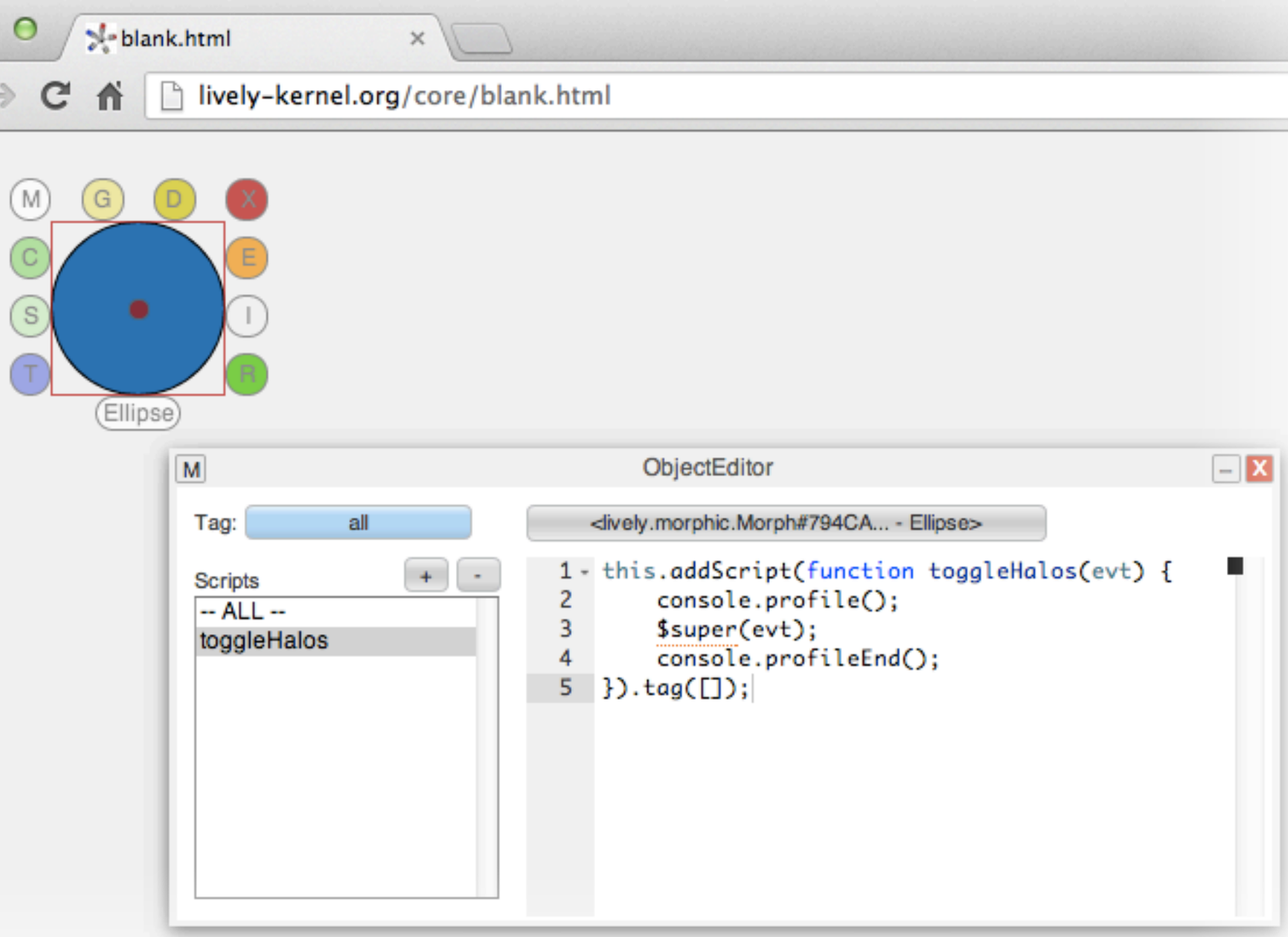
JavaScript / v8 Profiling

Lauritz Thamsen

Software Profiling Seminar
June 24, 2013

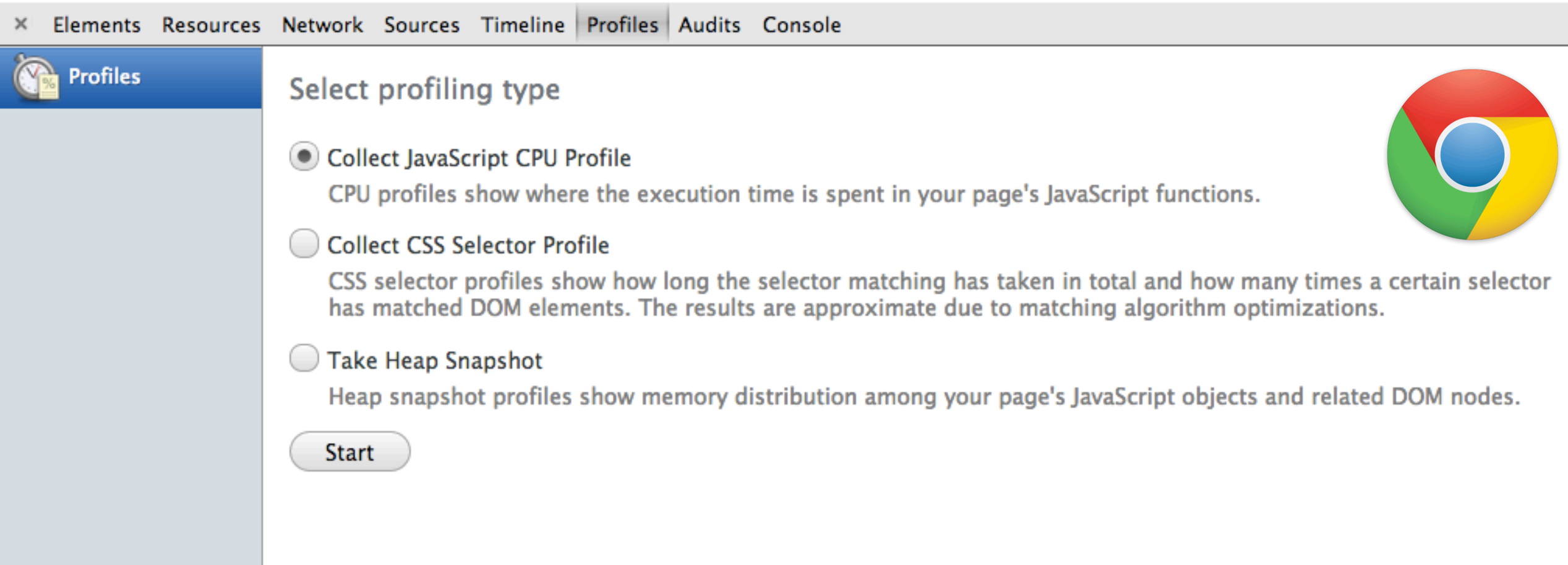
JavaScript / v8 Profiling

- CPU Profiling / Chrome
- Built-in v8 Profiler
- Custom Tracing
- Memory Profiling / Chrome
- Report Outlook



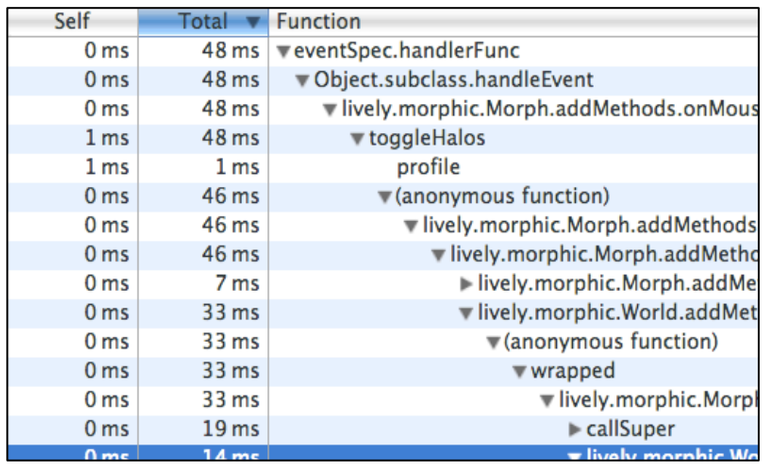
examples: Lively Kernel

CPU Profiling / Chrome



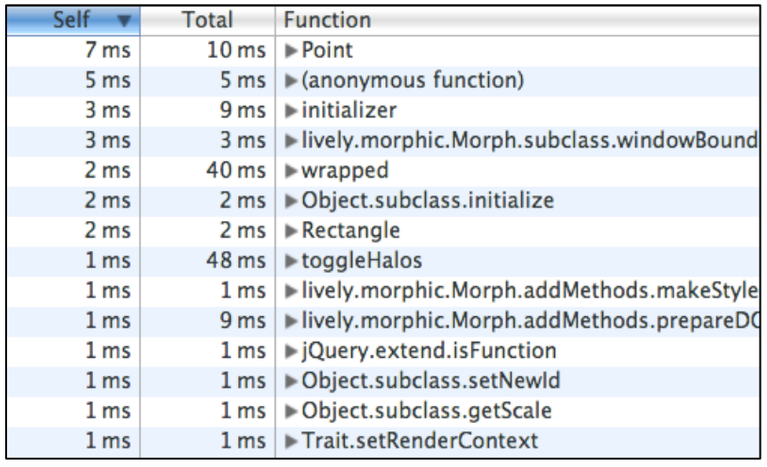
Chrome Developer Tools: CPU Profiling

- shows where time is statistically spent
- based on v8's built-in sampling profiler



