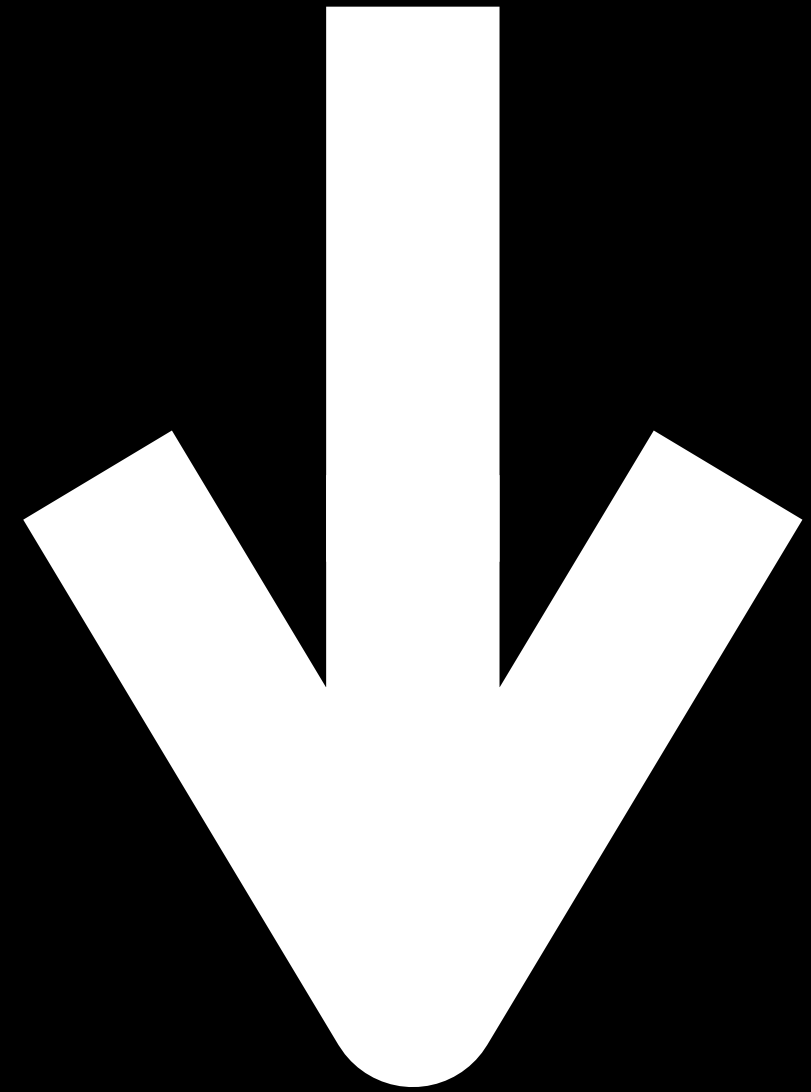


Server Sent Events



By Kornel Lesiński (@pornelski) for London JS, July 2011, <http://lanyrd.com/2011/ldnjs03/>

XHR

Everyone uses XMLHttpRequest. AJAX is in every developers' toolbox. And recently, WHATWG has re-invented TCP/IP with WebSockets.

XHR

**Web-
Sockets**

And a question has arisen: is there a room...

XHR

?

**Web-
Sockets**

...for a third category of API in the middle – something between XHR and WebSockets? And browser vendors pondered this for years as well. The bar is really high. In order to create a new category of API, it has to be far better at some key things: it has to be very easy to use. It has to have great security. It must be compatible with browsers, servers and proxies.

