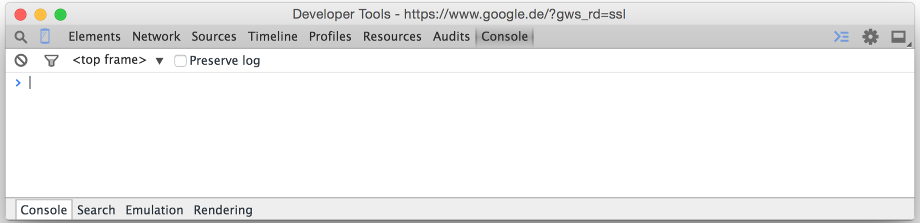
Webperformance, Debugging und Profiling

Martin Schuhfuss (@usefulthink) & Jakob Westhoff (@jakobwesthoff)

March 5th, 2015

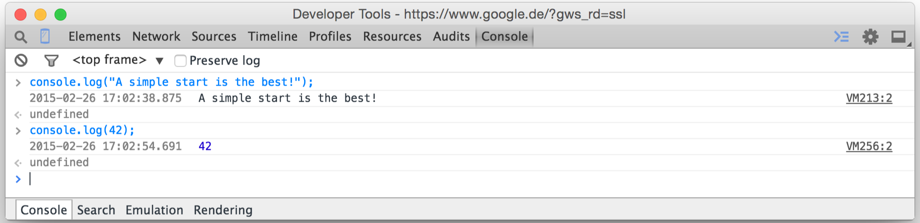
`console.*`

The Console


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Console Search Emulation Rendering

- The console is an integral part of the Developer Tools
- It can be programmatically accessed using the `console` object
- It is a lot more powerful, than you might think

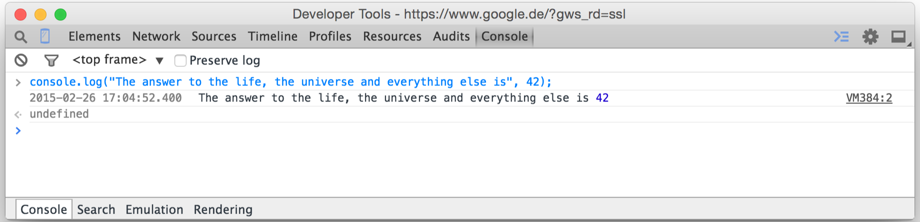
console.log


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Console Search Emulation Rendering

```
> console.log("A simple start is the best!");
  2015-02-26 17:02:38.875   A simple start is the best!                VM213:2
< undefined
> console.log(42);
  2015-02-26 17:02:54.691   42                                         VM256:2
< undefined
>
```

- `console.log(...)` is known to most developers
- It outputs arbitrary information to the dev-tools console

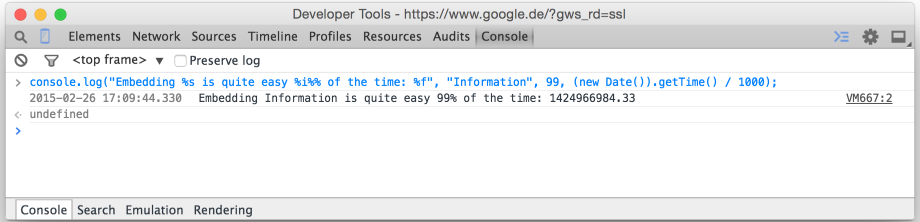
`console.log`


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log
Console Search Emulation Rendering

```
> console.log("The answer to the life, the universe and everything else is", 42);
  2015-02-26 17:04:52.400   The answer to the life, the universe and everything else is 42          VM384:2
<· undefined
>
```

- An arbitrary amount of arguments is accepted
- Output is concatenated

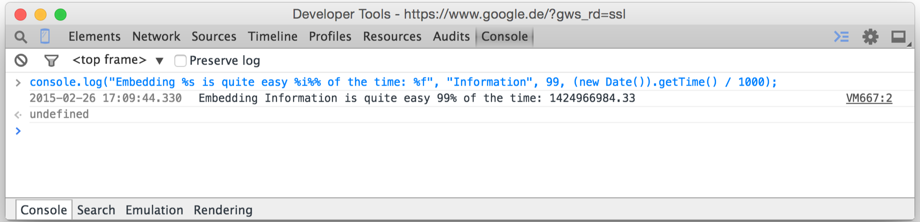
console.log


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log
Console Search Emulation Rendering

```
> console.log("Embedding %s is quite easy %i%% of the time: %f", "Information", 99, (new Date()).getTime() / 1000);
  2015-02-26 17:09:44.330   Embedding Information is quite easy 99% of the time: 1424966984.33              VM667:2
< undefined
>
```

- Format strings may be used to embed output
- Different output formats are available

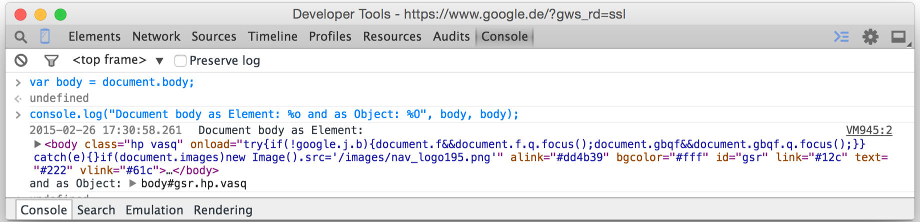
console.log


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> Preserve log
Console Search Emulation Rendering

```
> console.log("Embedding %s is quite easy %i%% of the time: %f", "Information", 99, (new Date()).getTime() / 1000);
  2015-02-26 17:09:44.330   Embedding Information is quite easy 99% of the time: 1424966984.33        VM667:2
<- undefined
>
```

