E08 – **"Once Upon a Time in Springfield"**

Föreläsning 8, HT2013
Debuggern, Timers

Johan Leitet

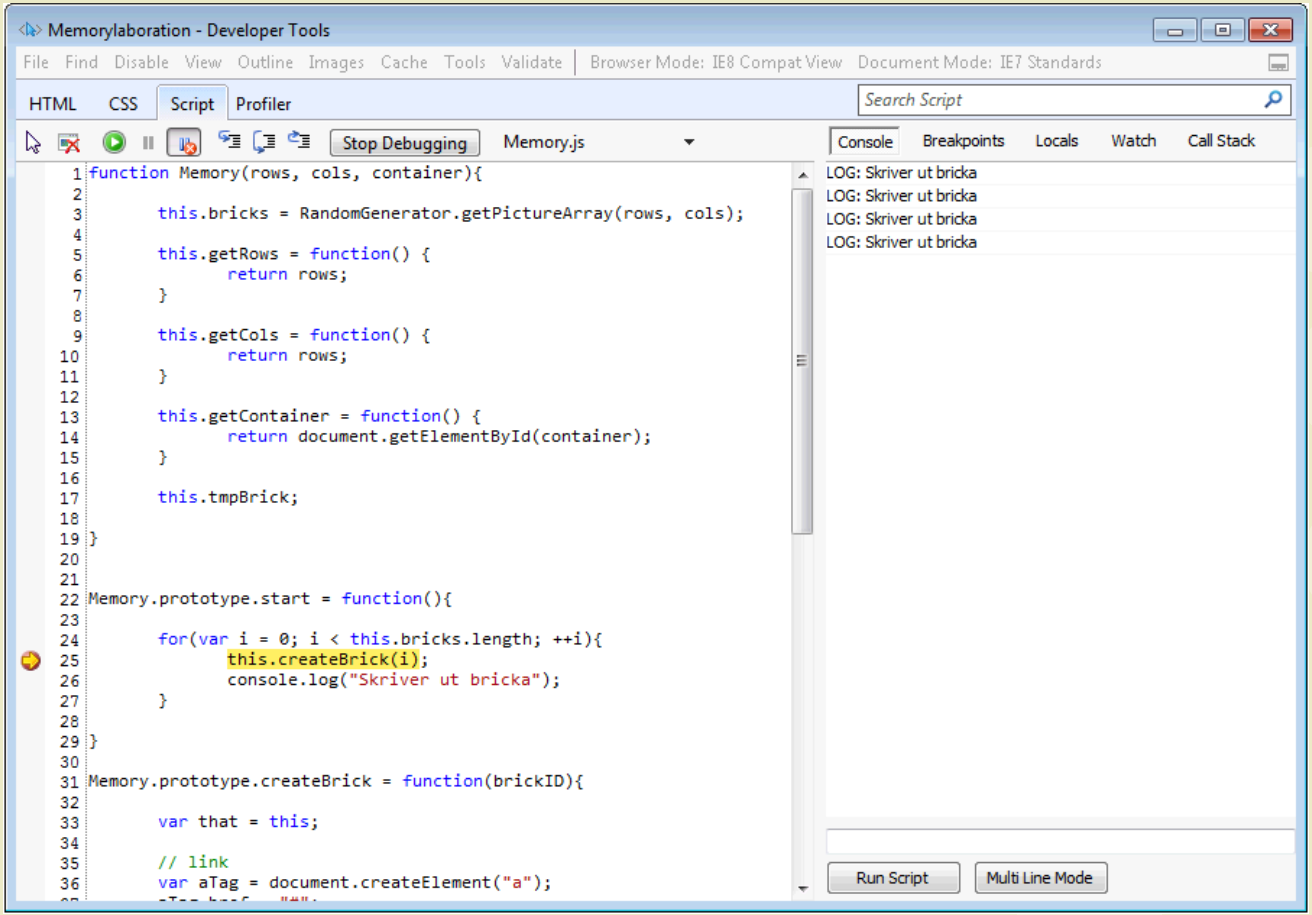
Kurs:
1dv403 Webbteknik I

E08 - Once Upon a Time in Springfield

Dagens agenda

- Konsollen
- Debuggern
- Logga felmeddelanden
- BOM (Browser Objekt Model)
- window-objektet
- Timers
- Intervall
- This, that?

