Custom Object Formatters in Chrome DevTools

Sergei Vasilinetc, Brian Slesinsky, Paul Irish
last update: April 2016
this doc: bit.ly/object-formatters

Object formatters allow you to control how the value of a JavaScript object appears in Chrome's console and debugger.

How do I turn it on?

1. Open Dev Tools.
2. Click the gear in the upper right to go to Settings.
3. Under "Console", check "Enable custom formatters".

When will this custom formatter be used?

- In the javascript console
- Watches/Scopes
- In the popover while debugging: While your javascript is paused, you can hover a variable in the editor, and devtools will show an popover with the value of this variable

Can I see some examples?

1. andrewdavey/immutable-devtools: Chrome Dev Tools custom formatter for Immutable-js values
2. disjukr/vector-devtools: vector formatter for chrome devtools
3. binaryage/cljs-devtools: A collection of Chrome DevTools enhancements for ClojureScript developers
4. Three.js object formatters:
    a. twitter.com/thespite/status/656585905151545344
    b. twitter.com/thespite/status/656499298230734849

API

When this feature is enabled, Dev Tools will look for three special functions in the tab being debugged:

```
var htmlTemplate = window.devtoolsFormatters[i].header(object, config);
```

- Returns a single-line summary of the object or null. If null is returned, the default format will be used. Config parameter is optional. You can specify “config” attribute in object tag and assign it to an arbitrary object, this object will be passed here as second parameter.

```
var bool = window.devtoolsFormatters[i].hasBody(object, config);
```

- Returns true if the object can be opened to show more detail. Config - same as in header function.

```
var htmlTemplate = window.devtoolsFormatters[i].body(object, config);
```

- Returns the expanded version of the object, to be displayed after it's opened. Config - same as in header function.

The input is always the object to inspect. See below for the template format.

HTML Template Format

Each HTML template is encoded in a format based on the JsonML standard. Each element is represented as list in the format:

```
[tagName, {"style": "name: value; ..."}, child1, ...]
```

The following HTML tags are allowed:

```
div, span, ol, li, table, tr, td.
```

The optional `style` attribute may contain any valid css key.

A child can be another element, a string, or an object reference.

Child objects can be substituted into template by using a object reference, which works sort of like a template variable. The format of an object reference is:

["object", {"object": *objectToInspect, "config": configObject*}]

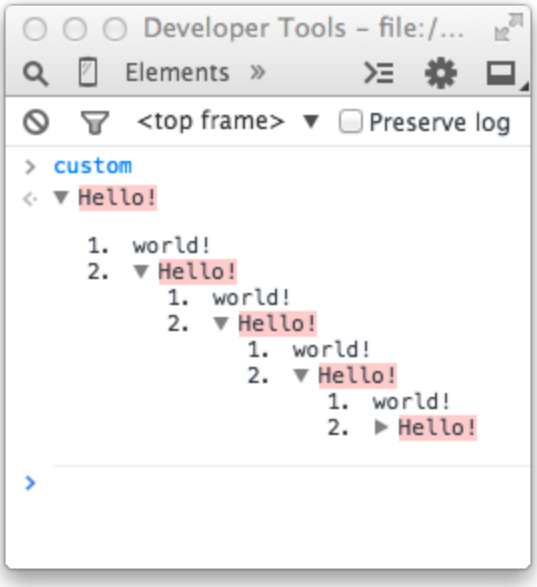
.

A template that contains object references isn't actually valid JSON. Since Dev Tools supports remote debugging, it will automatically substitute the appropriate object id before sending it to the Dev Tools front end.

(The templates are parsed in CustomPreviewComponent.ts.)

Example: Recursive Hello World

Here's a web page with with one custom object that points to itself.