- %s - Strings
- %i or %d - Integer values
- %f - Floating point Values

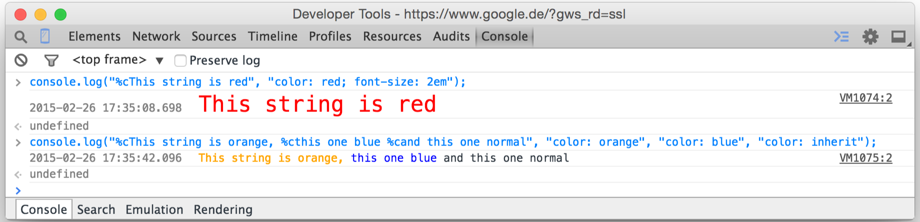
console.log


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Console Search Emulation Rendering

```
> var body = document.body;
< undefined
> console.log("Document body as Element: %o and as Object: %O", body, body);
  2015-02-26 17:30:58.261   Document body as Element:                                                         VM945:2
   ▶<body class="hp vasq" onload="try{if(!google.j.b){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}}
   catch(e){}if(document.images)new Image().src='/images/nav_logo195.png'" alink="#dd4b39" bgcolor="#fff" id="gsr" link="#12c" text=
   "#222" vlink="#61c">…</body>
  and as Object: ▶ body#gsr.hp.vasq
```

- Objects and DOM elements may be embedded as well
- %o - Expandable DOM element
- %0 - Expandable JavaScript Object

console.log

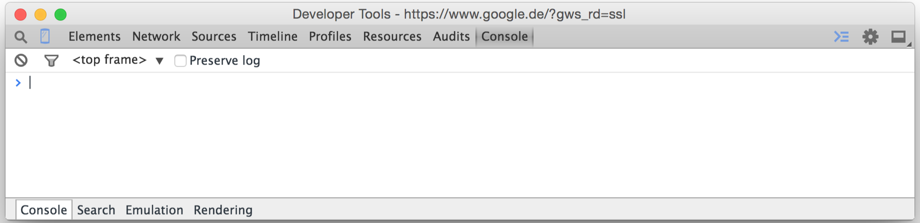

Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Console Search Emulation Rendering

```
> console.log("%cThis string is red", "color: red; font-size: 2em");
  2015-02-26 17:35:08.698 This string is red                                   VM1074:2
< undefined
> console.log("%cThis string is orange, %cthis one blue %cand this one normal", "color: orange", "color: blue", "color: inherit");
  2015-02-26 17:35:42.096 This string is orange, this one blue and this one normal   VM1075:2
< undefined
>
```

- The dev-tools are a webview as well
- Output maybe formatted using CSS

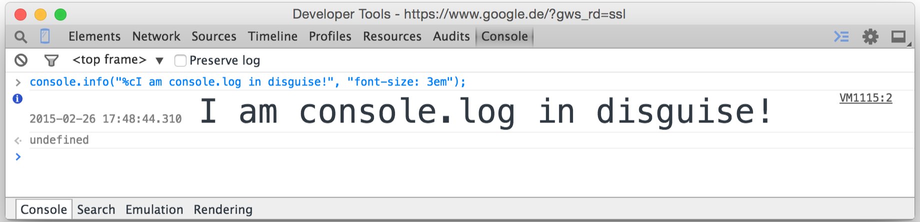
```
console.info
console.warn
console.error
```

info, warn and error


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> Preserve log
Console Search Emulation Rendering

- Different log levels exist within the `console` object

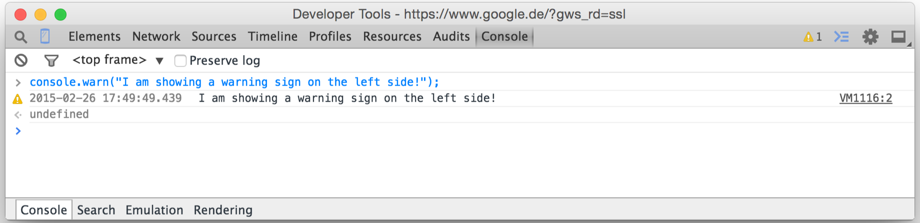
`console.info`


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Console Search Emulation Rendering

```
> console.info("%cI am console.log in disguise!", "font-size: 3em");
                                                                   VM1115:2
  2015-02-26 17:48:44.310 I am console.log in disguise!
< undefined
>
```

- `console.info` is an alias to `console.log`
- In addition it displays an info sign on the left side of the message

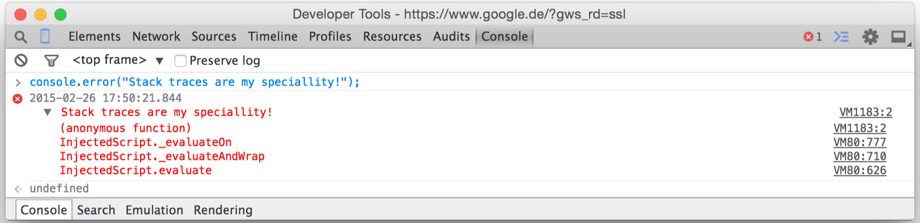
`console.warn`


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log
Console Search Emulation Rendering

```
> console.warn("I am showing a warning sign on the left side!");
⚠ 2015-02-26 17:49:49.439   I am showing a warning sign on the left side!          VM1116:2
< undefined
>
```

- `console.warn` behaves like `console.log`, but displays a warning sign left to the output
- Allows for easy highlighting of important messages

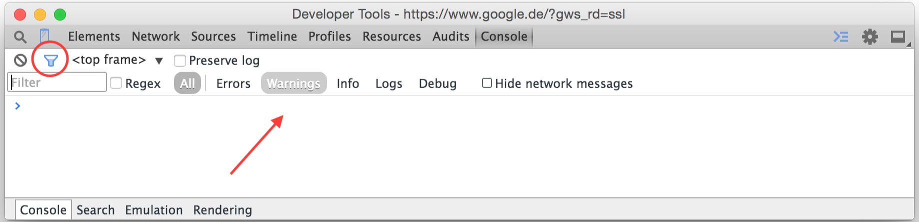
`console.error`


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> Preserve log
Console Search Emulation Rendering

```
> console.error("Stack traces are my speciallity!");
2015-02-26 17:50:21.844
  ▼ Stack traces are my speciallity!                                    VM1183:2
    (anonymous function)                                               VM1183:2
    InjectedScript._evaluateOn                                         VM80:777
    InjectedScript._evaluateAndWrap                                    VM80:710
    InjectedScript.evaluate                                            VM80:626
< undefined
```