Developer tools




```
function Memory(rows, cols, container){

	this.bricks = RandomGenerator.getPictureArray(rows, cols);

	this.getRows = function() {
		return rows;
	}

	this.getCols = function() {
		return rows;
	}

	this.getContainer = function() {
		return document.getElementById(container);
	}

	this.tmpBrick;

}


Memory.prototype.start = function(){

	for(var i = 0; i < this.bricks.length; ++i){
		this.createBrick(i);
		console.log("Skriver ut bricka");
	}

}

Memory.prototype.createBrick = function(brickID){

	var that = this;

	// link
	var aTag = document.createElement("a");
```

Web inspector

Webbgranskare — lnu.se

Resurser | Tidslinjer | Felsökare | Konsol | Granska | Resurs

Sök i resursinnehåll

- lnu.se
  - Bilder
  - Skript
    - bootstrap-collapse.min.js
    - jquery-1.7.2.min.js
    - jquery-ui-1.8.6.custom.min.js
    - jquery.ajaxmanager.min.js
    - jquery.easing.compatibility.min.js
    - jquery.royalslider.min.js
    - jquery.scrollTo.min.js
    - jwplayer.js
    - modernizr-2.6.2.min.js
    - script-2013-08-26-01.js
    - swfobject.js
    - embed — student.lnu.se
    - sv — student.lnu.se
    - ga.js — www.google-analytics.com
  - Stilmallar
- Cookies — lnu.se
- Lokalt utrymme — lnu.se
- Sessionslagring — lnu.se

Filtrera resurslistor

lnu.se › Skript › jquery.ajaxmanager.min.js

```
(function(c) {
    var l = {}, h = {};
    c.manageAjax = {
        create: function(b, a) {
            l[b] = new c.manageAjax._manager(b, a);
            return l[b]
        },
        destroy: function(b) {
            l[b] && (l[b].clear(!0), delete l[b])
        }
    };
    c.manageAjax._manager = function(b, a) {
        this.requests = {};
        this.inProgress = 0;
        this.qName = this.name = b;
        this.opts = c.extend({}, c.manageAjax.defaults, a);
        a && (a.queue&&!0 !== a.queue && "string" === typeof a.queue && "clear" !== a.queue) && (this.qName = a.queue)
    };
    c.manageAjax._manager.prototype = {
        add: function(b, a) {
            "object" == typeof b ? a = b : "string" == typeof b && (a =
            c.extend(a || {}, {
                url: b
            }));
            a = c.extend({}, this.opts, a);
            var g = a.complete || c.noop, f = a.success || c.noop, d = a.beforeSend || c.noop, e = a.error || c.noop, h =
"string" == typeof a.data ? a.data: c.param(a.data || {}), j = a.type + a.url + h, k = this, m = this._createAjax(j, a, f, g);
            a.preventDoubleRequests && a.queueDuplicateRequests && (a.preventDoubleRequests && (a.queueDuplicateRequests=!1),
setTimeout(function() {
                throw "preventDoubleRequests and queueDuplicateRequests can't be both true";
            }, 0));
            if (!this.requests[j] ||!a.preventDoubleRequests)
                return m.xhrID =
                j, a.xhrID = j, a.beforeSend = function(a, b) {
                var c = d.call(this, a, b);
                !1 === c && k._removeXHR(j);
                return c
            }, a.complete = function(b, c) {
                k._complete.call(k, this, g, b, c, j, a)
            }, a.success = function(b, c, d) {
                k._success.call(k, this, f, b, c, d, a)
            }, a.error = function(b, c, d) {
                var g = "", f = "";
                "timeout" !== c && b && (g = b.status, f = b.responseXML || b.responseText);
                e ? e.call(this, b, c, d, a) : setTimeout(function() {
                    throw c + "| status: " + g + " | URL: " + a.url + " | data: " + h + " | thrown: " + d + " | response: " +
f;
                }, 0);
                b = null
            }, "clear" === a.queue && c(document).clearQueue(this.qName),
            a.queue || a.queueDuplicateRequests && this.requests[j] ? (c.queue(document, this.qName, m), this.inProgress <
a.maxRequests && (!this.requests[j] ||!a.queueDuplicateRequests) && c.dequeue(document, this.qName), j) : m()
        },
        _createAjax: function(b, a, g, f) {
            var d = this;
            return function() {
                if (!1 !== a.beforeCreate.call(a.context || d, b, a)) {
                    d.inProgress++;
                    1 === d.inProgress && c.event.trigger(d.name + "AjaxStart");
                    a.cacheResponse && h[b] && (!h[b].cacheTTL || 0 > h[b].cacheTTL || (new Date).getTime() - h[b].timestamp <
h[b].cacheTTL ? (d.requests[b] = {},
```