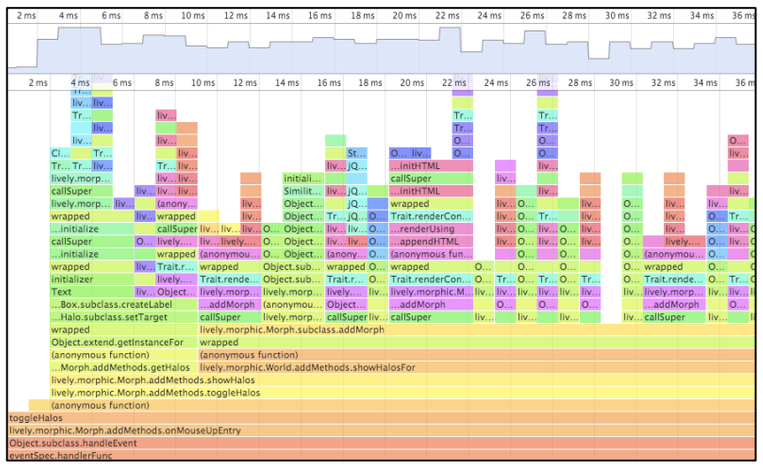
Self	Total	Function
0 ms	48 ms	▼eventSpec.handlerFunc
0 ms	48 ms	▼Object.subclass.handleEvent
0 ms	48 ms	▼lively.morphic.Morph.addMethods.onMous
1 ms	48 ms	▼toggleHalos
1 ms	1 ms	profile
0 ms	46 ms	▼(anonymous function)
0 ms	46 ms	▼lively.morphic.Morph.addMethods
0 ms	46 ms	▼lively.morphic.Morph.addMetho
0 ms	7 ms	▶lively.morphic.Morph.addMe
0 ms	33 ms	▼lively.morphic.World.addMet
0 ms	33 ms	▼(anonymous function)
0 ms	33 ms	▼wrapped
0 ms	33 ms	▼lively.morphic.Morpl
0 ms	19 ms	▶callSuper

Tree
(Top Down)



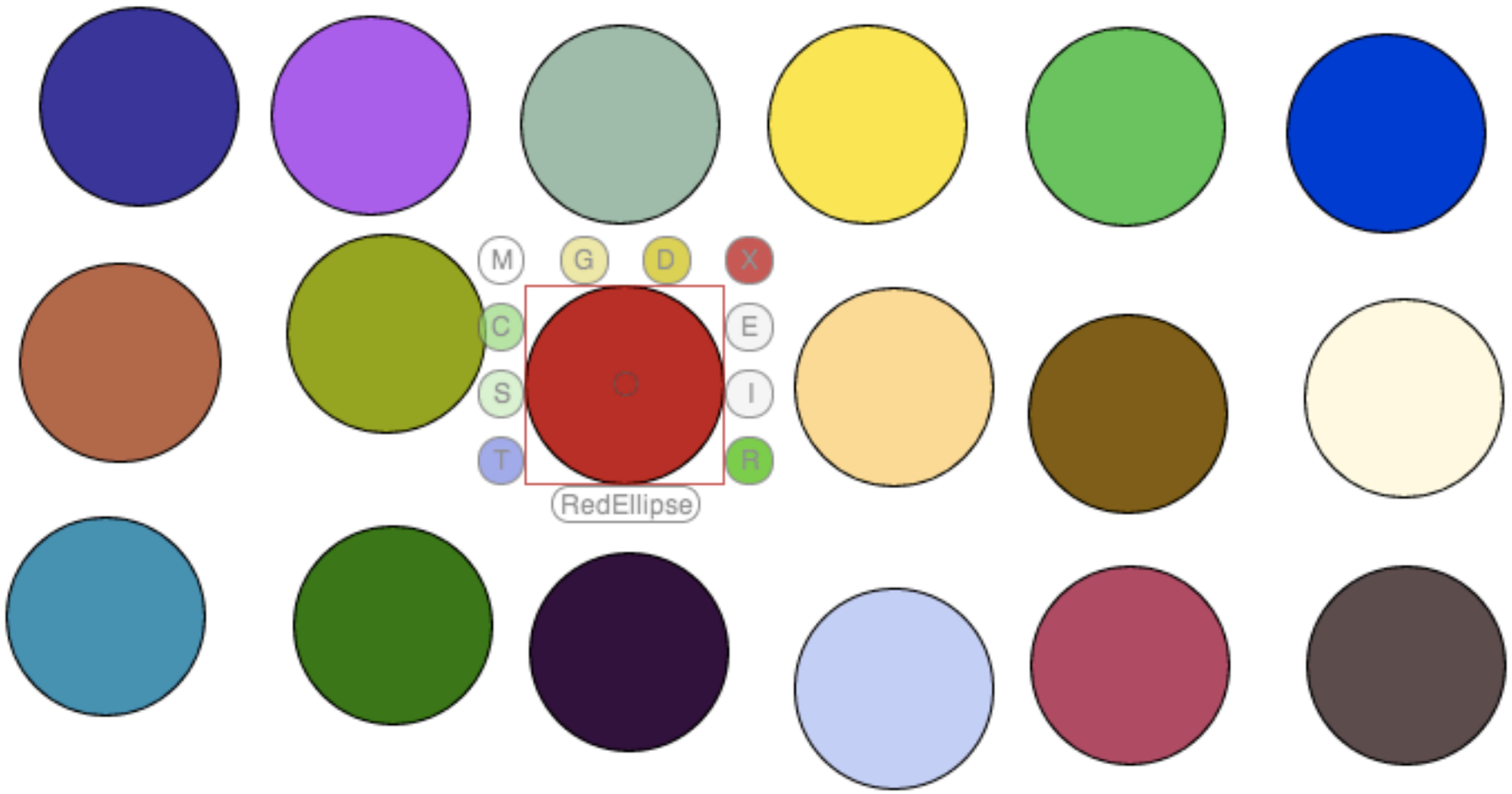
Self	Total	Function
7 ms	10 ms	▶Point
5 ms	5 ms	▶(anonymous function)
3 ms	9 ms	▶initializer
3 ms	3 ms	▶lively.morphic.Morph.subclass.windowBound
2 ms	40 ms	▶wrapped
2 ms	2 ms	▶Object.subclass.initialize
2 ms	2 ms	▶Rectangle
1 ms	48 ms	▶toggleHalos
1 ms	1 ms	▶lively.morphic.Morph.addMethods.makeStyle
1 ms	9 ms	▶lively.morphic.Morph.addMethods.prepareDC
1 ms	1 ms	▶jQuery.extend.isFunction
1 ms	1 ms	▶Object.subclass.setNewId
1 ms	1 ms	▶Object.subclass.getScale
1 ms	1 ms	▶Trait.setRenderContext

Heavy
(Bottom Up)



Flame Chart

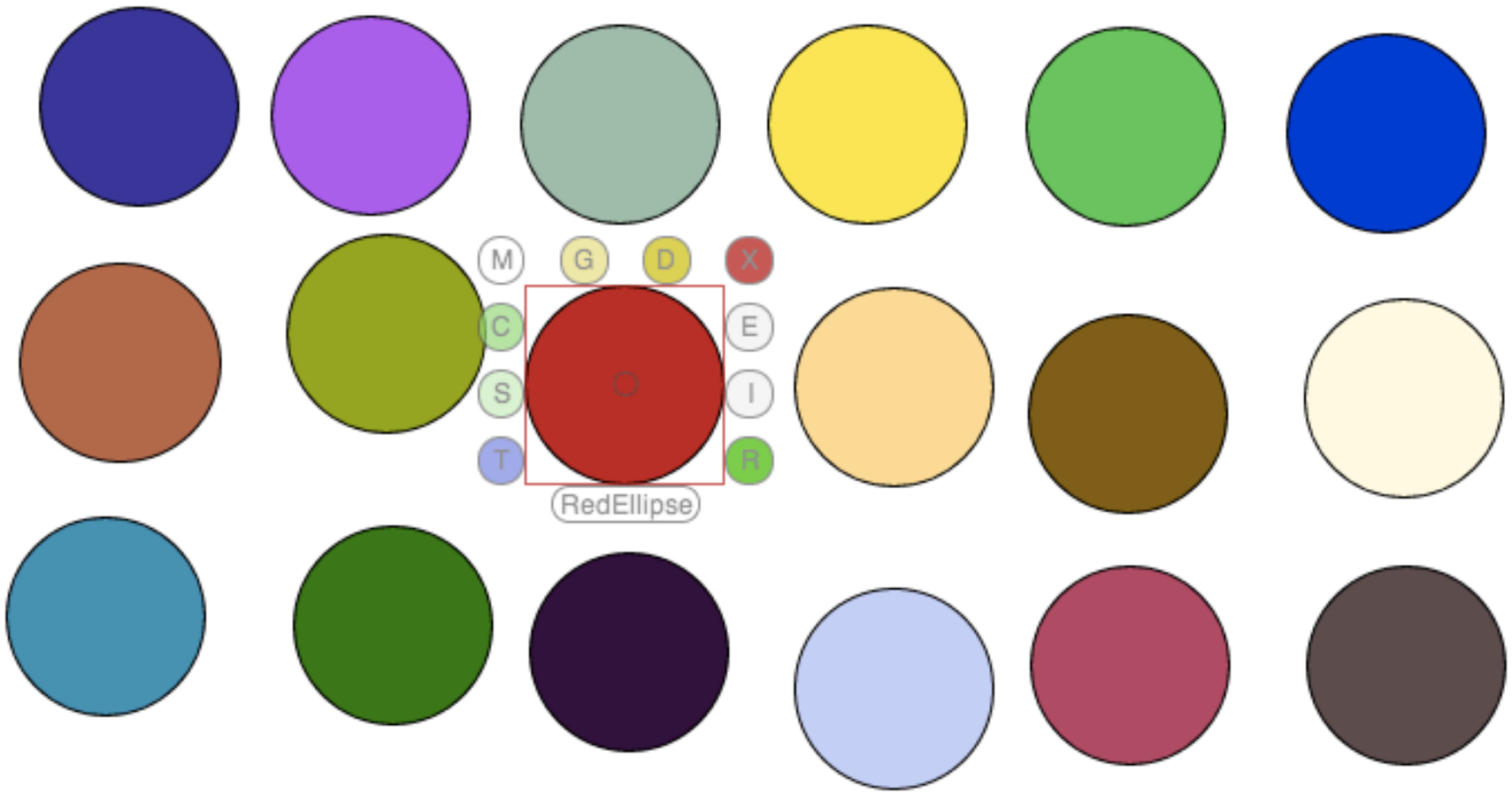
Issue #342: ‘Halo Performance’