- `console.error` is similar to `console.error` with an appended stack trace from where the method has been called
- Allows for easy backtracing the execution path, that caused an error

console.error

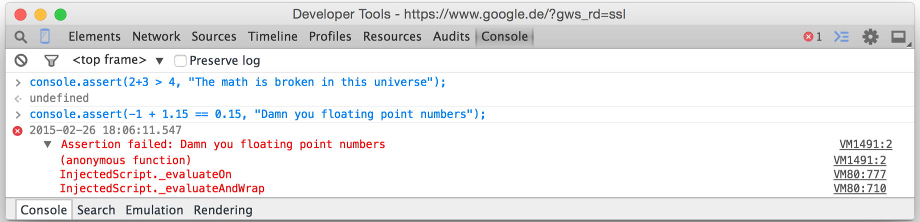


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Filter Regex All Errors Warnings Info Logs Debug Hide network messages
Console Search Emulation Rendering

- Filtering for messages with a certain log level is possible

console.assert

console.assert

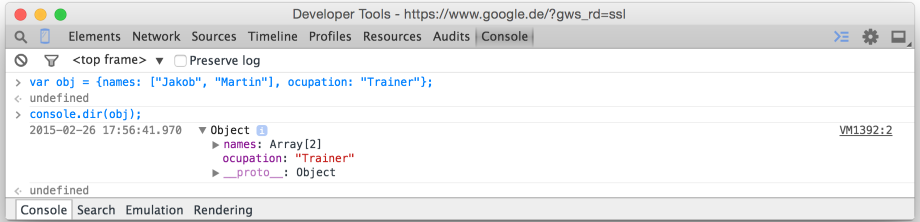

Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
1
<top frame>
Preserve log
Console Search Emulation Rendering

```
> console.assert(2+3 > 4, "The math is broken in this universe");
<· undefined
> console.assert(-1 + 1.15 == 0.15, "Damn you floating point numbers");
✖ 2015-02-26 18:06:11.547
  ▼ Assertion failed: Damn you floating point numbers                VM1491:2
     (anonymous function)                                            VM1491:2
     InjectedScript._evaluateOn                                      VM80:777
     InjectedScript._evaluateAndWrap                                 VM80:710
```

- `console.assert` is a conditional `console.error`
- A Stack trace as well as the error message is shown if the given expression is true

console.dir

`console.dir`

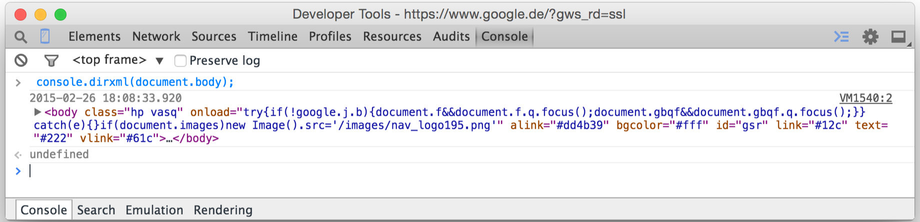

Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Console Search Emulation Rendering

```
> var obj = {names: ["Jakob", "Martin"], ocupation: "Trainer"};
< undefined
> console.dir(obj);
  2015-02-26 17:56:41.970   ▼ Object                                                    VM1392:2
                              ▶ names: Array[2]
                                ocupation: "Trainer"
                              ▶ __proto__: Object
< undefined
```

- Print an expandable version of a JavaScript object
- Identical to: `console.log("%O", object);`

console.dirxml

console.dirxml

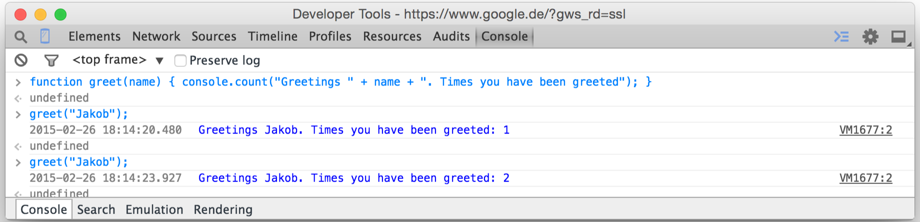

Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Console Search Emulation Rendering

```
> console.dirxml(document.body);
  2015-02-26 18:08:33.920                                                                    VM1540:2
  ▶<body class="hp vasq" onload="try{if(!google.j.b){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}}
  catch(e){}if(document.images)new Image().src='/images/nav_logo195.png'" alink="#dd4b39" bgcolor="#fff" id="gsr" link="#12c" text=
  "#222" vlink="#61c">…</body>
< undefined
> |
```

- Print an expandable version of a DOM Node
- Identical to: `console.log("%o", node);`

console.count

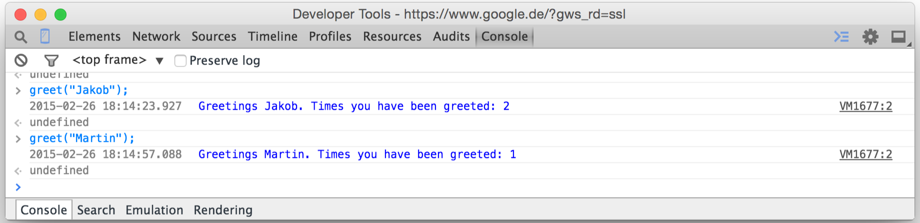
console.count


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Console Search Emulation Rendering

```
> function greet(name) { console.count("Greetings " + name + ". Times you have been greeted"); }
< undefined
> greet("Jakob");
  2015-02-26 18:14:20.480   Greetings Jakob. Times you have been greeted: 1                          VM1677:2
< undefined
> greet("Jakob");
  2015-02-26 18:14:23.927   Greetings Jakob. Times you have been greeted: 2                          VM1677:2
< undefined
```

- Like `console.log`, but adding an increasing number after each print
- The number is incremented each time the function is called from the same line using the same label