Typ
- MIME-typ: text/javascript
- Resurstyp: Skript

Plats
- Full URL: http://lnu.se/js/jquery.ajaxmanager.min.js
- Schema: http
- Värd: lnu.se
- Sökväg: /js/jquery.ajaxmanager.min.js
- Filnamn: jquery.ajaxmanager.min.js

Begäran och svar

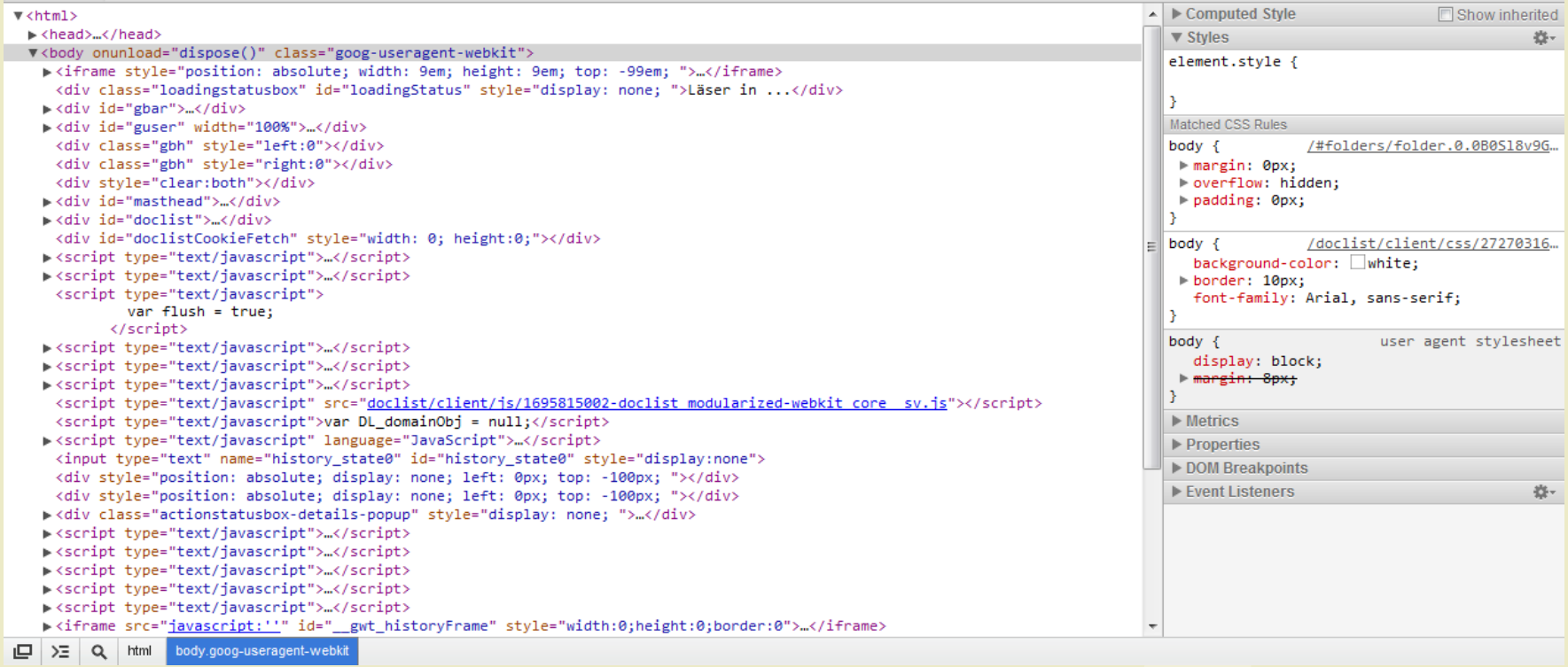
Begärans rubriker

Inga rubriker i begäran

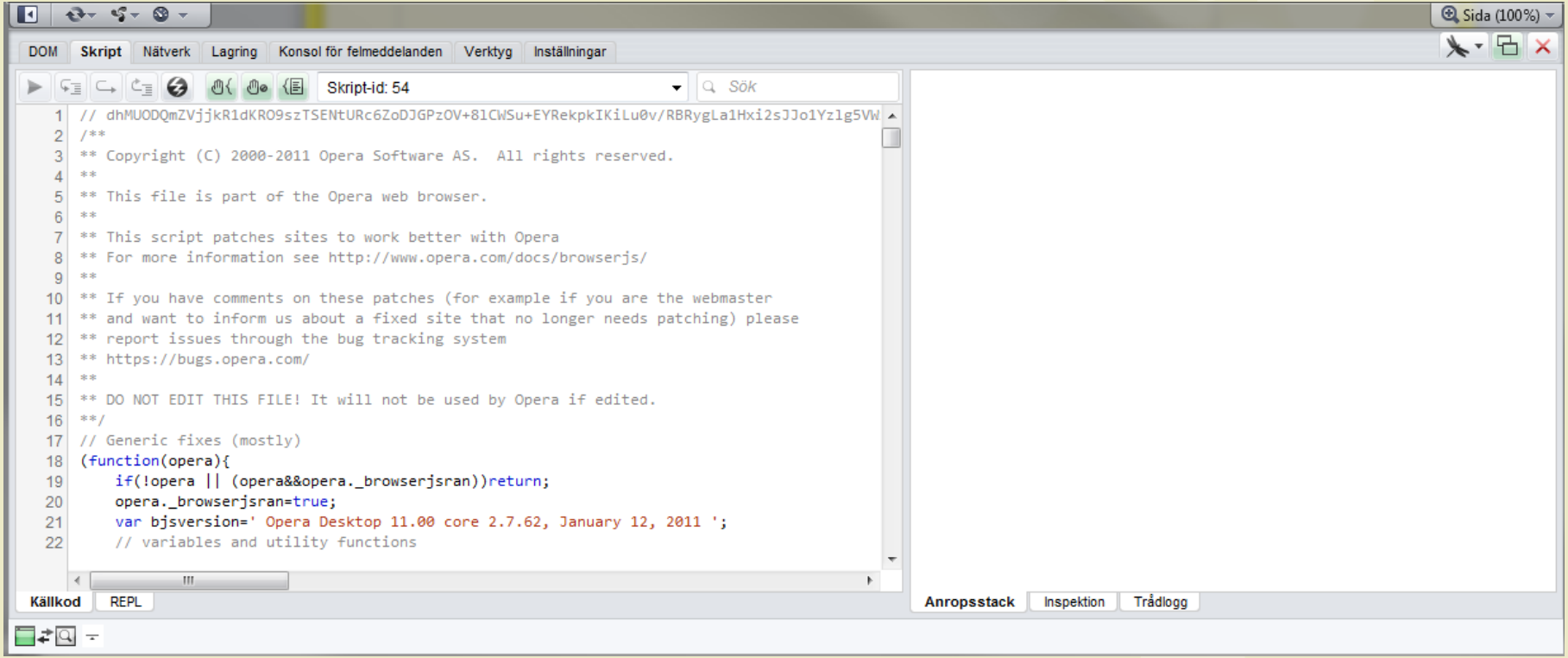
Svarets rubriker

Inga rubriker i svaret

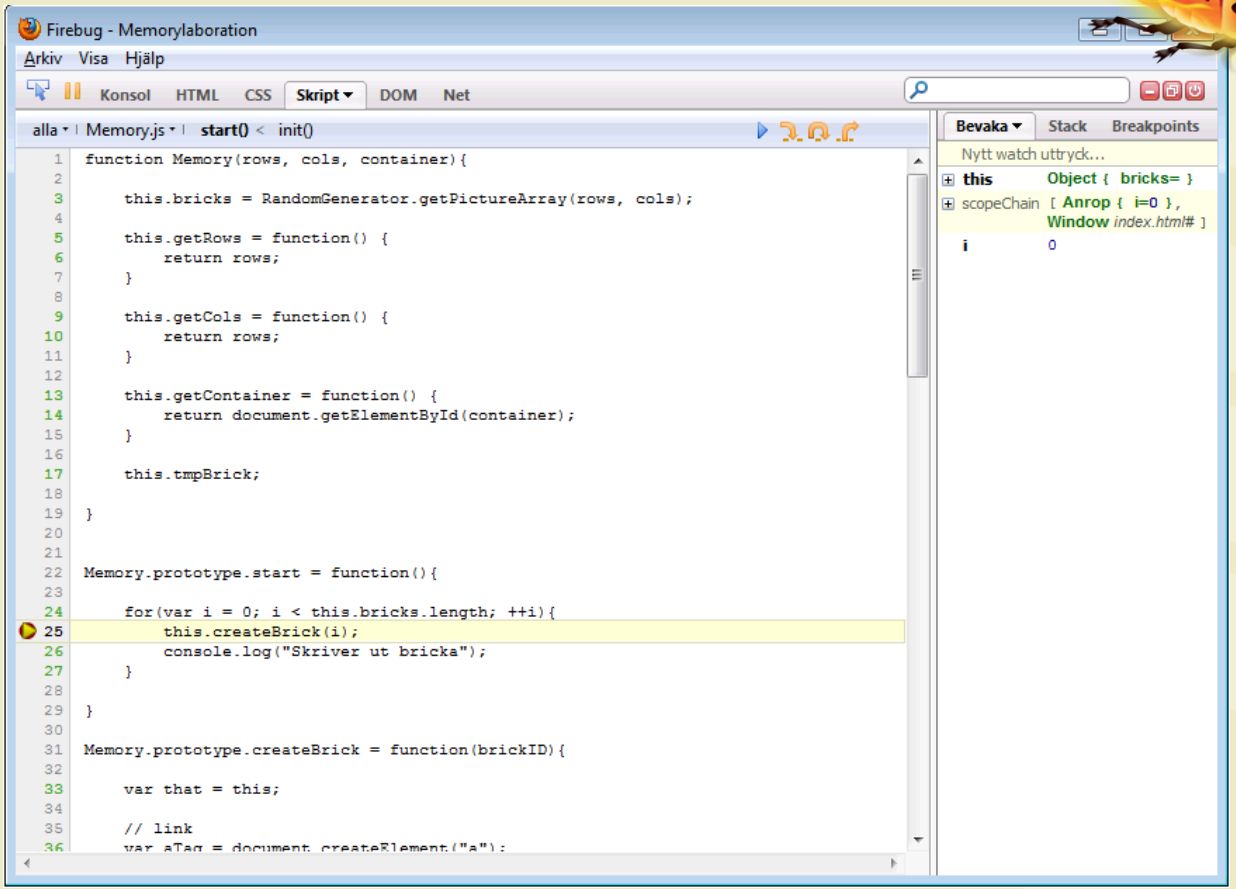
Developer tools



Dragonfly

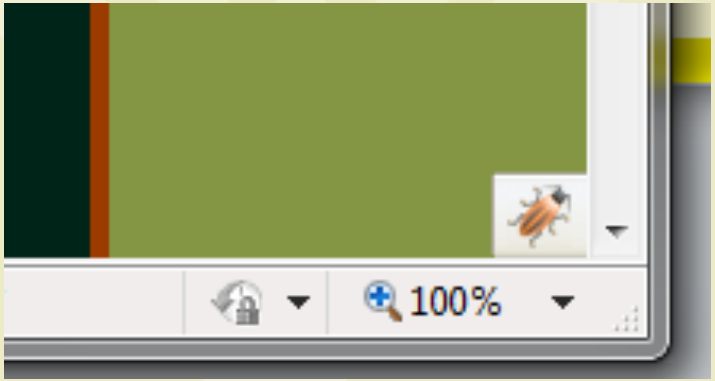

Sida (100%)