#342 on 3/26/12, at **version 147913**

“Opening halos in a world with many morphs is really slow. This seems to be related to the *getTopmostMorph* calls..”

Issue #342: ‘Halo Performance’

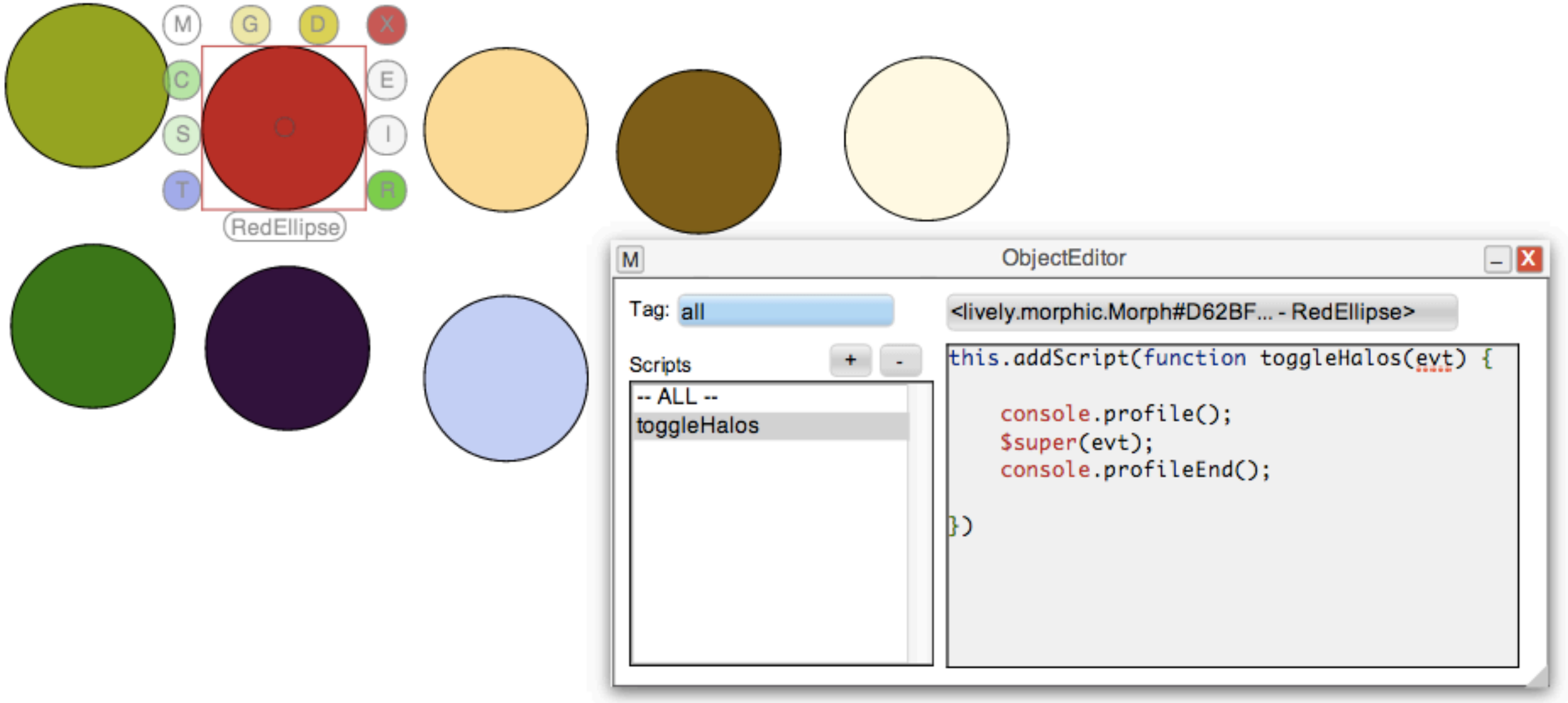


#342 on 3/26/12, at **version 147913**

“Opening halos in a world with many morphs is really slow. This seems to be related to the