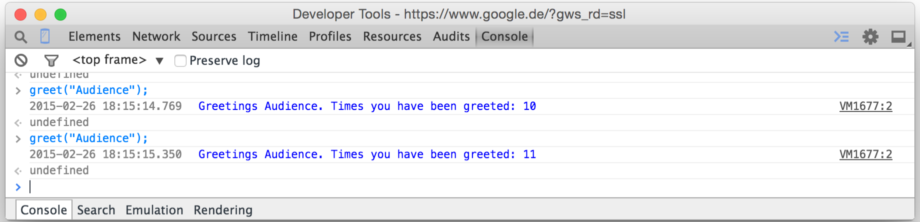
`console.count`


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log
Console Search Emulation Rendering

```
<· undefined
> greet("Jakob");
  2015-02-26 18:14:23.927  Greetings Jakob. Times you have been greeted: 2      VM1677:2
<· undefined
> greet("Martin");
  2015-02-26 18:14:57.088  Greetings Martin. Times you have been greeted: 1     VM1677:2
<· undefined
>
```

- Like `console.log`, but adding an increasing number after each print
- The number is incremented each time the function is called from the same line using the same label

console.count

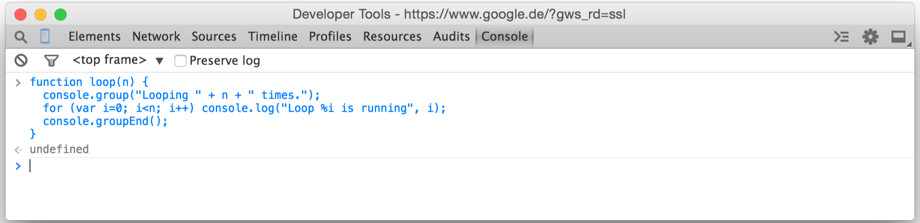

Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log
Console Search Emulation Rendering

```
> greet("Audience");
  2015-02-26 18:15:14.769   Greetings Audience. Times you have been greeted: 10       VM1677:2
< undefined
> greet("Audience");
  2015-02-26 18:15:15.350   Greetings Audience. Times you have been greeted: 11       VM1677:2
< undefined
>
```

- Like `console.log`, but adding an increasing number after each print
- The number is incremented each time the function is called from the same line using the same label

```
console.group
console.groupEnd
```

console.group

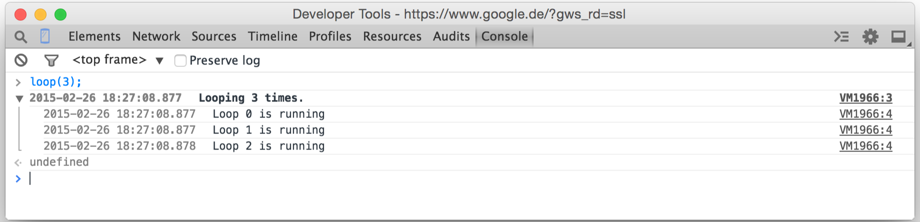


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> function loop(n) {
    console.group("Looping " + n + " times.");
    for (var i=0; i<n; i++) console.log("Loop %i is running", i);
    console.groupEnd();
  }
< undefined
>
```

- `console.group` in combination with `console.groupEnd` allows to structure messages in a tree

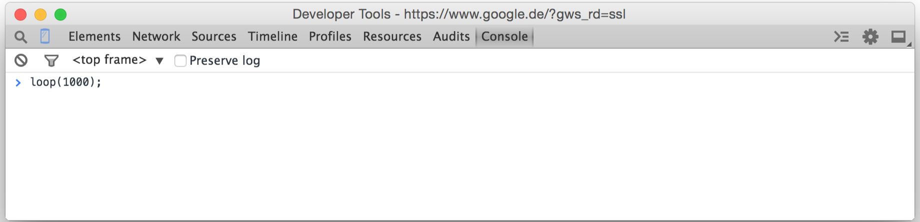
console.group


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> loop(3);
▼ 2015-02-26 18:27:08.877   Looping 3 times.          VM1966:3
    2015-02-26 18:27:08.877   Loop 0 is running       VM1966:4
    2015-02-26 18:27:08.877   Loop 1 is running       VM1966:4
    2015-02-26 18:27:08.878   Loop 2 is running       VM1966:4
< undefined
>
```

- `console.group` in combination with `console.groupEnd` allows to structure messages in a tree

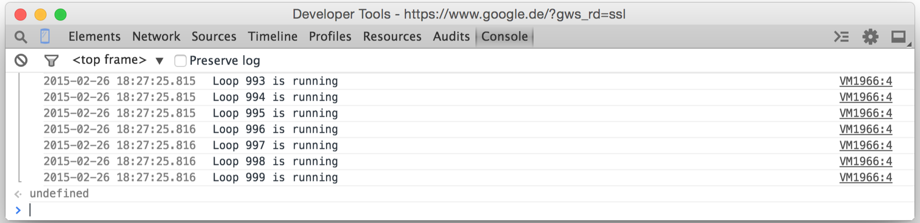
console.group


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> loop(1000);
```

- Groups work with an arbitrary amount of messages

console.group

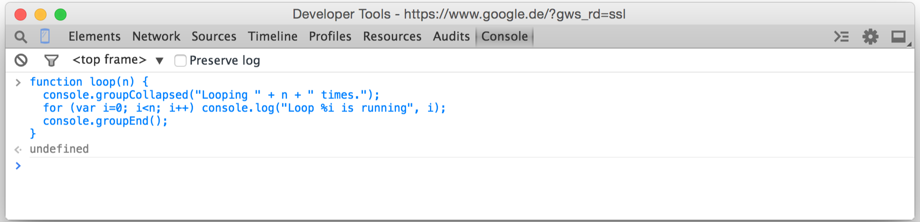