DOM
Skript
Nätverk
Lagring
Konsol för felmeddelanden
Verktyg
Inställningar
Skript-id: 54
Sök
1 // dhMUODQmZVjjkR1dKRO9szTSENtURc6ZoDJGPzOV+81CWSu+EYRekpkIKiLu0v/RBRygLa1Hxi2sJJo1Yzlg5VW
2 /**
3 ** Copyright (C) 2000-2011 Opera Software AS. All rights reserved.
4 **
5 ** This file is part of the Opera web browser.
6 **
7 ** This script patches sites to work better with Opera
8 ** For more information see http://www.opera.com/docs/browserjs/
9 **
10 ** If you have comments on these patches (for example if you are the webmaster
11 ** and want to inform us about a fixed site that no longer needs patching) please
12 ** report issues through the bug tracking system
13 ** https://bugs.opera.com/
14 **
15 ** DO NOT EDIT THIS FILE! It will not be used by Opera if edited.
16 **/
17 // Generic fixes (mostly)
18 (function(opera){
19 if(!opera || (opera&&opera._browserjsran))return;
20 opera._browserjsran=true;
21 var bjsversion=' Opera Desktop 11.00 core 2.7.62, January 12, 2011 ';
22 // variables and utility functions
Källkod
REPL
Anropsstack
Inspektion
Trådlogg