today at roughly r196426
no longer a problem

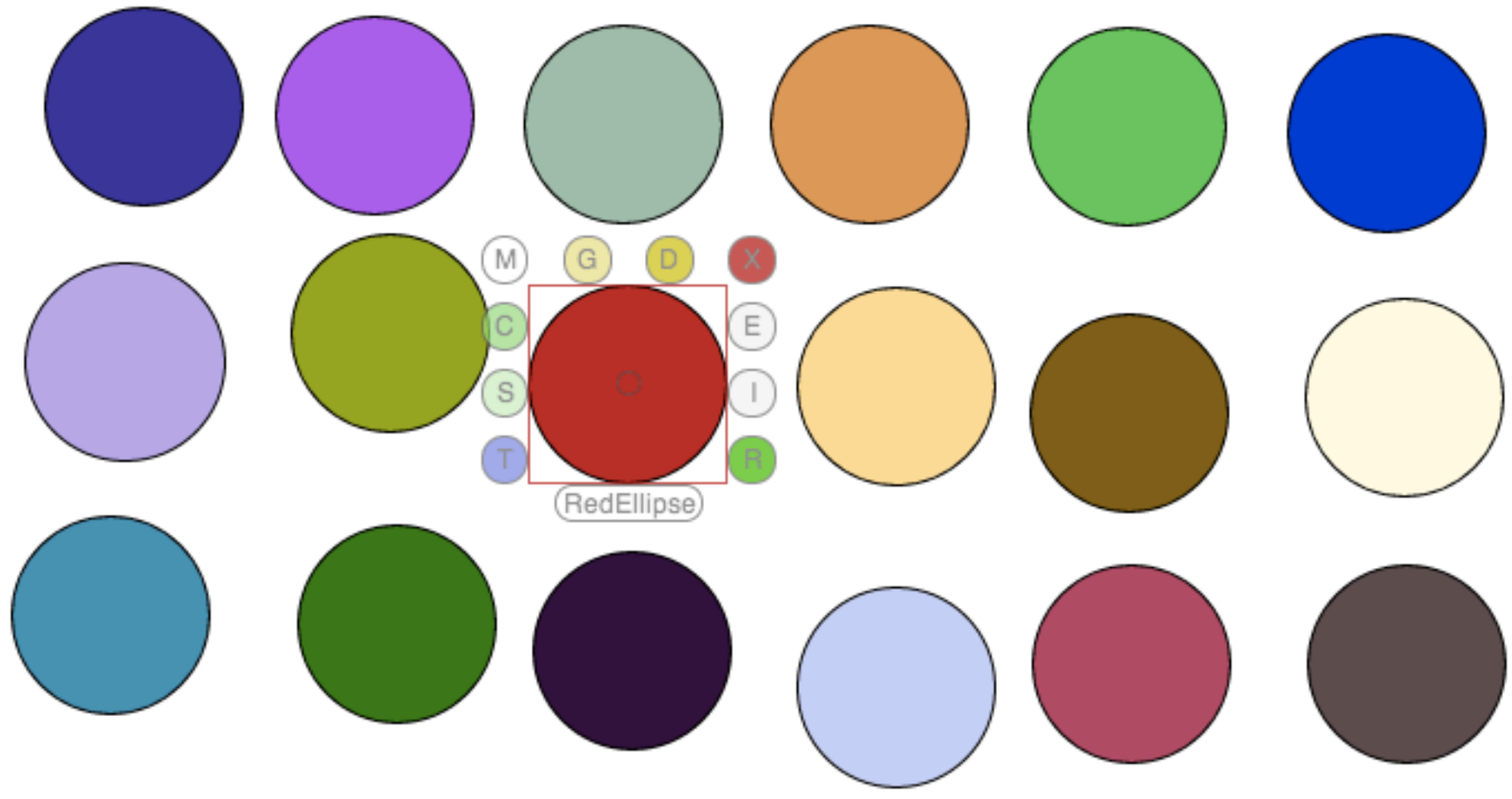
Issue #342, revision 147913



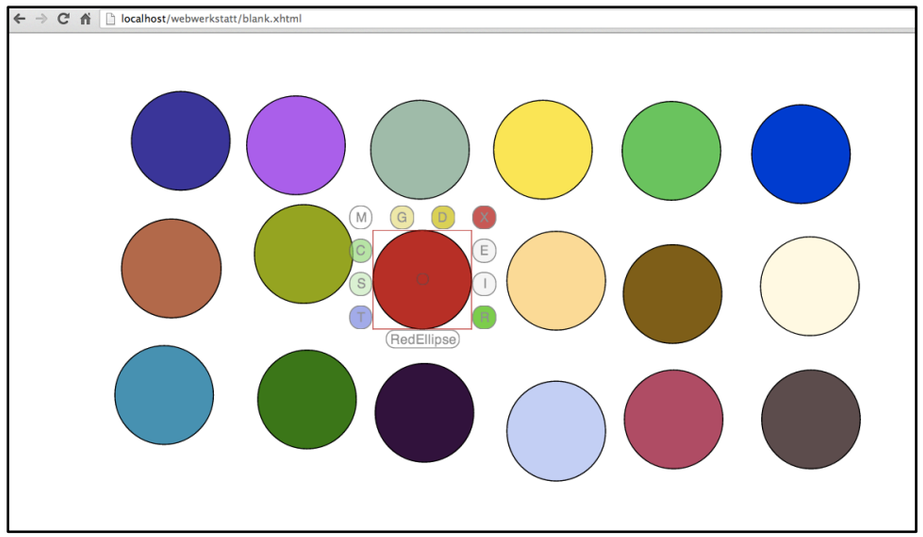
- console api
  - `console.profile([label]), console.profileEnd()`
  - `console.time(label), console.timeEnd(label)`
  - `console.count(label)`
  - `console.trace()`

Issue #342, revision 147913

- toggling halos on one out of **18 morphs**



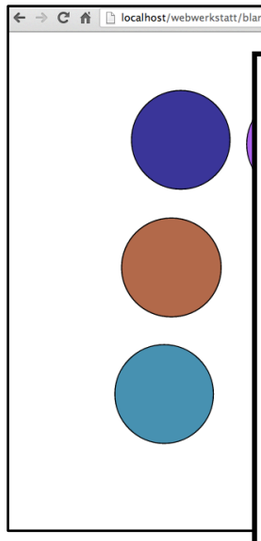
Issue #342, revision 147913



- toggling halos on one out of **18 morphs**

Self	Total	Function	
0 ms	34 ms	▼eventSpec.handlerFunc	Events.js?1371896489576:108
0 ms	34 ms	▼Object.subclass.handleEvent	Events.js?1371896489576:151
0 ms	34 ms	▼lively.morphic.Morph.addMethods.onMouseUpEntry	Events.js?1371896489576:716
1 ms	34 ms	▼toggleHalos	
0 ms	32 ms	▼(anonymous function)	
0 ms	32 ms	▼lively.morphic.Morph.addMethods.toggleHalos	Halos.js?1371896489866:60
0 ms	32 ms	▼lively.morphic.Morph.addMethods.showHalos	Halos.js?1371896489866:8
0 ms	20 ms	▼lively.morphic.World.addMethods.showHalosFor	Halos.js?1371896489866:78
0 ms	20 ms	▼(anonymous function)	Halos.js?1371896489866:83
0 ms	20 ms	▼wrapped	Function.js?1371896489374:37
0 ms	20 ms	▼lively.morphic.Morph.subclass.addMorph	Core.js?1371896489525:598
0 ms	13 ms	▶callSuper	Base.js?1371896489415:376
0 ms	7 ms	▼lively.morphic.World.addMethods.updateScrollFocus	Events.js?1371896489576:1683
0 ms	7 ms	▼(anonymous function)	Events.js?1371896489576:1685
0 ms	7 ms	▼lively.morphic.Morph.addMethods.getTopmostMorph	Events.js?1371896489576:1086
0 ms	7 ms	▼Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	4 ms	▶Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	3 ms	▶Object.subclass.fullContainsWorldPoint	Core.js?1371896489525:404

Issue #342, revision 147913

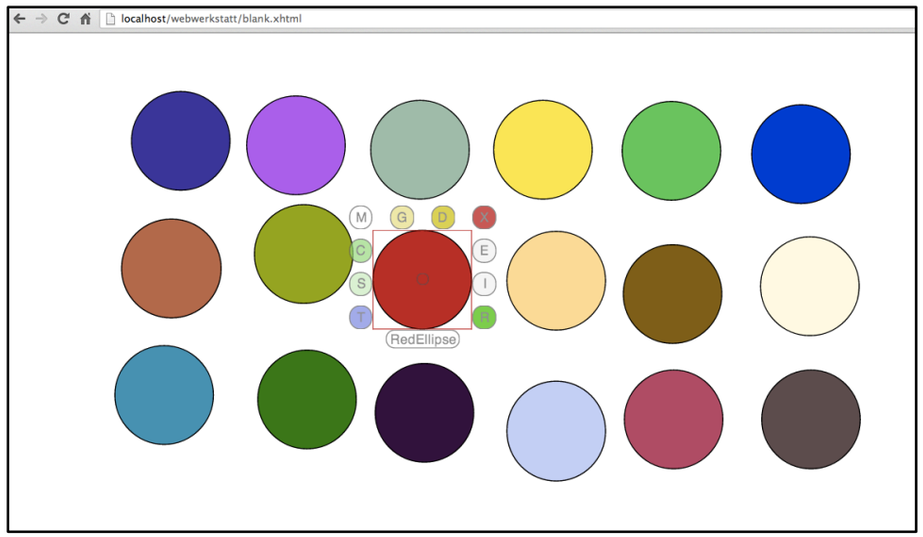


```
lively.morphic.World.addMethods(
'halos', {
    showHalosFor: function(morph, halos) {
        if (this.currentHaloTarget) {
            this.currentHaloTarget.removeHalos(this);
        }
        this.currentHaloTarget = morph;
        halos.forEach(function(halo) { this.addMorph(halo); }, this);
    }
});
```

```
function wrapped() {
    ...
}
```

Self	Total	Function	
0 ms	34 ms	▼eventSpec.handlerFunc	Events.js?1371896489576:108
0 ms	34 ms	▼Object.subclass.handleEvent	Events.js?1371896489576:151
0 ms	34 ms	▼lively.morphic.Morph.addMethods.onMouseUpEntry	Events.js?1371896489576:716
1 ms	34 ms	▼toggleHalos	
0 ms	32 ms	▼(anonymous function)	
0 ms	32 ms	▼lively.morphic.Morph.addMethods.toggleHalos	Halos.js?1371896489866:60
0 ms	32 ms	▼lively.morphic.Morph.addMethods.showHalos	Halos.js?1371896489866:8
0 ms	20 ms	▼lively.morphic.World.addMethods.showHalosFor	Halos.js?1371896489866:78
0 ms	20 ms	▼(anonymous function)	Halos.js?1371896489866:83
0 ms	20 ms	▼wrapped	Function.js?1371896489374:37
0 ms	20 ms	▼lively.morphic.Morph.subclass.addMorph	Core.js?1371896489525:598
0 ms	13 ms	▶callSuper	Base.js?1371896489415:376
0 ms	7 ms	▼lively.morphic.World.addMethods.updateScrollFocus	Events.js?1371896489576:1683
0 ms	7 ms	▼(anonymous function)	Events.js?1371896489576:1685
0 ms	7 ms	▼lively.morphic.Morph.addMethods.getTopmostMorph	Events.js?1371896489576:1086
0 ms	7 ms	▼Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	4 ms	▶Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	3 ms	▶Object.subclass.fullContainsWorldPoint	Core.js?1371896489525:404

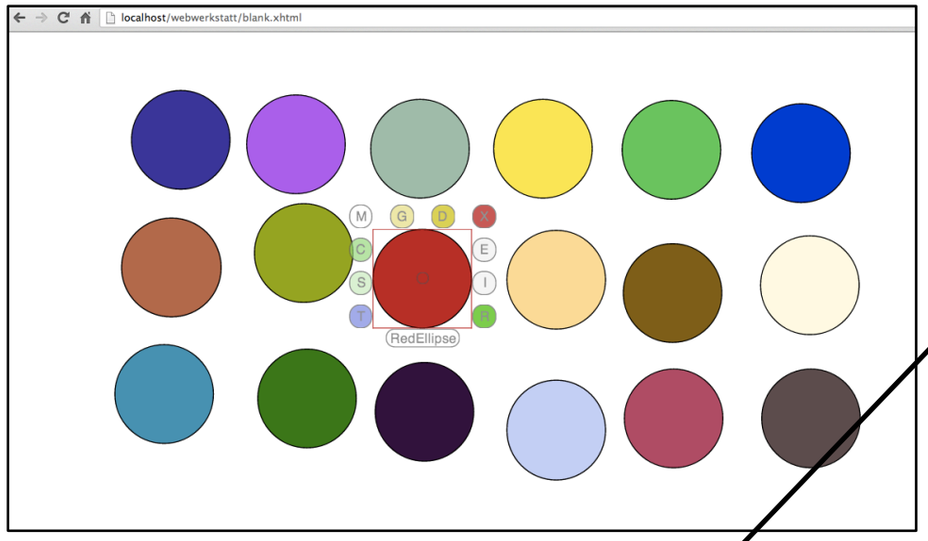
Issue #342, revision 147913



- toggling halos on one out of **18 morphs**

Self	Total	Function	
0 ms	34 ms	▼eventSpec.handlerFunc	Events.js?1371896489576:108
0 ms	34 ms	▼Object.subclass.handleEvent	Events.js?1371896489576:151
0 ms	34 ms	▼lively.morphic.Morph.addMethods.onMouseUpEntry	Events.js?1371896489576:716
1 ms	34 ms	▼toggleHalos	
0 ms	32 ms	▼(anonymous function)	
0 ms	32 ms	▼lively.morphic.Morph.addMethods.toggleHalos	Halos.js?1371896489866:60
0 ms	32 ms	▼lively.morphic.Morph.addMethods.showHalos	Halos.js?1371896489866:8
0 ms	20 ms	▼lively.morphic.World.addMethods.showHalosFor	Halos.js?1371896489866:78
0 ms	20 ms	▼(anonymous function)	Halos.js?1371896489866:83
0 ms	20 ms	▼wrapped	Function.js?1371896489374:37
0 ms	20 ms	▼lively.morphic.Morph.subclass.addMorph	Core.js?1371896489525:598
0 ms	13 ms	▶callSuper	Base.js?1371896489415:376
0 ms	7 ms	▼lively.morphic.World.addMethods.updateScrollFocus	Events.js?1371896489576:1683
0 ms	7 ms	▼(anonymous function)	Events.js?1371896489576:1685
0 ms	7 ms	▼lively.morphic.Morph.addMethods.getTopmostMorph	Events.js?1371896489576:1086
0 ms	7 ms	▼Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	4 ms	▶Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	3 ms	▶Object.subclass.fullContainsWorldPoint	Core.js?1371896489525:404