```
2015-02-26 18:27:25.815  Loop 993 is running    VM1966:4
2015-02-26 18:27:25.815  Loop 994 is running    VM1966:4
2015-02-26 18:27:25.815  Loop 995 is running    VM1966:4
2015-02-26 18:27:25.816  Loop 996 is running    VM1966:4
2015-02-26 18:27:25.816  Loop 997 is running    VM1966:4
2015-02-26 18:27:25.816  Loop 998 is running    VM1966:4
2015-02-26 18:27:25.816  Loop 999 is running    VM1966:4
< undefined
>
```

- With a lot of entries groups become as unreadable as simple log messages

`groupCollapsed`

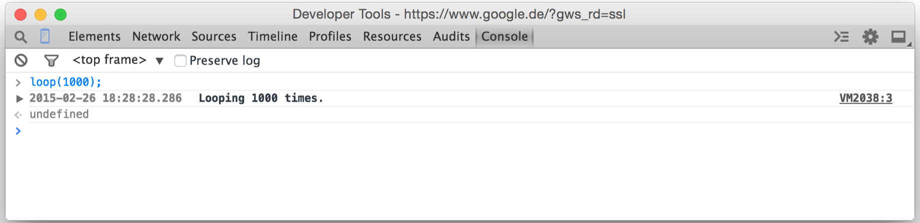
console.groupCollapsed


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> function loop(n) {
    console.groupCollapsed("Looping " + n + " times.");
    for (var i=0; i<n; i++) console.log("Loop %i is running", i);
    console.groupEnd();
  }
< undefined
>
```

- `console.groupCollapsed` is identical to `console.group` with the difference, that the group will be displayed collapsed by default

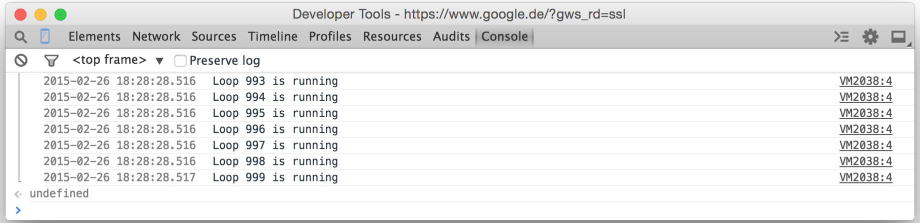
console.groupCollapsed


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame> ▼ Preserve log

```
> loop(1000);
▶ 2015-02-26 18:28:28.286   Looping 1000 times.                VM2038:3
< undefined
>
```

- `console.groupCollapsed` is identical to `console.group` with the difference, that the group will be displayed collapsed by default

`console.groupCollapsed`

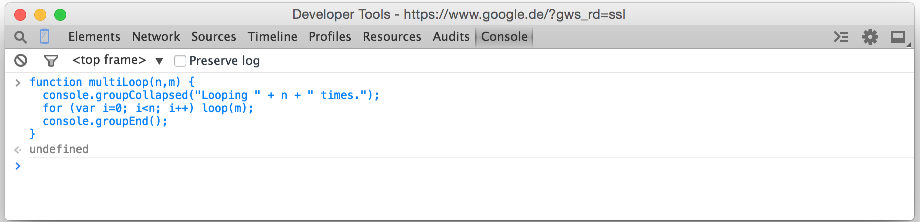

Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
2015-02-26 18:28:28.516  Loop 993 is running          VM2038:4
2015-02-26 18:28:28.516  Loop 994 is running          VM2038:4
2015-02-26 18:28:28.516  Loop 995 is running          VM2038:4
2015-02-26 18:28:28.516  Loop 996 is running          VM2038:4
2015-02-26 18:28:28.516  Loop 997 is running          VM2038:4
2015-02-26 18:28:28.516  Loop 998 is running          VM2038:4
2015-02-26 18:28:28.517  Loop 999 is running          VM2038:4
<· undefined
>
```

- Once the group is opened it is fully identical to a `console.group` output

Nesting Groups

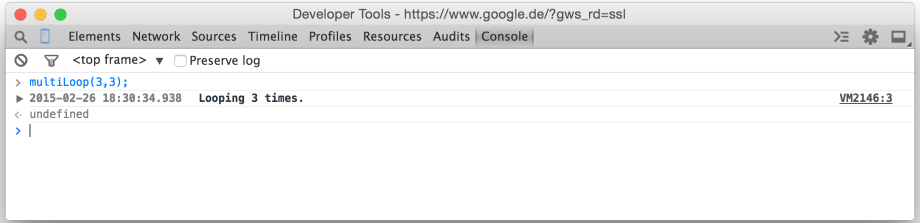
Nesting `console.group`



```
> function multiLoop(n,m) {
    console.groupCollapsed("Looping " + n + " times.");
    for (var i=0; i<n; i++) loop(m);
    console.groupEnd();
  }
< undefined
>
```

- Groupds can be nested to allow for more complex structures

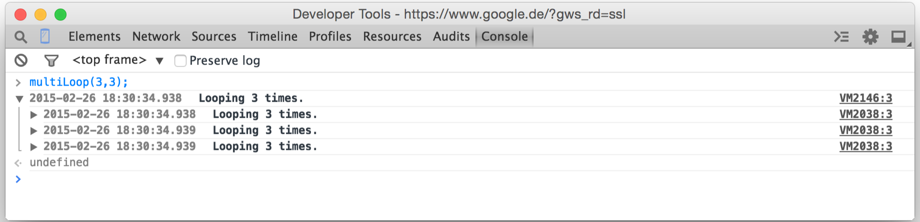
Nesting `console.group`


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> multiLoop(3,3);
▶ 2015-02-26 18:30:34.938   Looping 3 times.                                VM2146:3
<· undefined
>
```

