Javascript Tuning Techniques

Mark Stetzer
@stetzer

Introduction

- Javascript is universal on the web; almost all sites use it in some form
- Almost 90% of page performance is on the front-end (*http://www.stevesouders.com/blog/2012/02/10/the-performance-golden-rule/*)
- If we're all going to be slinging JS, let's at least learn to write it efficiently

Overview

- Script positioning
- Testing with jsPerf
- Managing scope
- DOM access

Script positioning

- Do not put *script* tags in *head* of doc
- *script* tags block page render until download & execution are complete

Script positioning

- Defer until bottom of *body*
- Most sites don't need JS functionality immediately anyway
- Downloaded sequentially in IE 6,7

Non-blocking scripts

- *defer* attribute for browsers that support
- Indicates DOM won't be modified; download starts when tag is encountered
- Rendering blocked during parse/exec (just like when at bottom of *body*)

Docwriting scripts

- Can docwrite script element into DOM
- Will download & execute w/o affecting rest of page
- DOM elements below inserted script blocked until script done downloading
- Insert in head or IE may encounter an *operation aborted*

Docwriting scripts

- "Blocks other dynamic script. One exception is if multiple scripts are inserted using document.write within the same SCRIPT block"
- *http://www.stevesouders.com/blog/2012/04/10/dont-docwrite-scripts/*

Async scripts

- Similar to *defer*; download starts immediately, execution is asynchronous (order not guaranteed)
- Not supported in IE < 10

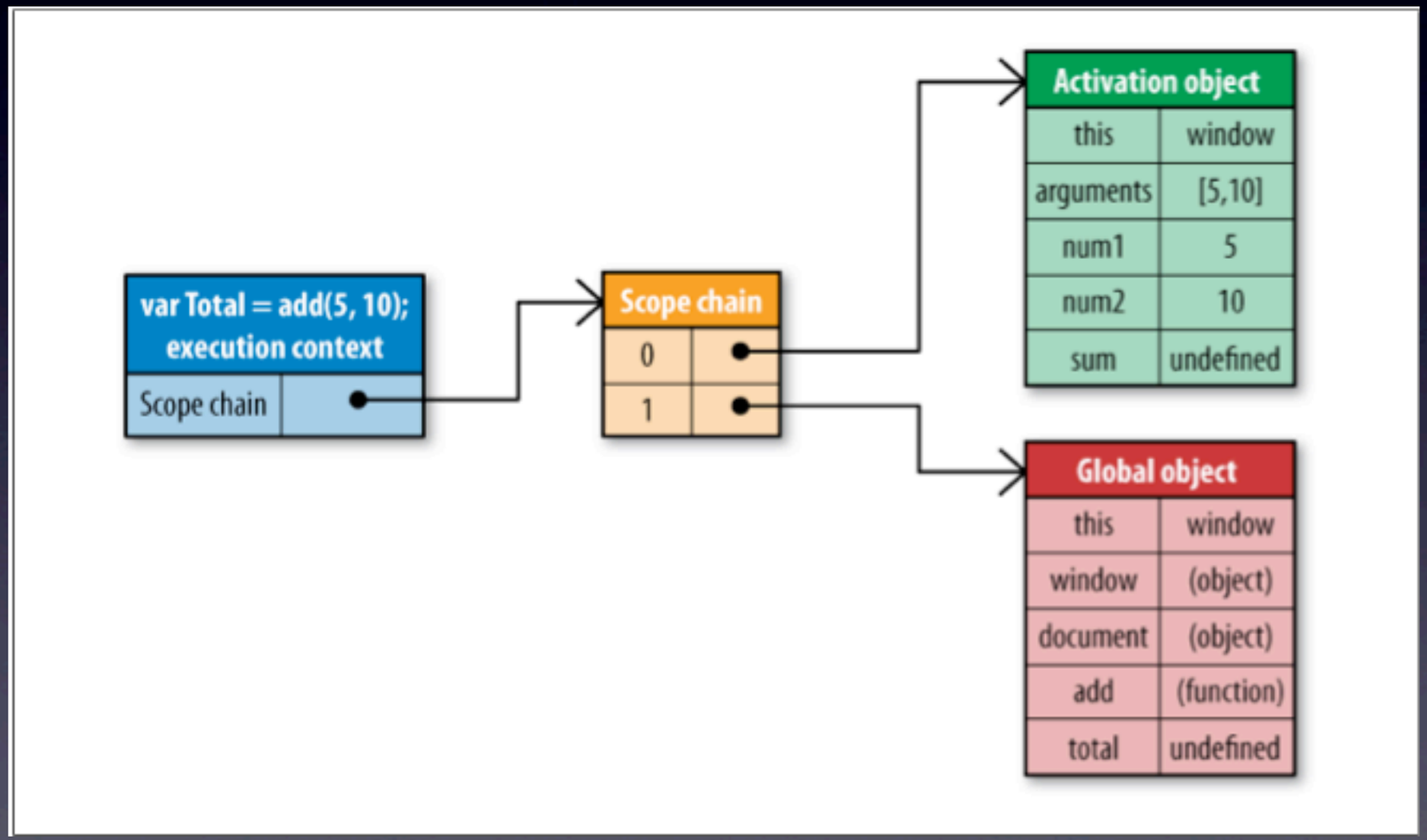
Other alternatives?

- Could load via XHR (similar to docwrite but supported on ALL modern browsers)
- Problem is XSS; excludes use of CDNs
- Script loader?
- Great advice at *http://www.stevesouders.com/blog/2009/04/27/loading-scripts-without-blocking/*

Testing with jsPerf

- *http://jsperf.com/*

Managing Scope



High Performance JavaScript, Zakas (O'Reilly)

Scope Chains

- Literals & locals good, array items & object members not as much
- Locals better than out-of-scope; assign to locals if using more than once
- *http://jsperf.com/globals-vs-locals-test*

Scope chain augmentation

- Don't use *with* because it augments execution context scope chain
- Same for *try-catch* - avoid catching if possible
- Call a function if you have to catch

Closure performance

- Closures require more memory
- IE implements DOM objects as non-native JS objects, so closures can cause memory leaks

DOM access

- Access involves crossing "toll bridge"; cross "bridge" as little as possible
- *http://jsperf.com/innerhtml-loop*
- With latest modern browsers DOM access is fastest
- *http://jsperf.com/innerhtml-vs-dom-table*

DOM access

- Just like local vs global variables, local vars faster than *HTMLCollection* methods (except for Chrome probably because of optimizations - inconsistent test results)
- *http://jsperf.com/count-htmlcollection*
- For older IE versions, *nextSibling* is much faster than *childNodes*

querySelectorAll

- Less code, faster than iterating on older browsers, but not with newest browsers according to jsPerf
- Supported on IE 8+, FF 3.5+, Chrome 4+, Safari 3.1+, Opera 10+
- *http://jsperf.com/queryselectorall-vs-iterate/2*

Reflow/Repaint

- Repaints & reflows slow things down; strive for operations in batches
- When layout info requested, queue flushed & changes applied to provide updated info
- Position animations absolutely vs relative (reflow)
- *http://jsperf.com/cached-layout-info*

The story continues...

- High Performance Javascript (*http://shop.oreilly.com/product/9780596802806.do*)
- Even Faster Web Sites (*http://shop.oreilly.com/product/9780596522315.do*)
- *http://www.stevesouders.com/*
- Advanced Javascript Tuning Techniques?

Questions?

- These slides will be available on GitHub at *https://github.com/stetzer/IndyWebPerf*