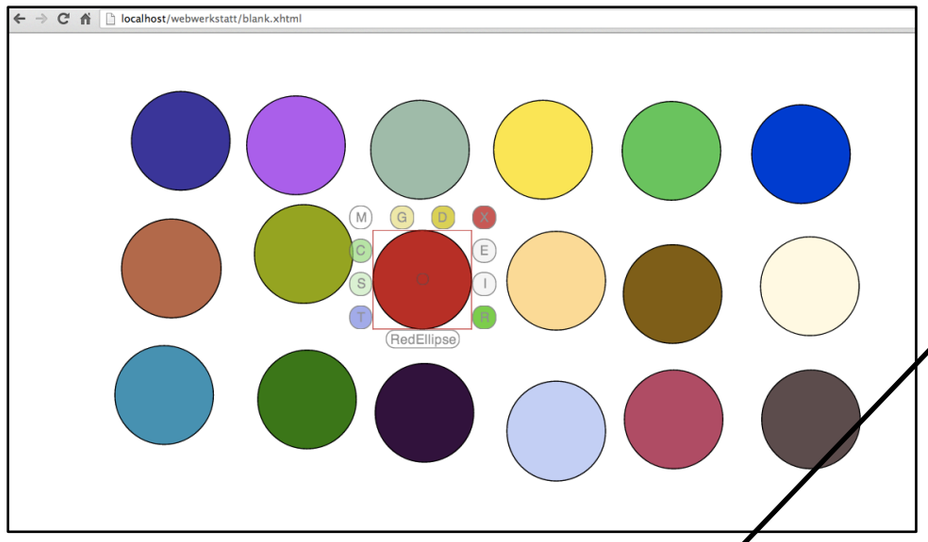
Issue #342, revision 147913




```
this.addScript(function toggleHalos(evt) {
    console.profile();
    $super(evt);
    console.profileEnd();
})
```

Self	Total	Function	
0 ms	34 ms	▼eventSpec.handlerFunc	Events.js?1371896489576:108
0 ms	34 ms	▼Object.subclass.handleEvent	Events.js?1371896489576:151
0 ms	34 ms	▼lively.morphic.Morph.addMethods.onMouseUpEntry	Events.js?1371896489576:716
1 ms	34 ms	▼toggleHalos	
0 ms	32 ms	▼(anonymous function)	
0 ms	32 ms	▼lively.morphic.Morph.addMethods.toggleHalos	Halos.js?1371896489866:60
0 ms	32 ms	▼lively.morphic.Morph.addMethods.showHalos	Halos.js?1371896489866:8
0 ms	20 ms	▼lively.morphic.World.addMethods.showHalosFor	Halos.js?1371896489866:78
0 ms	20 ms	▼(anonymous function)	Halos.js?1371896489866:83
0 ms	20 ms	▼wrapped	Function.js?1371896489374:37
0 ms	20 ms	▼lively.morphic.Morph.subclass.addMorph	Core.js?1371896489525:598
0 ms	13 ms	▶callSuper	Base.js?1371896489415:376
0 ms	7 ms	▼lively.morphic.World.addMethods.updateScrollFocus	Events.js?1371896489576:1683
0 ms	7 ms	▼(anonymous function)	Events.js?1371896489576:1685
0 ms	7 ms	▼lively.morphic.Morph.addMethods.getTopmostMorph	Events.js?1371896489576:1086
0 ms	7 ms	▼Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	4 ms	▶Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	3 ms	▶Object.subclass.fullContainsWorldPoint	Core.js?1371896489525:404

Issue #342, revision 147913

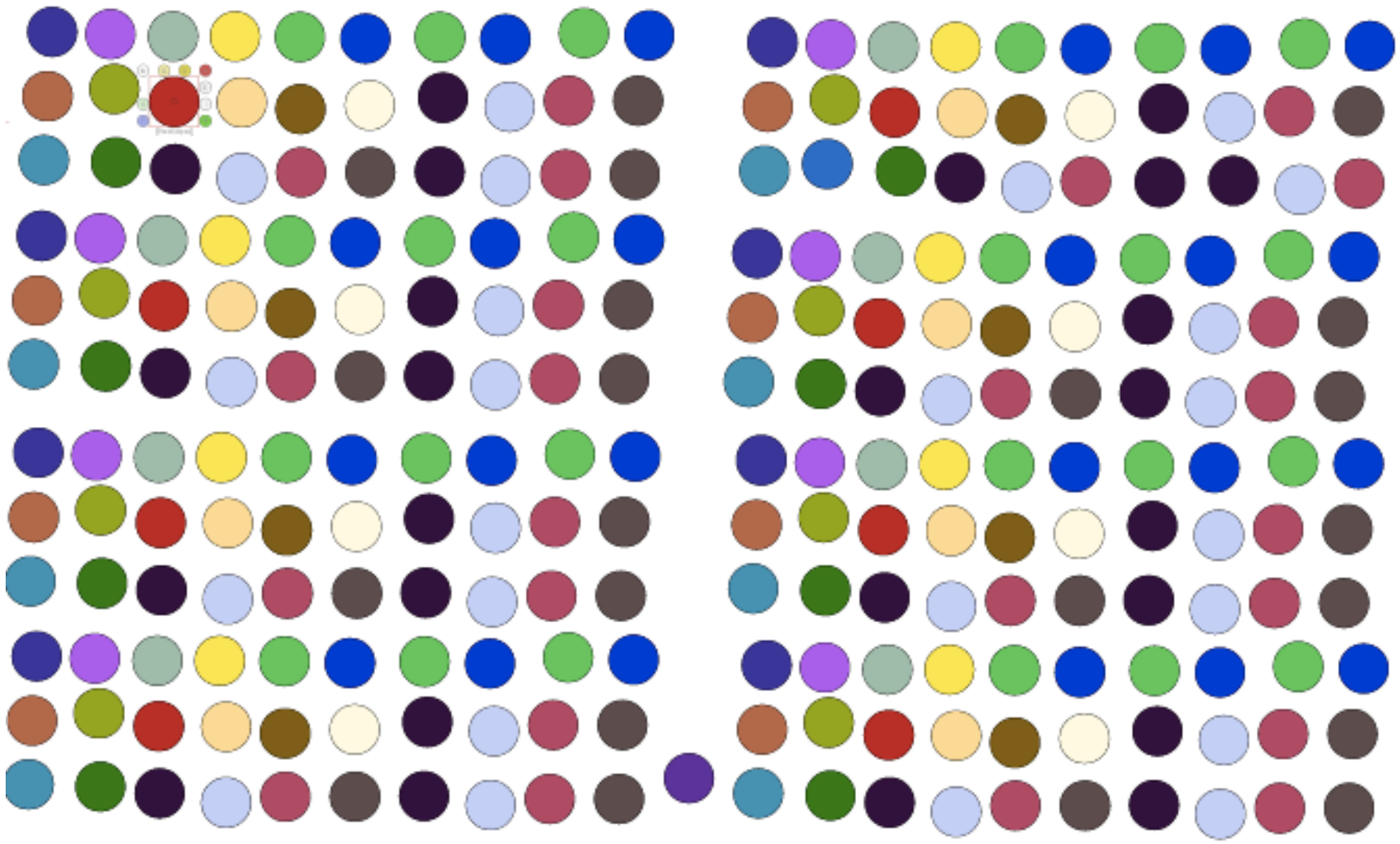



```
this.addScript(function toggleHalos(evt) {
    console.profile();
    $super(evt);
    console.profileEnd();
})
```