Firebug


Firebug - Memorylaboration
Arkiv Visa Hjälp
Konsol HTML CSS Skript DOM Net
alla Memory.js start() < init()
Bevaka Stack Breakpoints
Nytt watch uttryck...
this Object { bricks= }
scopeChain [Anrop { i=0 }, Window index.html#]
i 0
1 function Memory(rows, cols, container){
2
3 this.bricks = RandomGenerator.getPictureArray(rows, cols);
4
5 this.getRows = function() {
6 return rows;
7 }
8
9 this.getCols = function() {
10 return rows;
11 }
12
13 this.getContainer = function() {
14 return document.getElementById(container);
15 }
16
17 this.tmpBrick;
18
19 }
20
21
22 Memory.prototype.start = function(){
23
24 for(var i = 0; i < this.bricks.length; ++i){
25 this.createBrick(i);
26 console.log("Skriver ut bricka");
27 }
28
29 }
30
31 Memory.prototype.createBrick = function(brickID){
32
33 var that = this;
34
35 // link
36 var aTag = document.createElement("a");

Firebug Lite

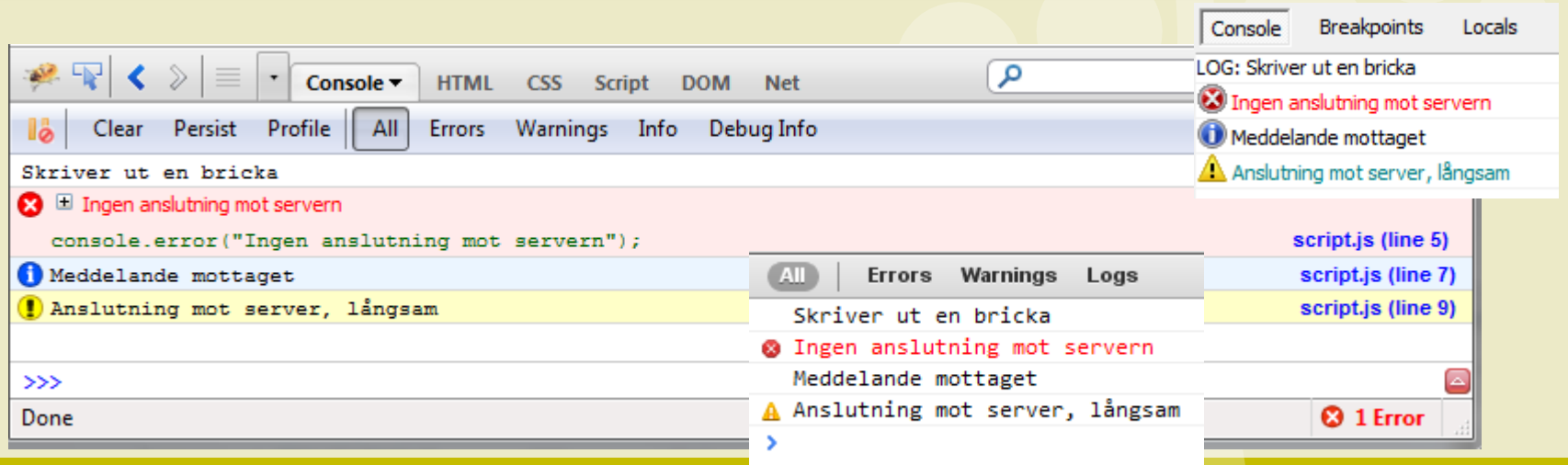
```
    <link rel="stylesheet" type="text/css" href="style.css" />

    <script type="text/javascript" src="https://getfirebug.com/firebug-lite.js"></script>

</head>
```



Logga till console

Vi har tillgång till ett objekt, console, som vi kan använda för att skriva till debuggerns konsolfönster. *(FireFox (FireBug), Internet Explorer, Safari, Chrome)*

```
console.log("Skriver ut en bricka");
console.error("Ingen anslutning mot servern");
console.info("Meddelande mottaget");
console.warn("Anslutning mot server, långsam");
```