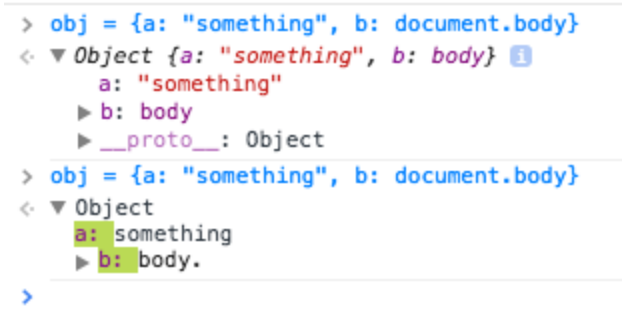

```
<!doctype html>
<html>
<head>
  <title>Hello, Custom Formatter</title>
</head>
<body>
Custom Formatter Demo. In the console, try inspecting the <i>custom</i> variable.
<script>
  var custom = {};
  (function() {
         var formatter = {
               header: function(x) {
                      if (x === custom) {
                             return ["span", {"style": "background-color: #fcc"}, "Hello!"];
                      } else {
                             return null;
                      }
               },
               hasBody: function(x) {
                      return x === custom;
               },
               body: function(x) {
                      var selfRef = ["object", {"object": custom}];
                      return ["ol", {}, ["li", {}, "world!"], ["li", {}, selfRef]];
               }
         };
         window.devtoolsFormatters = [formatter];
  }
)();
</script>
```

```
</body>
</html>
```

Example: Sophisticated Simple Formatter

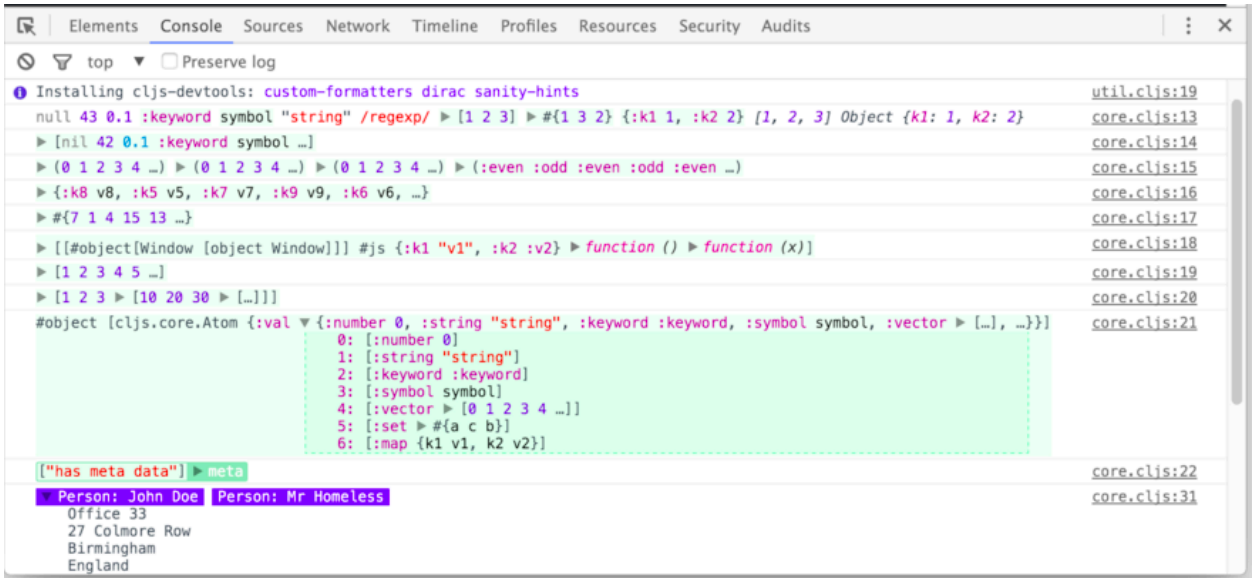
Source: gist.github.com/svasilinets/a422418689ed2e2e2729
JSbin demo: http://jsbin.com/sikufefice/1/edit?js,output

Try logging this:
obj = {a: "something", b: document.body}



Example: ClojureScript objects:

Source: github.com/binaryage/cljs-devtools



Example: GWT Objects

Source: gist.github.com/svasilinets/a422418689ed2e2e2729#file-gwt-formatter-js

Video:
drive.google.com/file/d/0BzvYe7bYFf--b19Mam1kaEZXWTg/view