Self	Total	Function	
0 ms	34 ms	eventSpec.handlerFunc	Events.js?1371896489576:108
0 ms	34 ms	Object.subclass.handleEvent	Events.js?1371896489576:151
0 ms	34 ms	lively.morphic.Morph.addMethods.onMouseUpEntry	Events.js?1371896489576:716
1 ms	34 ms	toggleHalos	
0 ms	32 ms	(anonymous function)	
0 ms	32 ms	lively.morphic.Morph.addMethods.toggleHalos	Halos.js?1371896489866:60
0 ms	32 ms	lively.morphic.Morph.addMethods.showHalos	Halos.js?1371896489866:8
0 ms	20 ms	lively.morphic.World.addMethods.showHalosFor	Halos.js?1371896489866:78
0 ms	20 ms	(anonymous function)	Halos.js?1371896489866:83
0 ms	20 ms	wrapped	Function.js?1371896489374:37
0 ms	20 ms	lively.morphic.Morph.subclass.addMorph	Core.js?1371896489525:598
0 ms	13 ms	callSuper	Base.js?1371896489415:376
0 ms	7 ms	lively.morphic.World.addMethods.updateScrollFocus	Events.js?1371896489576:1683
0 ms	7 ms	(anonymous function)	Events.js?1371896489576:1685
0 ms	7 ms	lively.morphic.Morph.addMethods.getTopmostMorph	Events.js?1371896489576:1086
0 ms	7 ms	Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	4 ms	Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	3 ms	Object.subclass.fullContainsWorldPoint	Core.js?1371896489525:404

Issue #342, revision 147913

- toggling halos on one out of **250 morphs**



Issue #342, revision 147913

- toggling halos on one out of **250 morphs**

Self	Total	Function	
0 ms	86 ms	▼eventSpec.handlerFunc	Events.js?1371896489576:108
0 ms	86 ms	▼Object.subclass.handleEvent	Events.js?1371896489576:151
0 ms	86 ms	▼lively.morphic.Morph.addMethods.onMouseUpEntry	Events.js?1371896489576:716
1 ms	86 ms	▼toggleHalos	
0 ms	84 ms	▼(anonymous function)	
0 ms	84 ms	▼lively.morphic.Morph.addMethods.toggleHalos	Halos.js?1371896489866:60
0 ms	84 ms	▼lively.morphic.Morph.addMethods.showHalos	Halos.js?1371896489866:8
0 ms	70 ms	▼lively.morphic.World.addMethods.showHalosFor	Halos.js?1371896489866:78
0 ms	70 ms	▼(anonymous function)	Halos.js?1371896489866:83
0 ms	70 ms	▼wrapped	Function.js?1371896489374:37
0 ms	70 ms	▼lively.morphic.Morph.subclass.addMorph	Core.js?1371896489525:598
0 ms	55 ms	▼lively.morphic.World.addMethods.updateScrollFocus	Events.js?1371896489576:1683
0 ms	55 ms	▼(anonymous function)	Events.js?1371896489576:1685
0 ms	55 ms	▼lively.morphic.Morph.addMethods.getTopmostMorph	Events.js?1371896489576:1086
2 ms	55 ms	▼Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
1 ms	29 ms	▶Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	24 ms	▶Object.subclass.fullContainsWorldPoint	Core.js?1371896489525:404
0 ms	15 ms	▶callSuper	Base.js?1371896489415:376

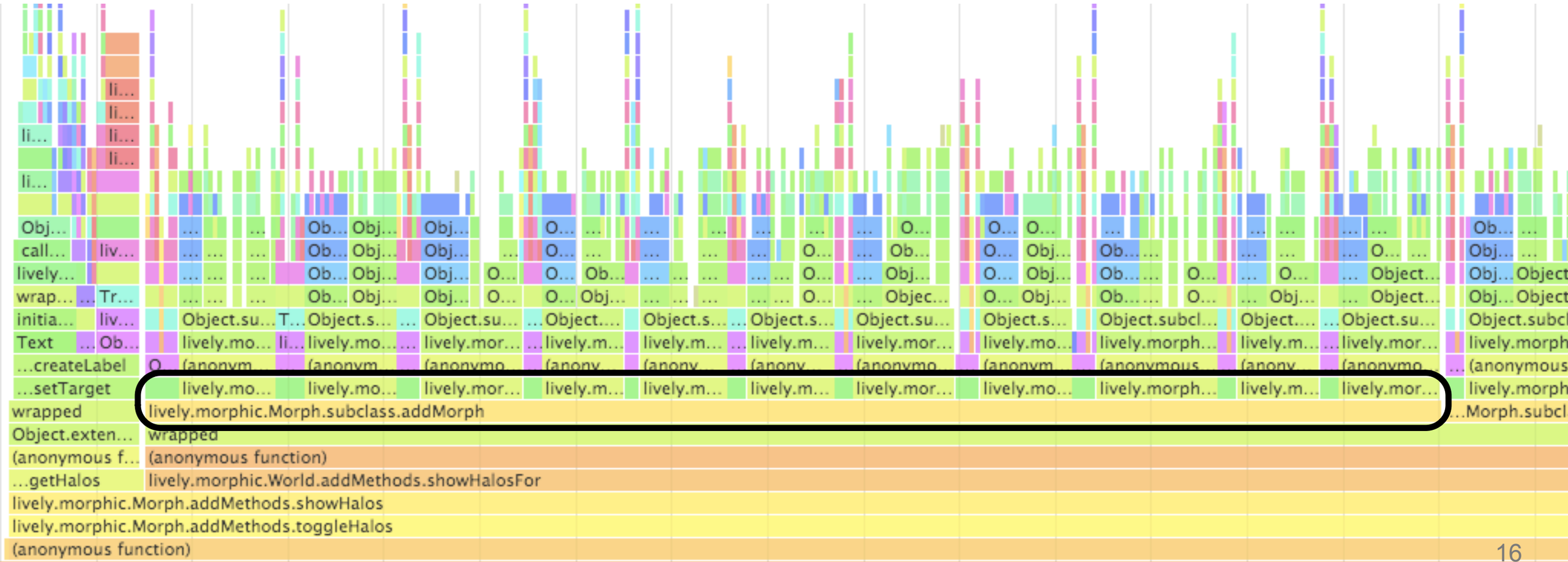
Issue #342, revision 147913

- toggling halos on one out of **250 morphs**

Self	Total	Function	
0 ms	86 ms	▼eventSpec.handlerFunc	Events.js?1371896489576:108
0 ms	86 ms	▼Object.subclass.handleEvent	Events.js?1371896489576:151
0 ms	86 ms	▼lively.morphic.Morph.addMethods.onMouseUpEntry	Events.js?1371896489576:716
1 ms	86 ms	▼toggleHalos	
0 ms	84 ms	▼(anonymous function)	
0 ms	84 ms	▼lively.morphic.Morph.addMethods.toggleHalos	Halos.js?1371896489866:60
0 ms	84 ms	▼lively.morphic.Morph.addMethods.showHalos	Halos.js?1371896489866:8
0 ms	70 ms	▼lively.morphic.World.addMethods.showHalosFor	Halos.js?1371896489866:78
0 ms	70 ms	▼(anonymous function)	Halos.js?1371896489866:83
0 ms	70 ms	▼wrapped	Function.js?1371896489374:37
0 ms	70 ms	▼lively.morphic.Morph.subclass.addMorph	Core.js?1371896489525:598
0 ms	55 ms	▼lively.morphic.World.addMethods.updateScrollFocus	Events.js?1371896489576:1683
0 ms	55 ms	▼(anonymous function)	Events.js?1371896489576:1685
0 ms	55 ms	▼lively.morphic.Morph.addMethods.getTopmostMorph	Events.js?1371896489576:1086
2 ms	55 ms	▼Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
1 ms	29 ms	▶Object.subclass.morphsContainingPoint	Core.js?1371896489525:298
0 ms	24 ms	▶Object.subclass.fullContainsWorldPoint	Core.js?1371896489525:404
0 ms	15 ms	▶callSuper	Base.js?1371896489415:376

Issue #342, revision 147913

- toggling halos on one out of **250 morphs**



Issue #342, revision 147913

- toggling halos on one out of **250 morphs**

Issue #342, revision 147913

- toggling halos on one out of **250 morphs**

Object.subclass.morphsContainingPoint
lively.morphic.Morph.addMethods.getTopmostMorph
(anonymous function)
lively.morphic.World.addMethods.updateScrollFocus

```
showHalosFor: function(morph, halos) {
    if (this.currentHaloTarget) {
        this.currentHaloTarget.removeHalos(this);
    }
    this.currentHaloTarget = morph;
    halos.forEach(function(halo) { this.addMorph(halo); }, this);
}
```

wrapped
lively.morphic.Morph.subclass.addMorph
Object.exten…
wrapped
(anonymous f…
(anonymous function)
…getHalos
lively.morphic.World.addMethods.showHalosFor
lively.morphic.Morph.addMethods.showHalos
lively.morphic.Morph.addMethods.toggleHalos
(anonymous function)
toggleHalos

Issue #342, today

- diff r147913:r196426
  - bounds caching
  - no longer a call to #updateScrollFocus per halo button added

at r147913

Function
▼ eventSpec.handlerFunc
▼ Object.subclass.handleEvent
▼ lively.morphic.Morph.addMethods.onMouseUpEntry
▼ toggleHalos
▼ (anonymous function)
▼ lively.morphic.Morph.addMethods.toggleHalos
▼ lively.morphic.Morph.addMethods.showHalos
▼ lively.morphic.World.addMethods.showHalosFor
▼ (anonymous function)
▼ wrapped
▼ **lively.morphic.Morph.subclass.addMorph**
▶ Object.subclass.addMorph
▶ lively.morphic.World.addMethods.updateScrollFocus
▶ Enumerable.invoke
▶ lively.morphic.Morph.addMethods.getHalos