- Groupds can be nested to allow for more complex structures

Nesting `console.group`

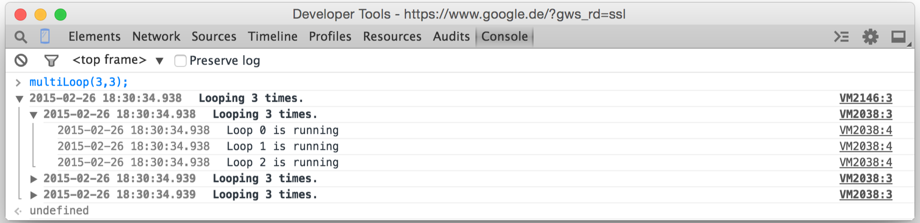


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> multiLoop(3,3);
▼ 2015-02-26 18:30:34.938   Looping 3 times.                VM2146:3
  ▶ 2015-02-26 18:30:34.938   Looping 3 times.              VM2038:3
  ▶ 2015-02-26 18:30:34.939   Looping 3 times.              VM2038:3
  ▶ 2015-02-26 18:30:34.939   Looping 3 times.              VM2038:3
<· undefined
>
```

- Groupds can be nested to allow for more complex structures

Nesting `console.group`

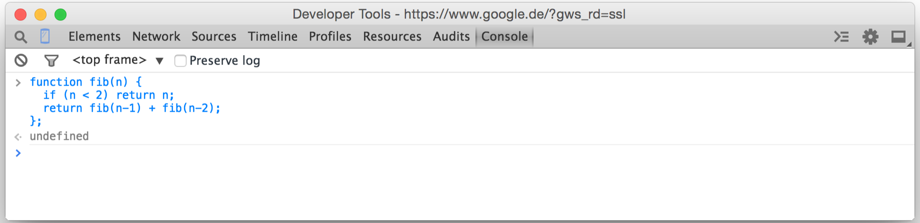

Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> multiLoop(3,3);
▼ 2015-02-26 18:30:34.938  Looping 3 times.                                                        VM2146:3
  ▼ 2015-02-26 18:30:34.938  Looping 3 times.                                                      VM2038:3
      2015-02-26 18:30:34.938  Loop 0 is running                                                   VM2038:4
      2015-02-26 18:30:34.938  Loop 1 is running                                                   VM2038:4
      2015-02-26 18:30:34.938  Loop 2 is running                                                   VM2038:4
  ▶ 2015-02-26 18:30:34.939  Looping 3 times.                                                      VM2038:3
  ▶ 2015-02-26 18:30:34.939  Looping 3 times.                                                      VM2038:3
<· undefined
```

- Groupds can be nested to allow for more complex structures

`console.time`

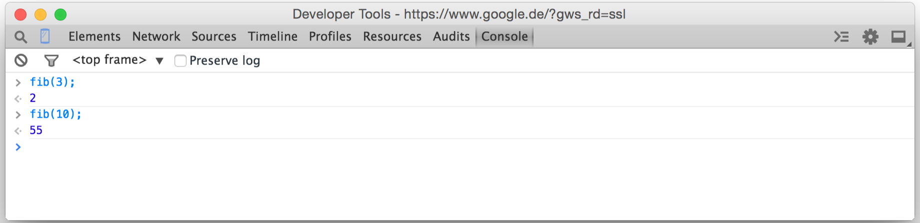
`console.time`, `console.timeEnd`



```
> function fib(n) {
    if (n < 2) return n;
    return fib(n-1) + fib(n-2);
  };
< undefined
>
```

- `console.time` can be used to precisely (sub-millisecond) time certain program flows

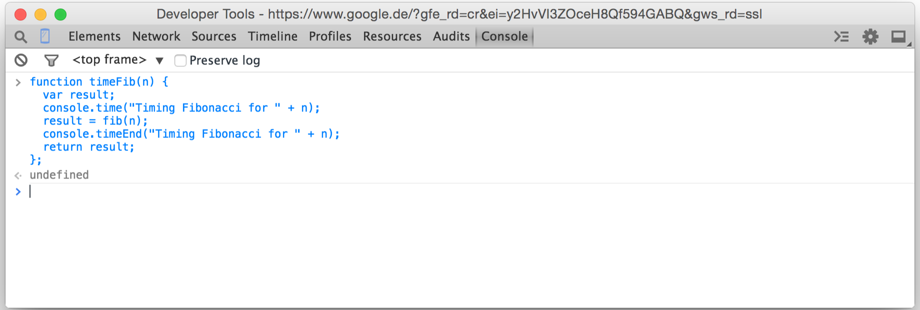
console.time, console.timeEnd


Developer Tools - https://www.google.de/?gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> fib(3);
< 2
> fib(10);
< 55
>
```

- `console.time` can be used to precisely (sub-millisecond) time certain program flows

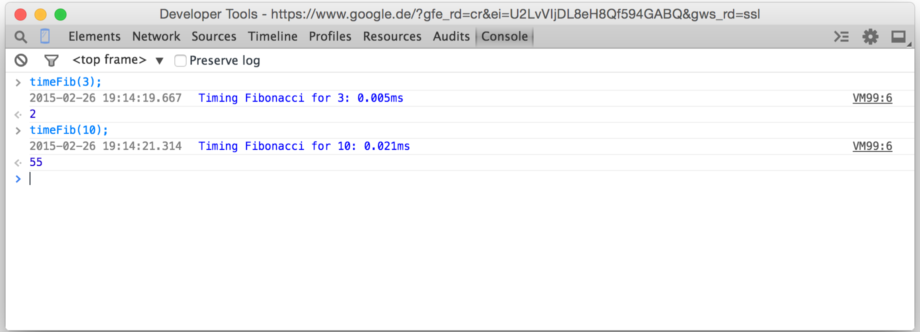
console.time, console.timeEnd


Developer Tools - https://www.google.de/?gfe_rd=cr&ei=y2HvVI3ZOceH8Qf594GABQ&gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> function timeFib(n) {
    var result;
    console.time("Timing Fibonacci for " + n);
    result = fib(n);
    console.timeEnd("Timing Fibonacci for " + n);
    return result;
  };
< undefined
>
```

- `console.time` can be used to precisely (sub-millisecond) time certain program flows