DOM och BOM

DOM

Document Object Model

BOM

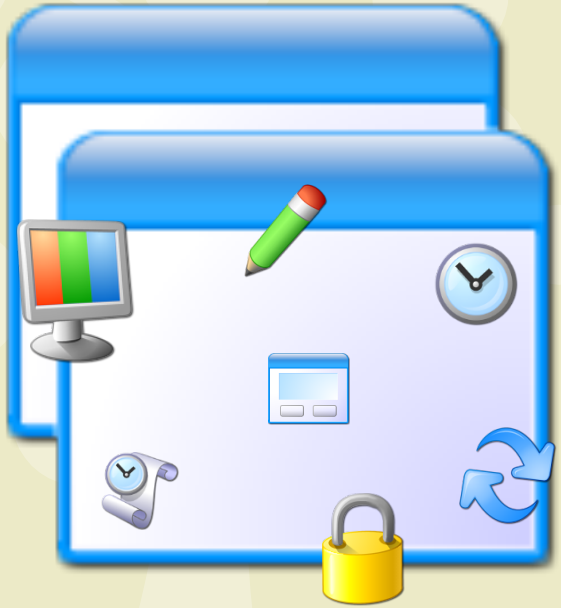
Browser Object Model

BOM

BOM (Browser Object Model) är gränssnittet mellan JavaScript och webbläsaren.

BOM är inte standardiserat.

Objektet **window** är centralt.



BOM hanterar

Insorterat under denna rubrik hittar vi:

Timers och intervall

Webbläsarfönster (och ramar, frames)
- Positioner
- Storlekar

Systemdialoger (alert, prompt, confirm)

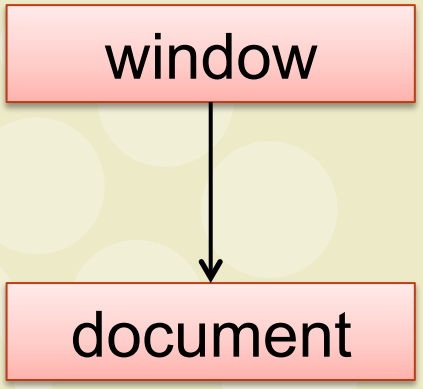
Location (adressfält)

Historik

...

window

`window`-objektet representerar en instans av webbläsarfönstret och motsvarar i webbläsaren det objekt som är **`Global`** i ECMAScript.



Alla globala variabler hamnar således på just window-objektet

```
var todo = "Go to work, not!";
alert(todo);          // Go to work, not!
alert(window.todo); // Go to work, not!
```

Timers

Två typer av timers i webbläsaren:

Timeout

Intervall

setTimeout

setTimeout kan vi använda när vi vill vänta och sedan utföra något.

```
setTimeout(myApp.goToSchool, 3000);
```

ms

```
setTimeout(function(){
    myApp.goToSchool();
}, 3000);
```

~~`setTimeout("myApp.goToSchool()", 3000);`~~

setTimeout ligger på window-objektet men eftersom detta är globalt behöver vi inte skriva window.setTimeout, men vi kan.

setInterval

setTimeout kan vi använda när vi vill vänta och sedan utföra något.

```
setInterval(myApp.writeOnBlackboard, 3000);
```

ms

```
setInterval(function(){

   myApp.writeOnBlackboard("I will not use inline JS in my HTML-pages.");

}, 3000);
```

När väl ett intervall startat så slutar det inte förrän man säger till det att stoppa. (Vilket kan innebära vissa problem, så kan man bör man undvika setInterval och förlita sig på setTimeout.)

clearInterval

Genom att spara undan ett id som returneras från setInterval så kan vi stoppa timern när vi önskar.

```
var timerID = setInterval(function(){

   myApp.writeOnBlackboard("I will not use inline JS in my HTML-pages.");

   if(myApp.isBlackboardFilled()){

        clearInterval(timerID);

   }

}, 3000);
```

På samma sätt fungerar metoden **clearTimeout**

```
function Homer(){
    var node = document.getElementById("belly");
    var that = this;

    // this?

    node.onclick = function(){

        // this?
        // that?

        setTimeout(function(){

            // this?
            // that?
        }, 1000);
    };
}
var h1 = new Homer();
var h2 = new Homer();
```

This?
That?

Douglas Crockford can travel back in time with a negative setTimeout

Källa: http://twitter.com/crockfordfacts