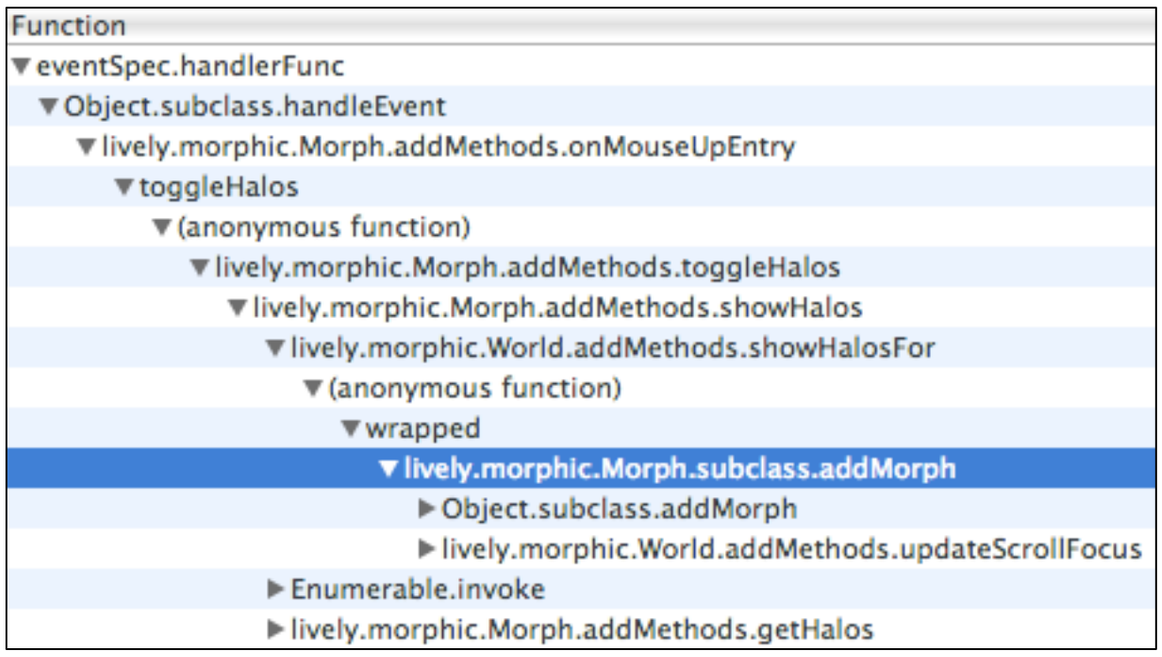
at r196426

Function
▼ eventSpec.handlerFunc
▼ Object.subclass.handleEvent
▼ lively.morphic.Morph.addMethods.onMouseUpEntry
▼ toggleHalos
▼ (anonymous function)
▼ lively.morphic.Morph.addMethods.toggleHalos
▼ lively.morphic.Morph.addMethods.showHalos
▼ lively.morphic.World.addMethods.showHalosFor
▼ (anonymous function)
▼ wrapped
▼ **lively.morphic.Morph.subclass.addMorph**
▼ callSuper
▶ Object.subclass.addMorph
▶ lively.morphic.Morph.addMethods.getHalos
▶ Object.extend.invoke

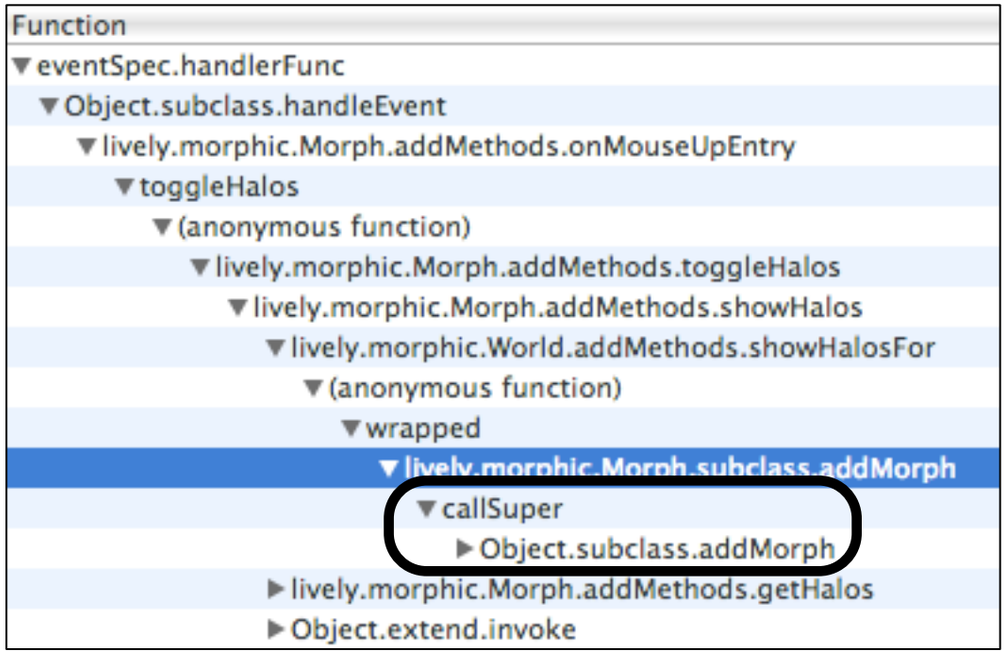
Issue #342, today

- diff r147913:r196426
  - bounds caching
  - no longer a call to #updateScrollFocus per halo button added

at r147913



at r196426



Limitations

- chrome developer tool
  - not much postprocessing: no searching, no easy filtering
  - limited to JavaScript execution, no info on [native code]

- v8's built-in profiler
  - no tallies: not clear how often a function got executed
  - call tree only approximated: e.g. missing frames
  - only function boundaries, no line-by-line info
  - not possible to profile just specific parts of a system, only global or dynamic scoping

Built-in v8 Profiler

v8‘s profiler

- statistical profiling / sampling: records runtime information at a fixed interval

- v8 profiler
  - records a tick sample every millisecond
  - allocates tick samples to the current stack frame

- sources
  - v8/src/cpu-profiler.cc
  - v8/src/profile-generator.cc

```
ProfileGenerator::RecordTickSample(const TickSample& sample)
// records stack frames + pc + function + vm-state
```

d8 --prof [js files]

- tick samples for shared libraries, c++, and JavaScript

```
d8 --prof base.js run.js
```

```
[...]
tick,0x25dcce424469,0x7fff5fbfefb0,20040498,0,0x175fba504121,0,0x25dcce423cae,0x25dcce0719eb,0x25dcce07180d,0x25dcce42c90f,
0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce42093d,0x7fff5fbfec90,20041662,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce424834,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce42093d,0x7fff5fbfec90,20042815,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce424834,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce46c910,0x7fff5fbfede0,20043968,0,0x402999999999999a,0,0x25dcce4248b0,0x25dcce423cae,0x25dcce0719eb,0x25dcce07180d,
0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce420974,0x7fff5fbfec90,20045157,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce4248e7,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce4208e6,0x7fff5fbfec90,20046312,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce4248e7,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce420955,0x7fff5fbfec90,20047350,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce4248e7,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce46a042,0x7fff5fbfedc0,20048528,0,0x5462769e771,0,0x25dcce424834,0x25dcce423cae,0x25dcce0719eb,0x25dcce07180d,
0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce420945,0x7fff5fbfec90,20049684,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce424834,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce420941,0x7fff5fbfec90,20050840,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce424834,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce46a0b6,0x7fff5fbfedc0,20051991,0,0x25dcce46ca61,0,0x25dcce4248e7,0x25dcce423cae,0x25dcce0719eb,0x25dcce07180d,
0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce420955,0x7fff5fbfec90,20053147,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce4248e7,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce42099a,0x7fff5fbfec90,20054308,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce4248e7,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce42444d,0x7fff5fbfefb0,20055468,0,0x175fba504121,0,0x25dcce423cae,0x25dcce0719eb,0x25dcce07180d,0x25dcce42c90f,
0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
tick,0x25dcce420907,0x7fff5fbfec90,20056657,0,0x3ff0000000000000,0,0x25dcce46a203,0x25dcce424834,0x25dcce423cae,0x25dcce0719eb,
0x25dcce07180d,0x25dcce42c90f,0x25dcce42c9aa,0x25dcce0708b4,0x25dcce0707b1,0x25dcce039c8b
[...]
```

Tick Processing

```
tools/mac-tick-processor v8.log
```

```
Statistical profiling result from v8.log, (17439 ticks, 26 unaccounted, 0
excluded).

 [JavaScript]:
   ticks  total  nonlib   name
   1903   11.2%   11.2%  LazyCompile: *montReduce crypto.js:583
   1340    7.9%    7.9%  LazyCompile: *project navier-stokes.js:239
   1073    6.3%    6.3%  LazyCompile: *Scheduler.schedule richards.js:188
    876    5.1%    5.1%  LazyCompile: *montSqrTo crypto.js:603
    689    4.0%    4.0%  LazyCompile: *rewrite_nboyer earley-boyer.js:3604
    557    3.3%    3.3%  LazyCompile: *Plan.execute deltablue.js:773
[...]

[C++]:
   ticks  total  nonlib   name
   2372   13.6%   13.6%  start
    267    1.5%    1.5%  ___add_ovflpage
     25    0.1%    0.1%  _getcontext
     24    0.1%    0.1%  ___fork
     20    0.1%    0.1%  _swapcontext
     12    0.1%    0.1%  _mach_port_insert_right
     12    0.1%    0.1%  _csinhl$fenv_access_off
[...]
```

Sampling (Profiling)

- record current frame at fixed intervals
- statistically where time is spent, low overhead
- but: potentially missed frames, interval allocation to just one frame, no tallies

vs.