`console.time`, `console.timeEnd`

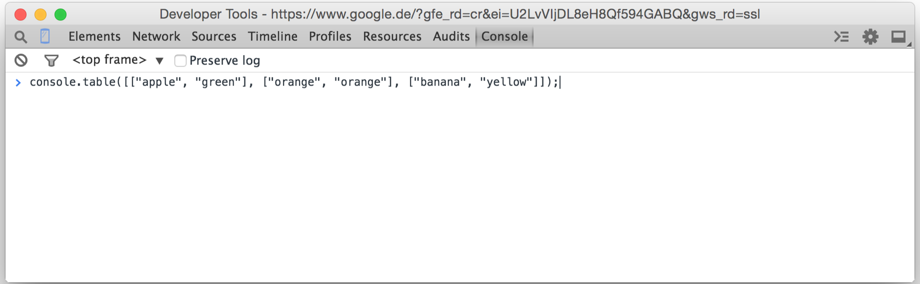

Developer Tools - https://www.google.de/?gfe_rd=cr&ei=U2LvVIjDL8eH8Qf594GABQ&gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> timeFib(3);
  2015-02-26 19:14:19.667   Timing Fibonacci for 3: 0.005ms                VM99:6
< 2
> timeFib(10);
  2015-02-26 19:14:21.314   Timing Fibonacci for 10: 0.021ms               VM99:6
< 55
>
```

- `console.time` can be used to precisely (sub-millisecond) time certain program flows

`console.table`

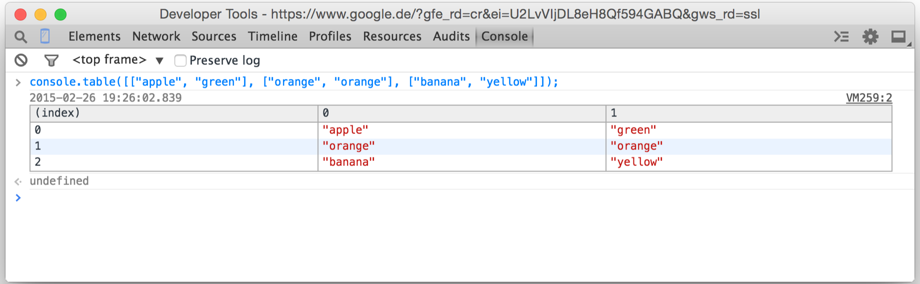
console.table, console.timeEnd


Developer Tools - https://www.google.de/?gfe_rd=cr&ei=U2LvVIjDL8eH8Qf594GABQ&gws_rd=ssl
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
<top frame>
Preserve log

```
> console.table([["apple", "green"], ["orange", "orange"], ["banana", "yellow"]]);
```

- `console.table` can be used to nicely print out all kinds of structures
- `console.table` can print arrays of arrays

`console.table`, `console.timeEnd`


Developer Tools - https://www.google.de/?gfe_rd=cr&ei=U2LvVIjDL8eH8Qf594GABQ&gws_rd=ssl
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
<top frame>
Preserve log

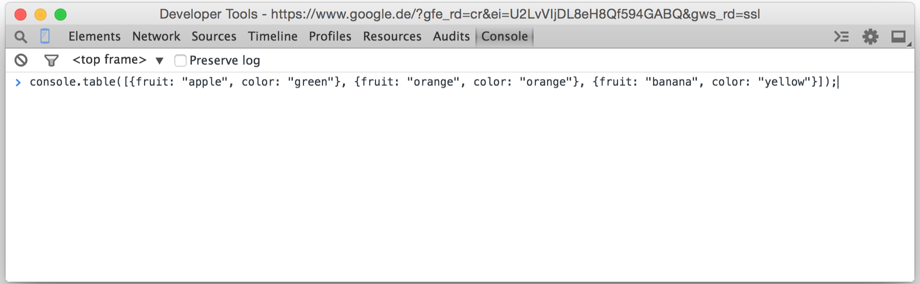
```
> console.table([["apple", "green"], ["orange", "orange"], ["banana", "yellow"]]);
  2015-02-26 19:26:02.839                                                    VM259:2
```

(index)	0	1
0	"apple"	"green"
1	"orange"	"orange"
2	"banana"	"yellow"

```
<· undefined
>
```

- `console.table` can be used to nicely print out all kinds of structures
- `console.table` can print arrays of arrays

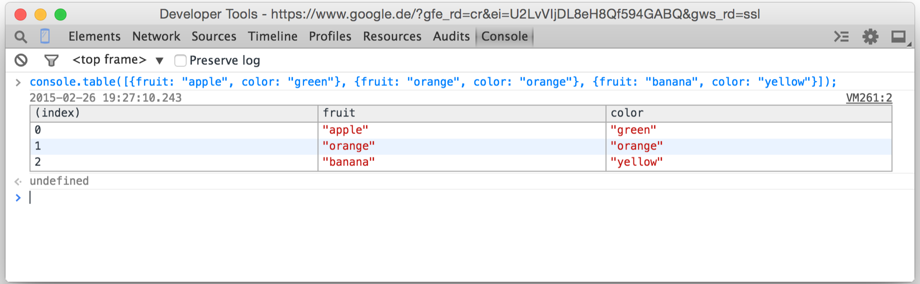
`console.table`, `console.timeEnd`


Developer Tools - https://www.google.de/?gfe_rd=cr&ei=U2LvVIjDL8eH8Qf594GABQ&gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> console.table([{fruit: "apple", color: "green"}, {fruit: "orange", color: "orange"}, {fruit: "banana", color: "yellow"}]);
```

- `console.table` can print arrays of objects

console.table, console.timeEnd


Developer Tools - https://www.google.de/?gfe_rd=cr&ei=U2LvVIjDL8eH8Qf594GABQ&gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

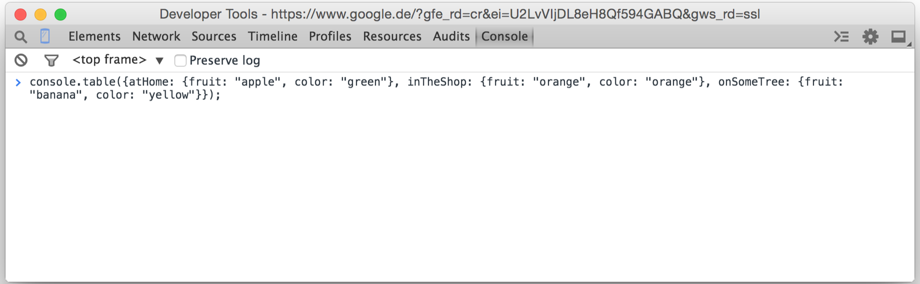
```
> console.table([{fruit: "apple", color: "green"}, {fruit: "orange", color: "orange"}, {fruit: "banana", color: "yellow"}]);
  2015-02-26 19:27:10.243                                                                    VM261:2
```

