?)

Change the image to

`width: 200px;`

And the container to:

`height: 400px;`

`display: flex;`

`justify-content: center;`

Can we script this?

"Show console drawer" to see the console at the same time of the elements inspector.

```
let camilo =  
  document.querySelector("img[src*='juan']");  
camilo.style.width = "200px";  
camilo.parentNode.style.display = "flex";  
camilo.parentNode.style.justifyContent = "center";
```

Bookmarklet

Automate with TamperMonkey
(Or GreaseMonkey for Firefox)

Jamming Facebook's notification icon

```
let bell =  
  document.querySelector("#fbNotificationsJewel");  
bell.className = "uiToggle _4962 _4xi2 _5orm";  
let bellCount = bell.querySelector(".jewelCount");  
bellCount.style.display = "none";
```

A bit deeper

The class gets overridden by facebook's API!
Let's try to set it at the element level:

```
let bell =  
  document.querySelector("#fbNotificationsJewel");  
let icon = bell.querySelector("._2n_9");  
icon.style.backgroundPosition = "0 -792px"  
let bellCount = bell.querySelector(".jewelCount");  
bellCount.style.display = "none";
```

Let's go even DEEPER

We want to see where it is that the notification button gets updated after loading the page

Search the sources

Search for fbNotificationsJewel, then we could add a source breakpoint.

The image shows the 'Settings' > 'Preferences' window in Chrome DevTools. The 'Sources' section is highlighted with a red box, containing the following options:

- Search in anonymous and content scripts
- Automatically reveal files in sidebar
- Enable JavaScript source maps
- Detect indentation
- Autocompletion
- Bracket matching

Other sections visible include:

- Preferences:**
 - Enable Ctrl + 1-9 shortcut to switch panels
 - Show third party URL badges
 - Show What's New after each update
 - Don't show Chrome Data Saver warning
 - Disable paused state overlay
- Console:**
 - Show native functions in JS Profile
 - Hide chrome frame in Layers view
 - Hide network messages
 - Selected context only
 - Log XMLHttpRequests
 - Show timestamps
 - Autocomplete from history
 - Group similar
 - Eager evaluation
 - Preserve log upon navigation
 - Enable custom formatters

The image shows the 'Console' settings panel in Chrome DevTools. The 'Preserve log' option is highlighted with a red box. The panel includes the following options:

- Hide network
- Preserve log
- Selected context only
- Group similar
- Log XMLHttpRequests
- Eager evaluation
- Autocomplete from history

Easier, a DOM breakpoint

Add a DOM breakpoint for fbNotificationsJewel

Network tracing

We found that the async request is
WebNotificationsPayloadPagelet

Intercepting the network data?
Not in the scope, but [here](#).

Memory viewer – detecting leaks

Memory-leak example.

Allocation timeline

The screenshot shows the Chrome DevTools interface with the Memory tab selected. The left sidebar displays a list of Allocation Timelines under the heading "ALLOCATION TIMELINES". The list includes six snapshots, each with a small icon and a memory usage value: Snapshot 2 (959 MB), Snapshot 3 (959 MB), Snapshot 4 (5.0 MB), Snapshot 5 (5.0 MB), and Snapshot 6 (959 MB). The main panel on the right is titled "Select profiling type" and offers three options: "Heap snapshot", "Allocation instrumentation on timeline" (which is selected), and "Allocation sampling". The "Allocation instrumentation on timeline" option is accompanied by a checkbox for "Record allocation stacks (extra performance overhead)". Below the profiling type selection, there is a section for "Select JavaScript VM instance" with a table showing a single instance: "959 MB / 974 MB" for "localhost:8080". At the bottom of the panel, there are two buttons: "Start" and "Load".

Elements Console Sources Network Performance **Memory** Application Security

Profiles

ALLOCATION TIMELINES

- Snapshot 2
959 MB
- Snapshot 3
959 MB
- Snapshot 4
5.0 MB
- Snapshot 5
5.0 MB
- Snapshot 6
959 MB

Select profiling type

- Heap snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript nodes.
- Allocation instrumentation on timeline
Allocation timelines show instrumented JavaScript memory allocations over time. You can select a time interval to see objects that were allocated within it and record. Use this profile type to isolate memory leaks.
 - Record allocation stacks (extra performance overhead)
- Allocation sampling
Record memory allocations using sampling method. This profile type has minimal overhead and can be used for long running operations. It provides good approximation of JavaScript execution stack.

Select JavaScript VM instance

959 MB / 974 MB	localhost:8080
-----------------	----------------

Start Load

Not covered:
Storage, Performance, Application

Exercise

Tamper away!

Make your own nice tampering and upload it to the group