Instrumentation (Tracing)

- insert recording code at function boundaries
- exact call graphs, tallies, flexible scoping, state logging
- but: high overhead that might impede measurements or usabilty

Custom Tracing

Runtime Instrumentation

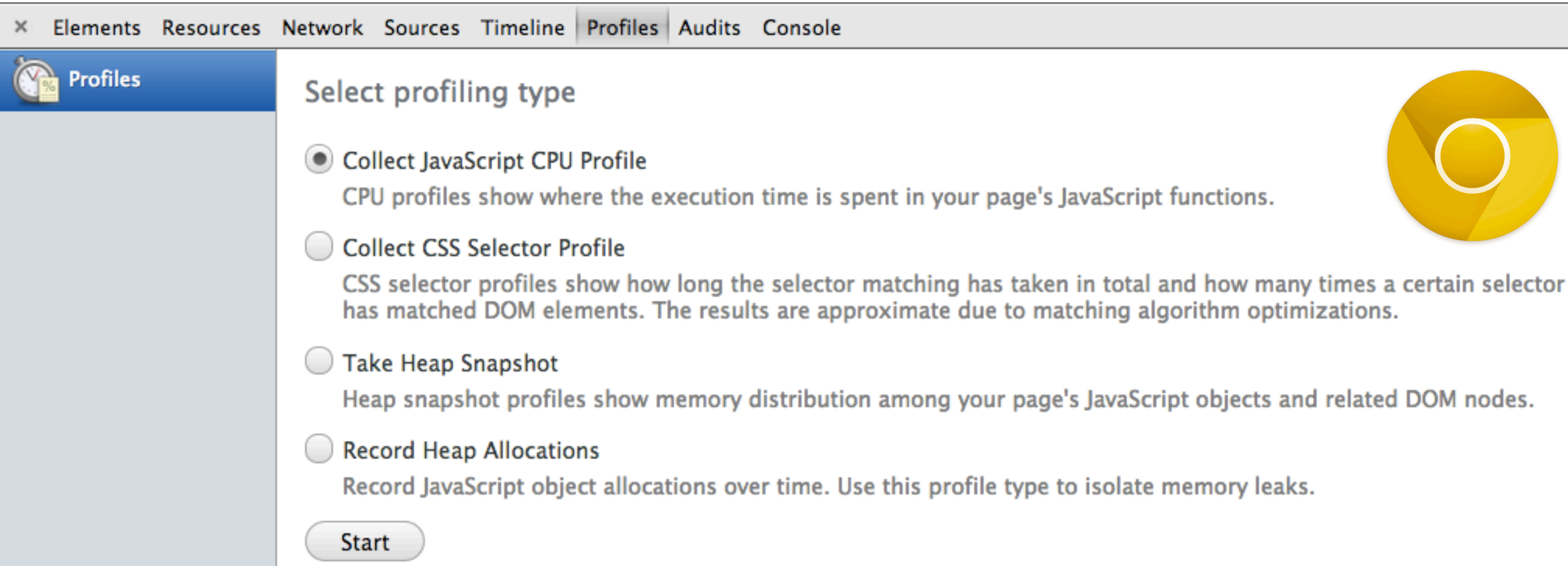
- wrapping functions at runtime

```
Tracer.trace = function(object, methodName) {
  var orgFunc = object[methodName];
  object[methodName] = function() {
    Tracer.log(this, methodName, arguments);
    return orgFunc.apply(this, arguments)
  }
}
```

[ContextJS Tracing]

- but: possible to instrument code multiple times, infinite recursion when tracing the instrumentation code
- better use a library, e.g. trace with ContextJS
- overhead

Memory Profiling / Chrome



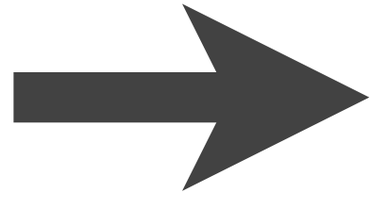
Chrome Developer Tools: Memory Profiling

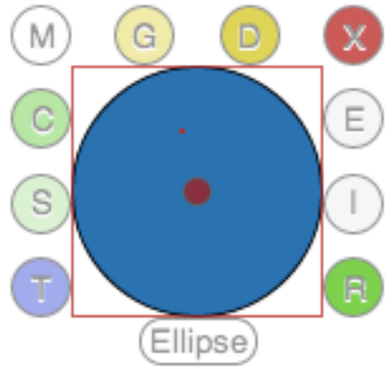
- snapshots
  - what objects are available at a particular moment
  - comparisons between snapshots

- allocation recordings (Chrome Canary)
  - what happened in a particular time frame

- info per object
  - distance from GC roots: e.g. to built-in globals
  - shallow size: size of memory held by the object itself
  - retained size: size of memory freed when object gets deleted
  - object's retaining tree: from this object to GC roots

Recording:





Elements Resources Network Sources Timeline Profiles Audits Console

Profiles

HEAP SNAPSHOTS

Snapshot 3

HEAP TIMELINES

Snapshot 3
14.7 MB

5.00 s 10.00 s 15.00 s 20.00 s 25.00 s

10.0 KB

Class filter

Constructor	Distance	Objects Count▼	Shallow Size	Retained Size
▶Array	4	19 0%	304 0%	776 0%
▶Point	5	19 0%	380 0%	836 0%
▶system / Context	4	14 0%	392 0%	536 0%
▶Color	5	13 0%	364 0%	988 0%
▶HTMLDivElement	2	4 0%	80 0%	412 0%
▶Rectangle	4	3 0%	108 0%	248 0%
▶NodeList	2	2 0%	40 0%	40 0%
▼Text	2	2 0%	32 0%	5 624 0%
▶Text @575201	2		20 0%	20 0%
▶Text @575427	6		12 0%	5 604 0%
▶EventHandler	5	1 0%	20 0%	1 000 0%
▶FocusEvent	10	1 0%	20 0%	144 0%
▶HTMLBRElement	2	1 0%	20 0%	20 0%
▶HTMLSpanElement	2	1 0%	20 0%	20 0%
▶InternalPackedArray	3	1 0%	16 0%	36 0%

Object's retaining tree

Object	Shallow Size	Retained Size	Distance ▲
▶morph in @574849	12 0%	32 0%	5
▼labelMorph in RenameHalo @50829	12 0%	2 312 0%	5
▼[10] in Array @575435	16 0%	92 0%	4
▼submorphs in World @29031	12 0%	18 508 0%	3
▶world in MouseEvent @576655	20 0%	2 200 0%	2
▶owner in HandMorph @51739	12 0%	3 088 0%	3
▶_world in HandMorph @51739	12 0%	3 088 0%	3
▶morph in EventHandler @232579	12 0%	1 644 0%	4
▶owner in Morph @6983	12 0%	1 048 0%	4
▶owner in InspectHalo @49133	12 0%	1 912 0%	4
▶owner in OriginHalo @59419	12 0%	1 836 0%	4

Report Outlook:

Related Tools

Summary

Summary

- Profiling versus tracing
  - sampling: overview, statistically where time was spent, but approximated call graphs
  - instrumentation: code comprehension and custom analysis, exactly what happened, but higher overhead
- Different tools
  - chrome developer tools: CPU profiles and heap snapshots
  - v8's built-in profiler: shared libraries, C++, JavaScript
  - custom tracing: correct tallies, flexible scoping, state logging
- And there are many more tools
  - network pane, css selector matching, event timelines, frame rate profiler, network and performance audits...

References

Web Resources

- Chrome Developer Tools:
  - developers.google.com/chrome-developer-tools
  - developers.google.com/chrome-developer-tools/docs/memory-analysis-101
- v8‘s built-in profiler: github.com/v8/v8
  - code.google.com/p/v8/wiki/V8Profiler
  - code.google.com/p/v8/wiki/ProfilingChromiumWithV8
  - developers.google.com/v8/profiler_example
- Node.js
  - v8 profiler bindings: github.com/bnoordhuis/node-profiler
  - Node Inspector: github.com/dannycoates/node-inspector
  - Node Webkit Agent: github.com/c4milo/node-webkit-agent
  - Nodetime: docs.nodetime.com
  - using available DTrace providers: blog.nodejs.org/2012/04/25/profiling-node-js/
  - creating custom DTrace providers: github.com/chrisa/node-dtrace-provider, mcavage.github.io/presentations/dtrace_conf_2012-04-03/
- Helper Monkey / DTrace for SpiderMonkey: blogs.oracle.com/brendan/entry/dtrace_meets_javascript
- Firefox: developer.mozilla.org/en-US/docs/Tools/Profiler
  - wiki.mozilla.org/Performance%3aLeak_Tools
  - built-in profiler: developer.mozilla.org/en-US/docs/Performance/Profiling_with_the_Built-in_Profiler
- Safari: developer.apple.com/library/safari/#documentation/AppleApplications/Conceptual/Safari_Developer_Guide/OptimizingYourWebsite/OptimizingYourWebsite.html
- Internet Explorer: msdn.microsoft.com/en-us/library/gg699341(v=vs.85).aspx

Published Resources

- **[COP]** Robert Hirschfeld, Pascal Costanza, and Oscar Nierstrasz. Context-oriented Programming. In Journal of Object Technology (JOT) 7, 3. 2008.
- **[ContextJS Tracing]** Jens Lincke, Robert Krahn, and Robert Hirschfeld. Implementing Scoped Method Tracing with ContextJS. In Proceedings of the Workshop on Context-oriented Programming (COP). 2011.