(index)	fruit	color
0	"apple"	"green"
1	"orange"	"orange"
2	"banana"	"yellow"

```
<· undefined
>
```

- `console.table` can print arrays of objects

console.table, console.timeEnd

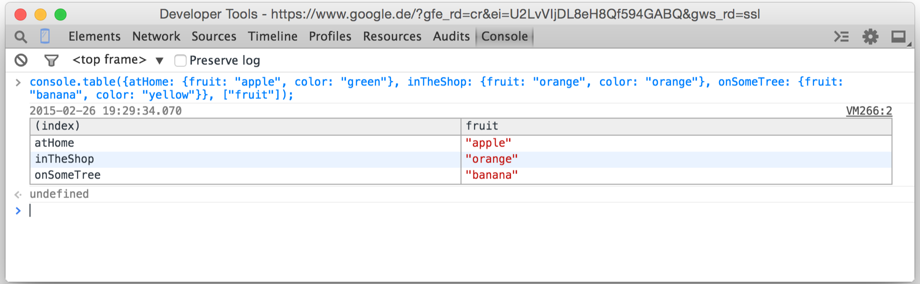


Developer Tools - https://www.google.de/?gfe_rd=cr&ei=U2LvVIjDL8eH8Qf594GABQ&gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> console.table({atHome: {fruit: "apple", color: "green"}, inTheShop: {fruit: "orange", color: "orange"}, onSomeTree: {fruit:
  "banana", color: "yellow"}});
```

- `console.table` can print objects of objects

console.table, console.timeEnd



Developer Tools - https://www.google.de/?gfe_rd=cr&ei=U2LvVIjDL8eH8Qf594GABQ&gws_rd=ssl
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
<top frame>
Preserve log

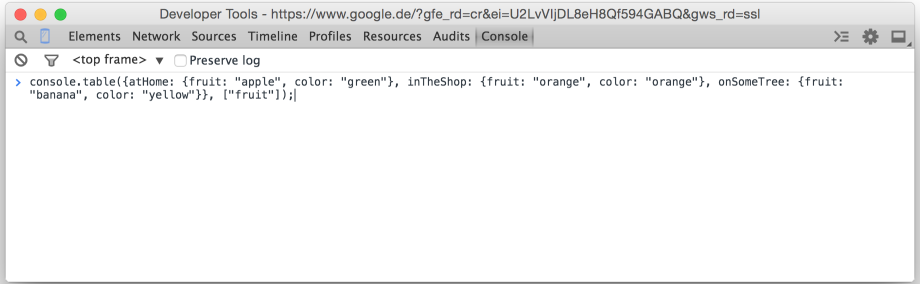
```
> console.table({atHome: {fruit: "apple", color: "green"}, inTheShop: {fruit: "orange", color: "orange"}, onSomeTree: {fruit:
  "banana", color: "yellow"}}, ["fruit"]);
  2015-02-26 19:29:34.070                                                                                                VM266:2
```

(index)	fruit
atHome	"apple"
inTheShop	"orange"
onSomeTree	"banana"

```
<· undefined
>
```

- `console.table` can print objects of objects

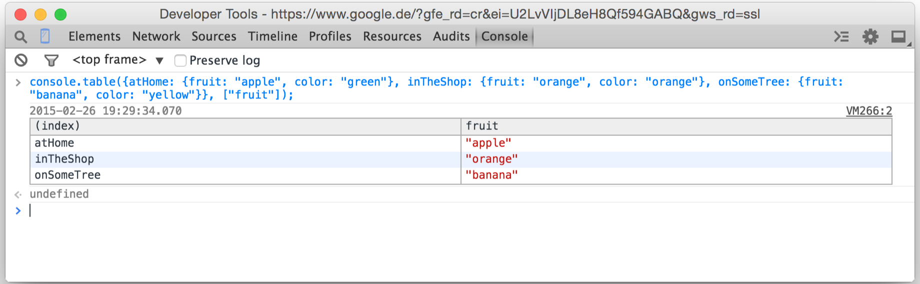
`console.table`, `console.timeEnd`


Developer Tools - https://www.google.de/?gfe_rd=cr&ei=U2LvVIjDL8eH8Qf594GABQ&gws_rd=ssl
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
<top frame>
Preserve log

```
> console.table({atHome: {fruit: "apple", color: "green"}, inTheShop: {fruit: "orange", color: "orange"}, onSomeTree: {fruit:
  "banana", color: "yellow"}}, ["fruit"]);
```

- The second argument can be used to restrict the view to certain columns.

console.table, console.timeEnd


Developer Tools - https://www.google.de/?gfe_rd=cr&ei=U2LvVIjDL8eH8Qf594GABQ&gws_rd=ssl
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log

```
> console.table({atHome: {fruit: "apple", color: "green"}, inTheShop: {fruit: "orange", color: "orange"}, onSomeTree: {fruit:
  "banana", color: "yellow"}}, ["fruit"]);
  2015-02-26 19:29:34.070                                                                                       VM266:2
  (index)                                         fruit
  atHome                                          "apple"
  inTheShop                                       "orange"
  onSomeTree                                      "banana"
< undefined
>
```

- The second argument can be used to restrict the view to certain columns.

Other `console.*` functions

The following Methods will be explained in there corresponding chapter

- `console.profile(...)`
- `console.profileEnd(...)`
- `console.timeStamp(...)`

Playtime

Time to play

1. Change the directory to `Material/01_console/Public`
2. Open up your local webserver there
   - `$ http-server -c-1`
3. Open your Chrome (`http://localhost:8080`)
4. Open the Dev-Tools
5. Play around with the different `console.*` functions
   - CSS Format output with `console.log`
   - Utilize `console.table`
   - Try out `console.count`
   - Play with `console.group`
   - Make use of `console.time`
   - ...

Further Reading

Further Reading: `console.*`

- Google Dev-Tools Console Documentation:
  `https://developer.chrome.com/devtools/docs/console-api`
- Mozilla Developer Network Console Documentation:
  `https://developer.mozilla.org/en-US/docs/Web/API/Console`