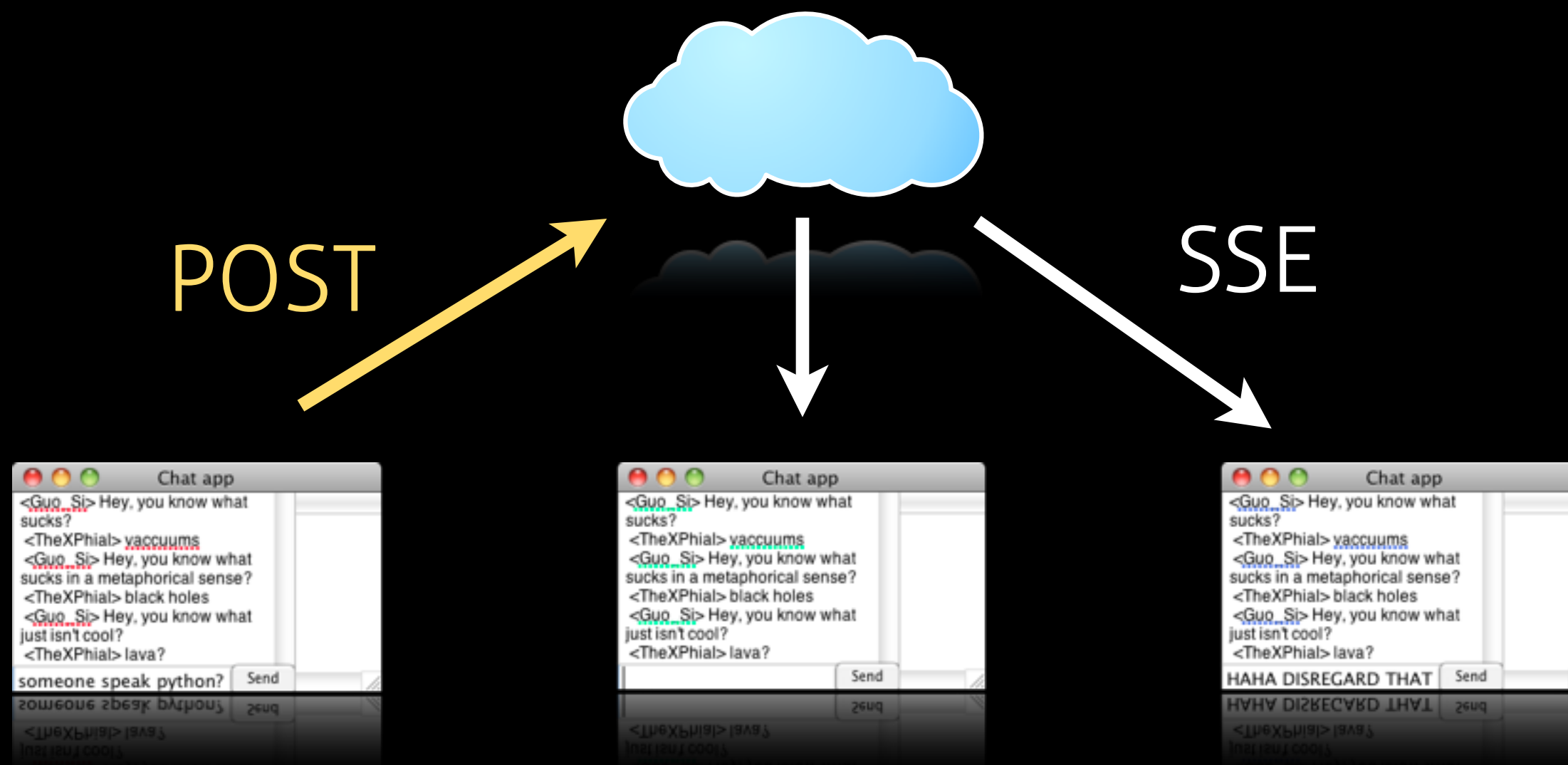
XHR

SSE

**Web-
Sockets**

And I think I've got something that is. And I'd like to show it to you to day. It's called the iPad^W^WServer Sent Events. It enables you to do fantastic things

Real-time chat



You can create real-time chat application. A posted message can be pushed to everyone in the chat, instantly.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800
801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900
901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000

You can use it to watch your server logs and performance indicators in real time. It's perfect for this.

GOOG 68.68 (12.98%)



Someone has tweeted!

6) OPR.OL 0.70 (2



Someone has tweeted!



Someone has tweeted!



Someone has tweeted!



Someone has tweeted!



Someone has tweeted!



Someone has tweeted!



Someone has tweeted!

You can do boring stuff, like up-to-date stock tickers. And you can do more interesting stuff, like instant notifications on your pages whenever a message is posted (as soon as sender presses Send). API for all of this is very easy:

Very simple API

```
var source = new EventSource("url");  
  
source.onmessage = function(event) {  
    alert(event.data);  
}
```

You create new event source, attach your event handler and read the data. That's it!

- Automatically reconnects/resumes
- Reuses connections to the same URL
- Avoids blocking other HTTP traffic

The important thing is all the things you DON'T have to do. You don't have to handle errors! Browser will the keep connection alive for you. You don't have to manage resources. If you create lots of EventSource objects, browser is allowed to open just one connection and distribute events to all objects. Browser knows that these are special long-lived connections and will ensure they're handled properly and don't hit limits or block queues for other resources.

text/event-stream

```
data: Hello World!\n\n
```

How does the stream look like? Couldn't be simpler. It's a text file! Lines prefixed data: set text to send (no length limit), and empty line fires the event.

The protocol can do a bit more, but you don't need to read a 200-page RFC.

data: Hello
data: World!
id: 123

Last-Event-Id: 123

ms

retry: 1000

event: bark
data: Woof!

: comment

The whole thing fits on one slide. You don't have protocol switching, framing, packet fragmentation or masking. To send multiline data, prefix each line with **data:**. If you set ID, browser will send it back to you when it reconnects. This way you can avoid sending same data twice (e.g. send scrollbar in a chat app to new users only). Retry lets you control how long to wait before reconnect. You can name events and use `addEventListener('name')` to avoid huge `onmessage` handler. Comments are for debug and heartbeat.

- Works with regular web servers
- Works with HTTP proxies
- Supports long polling (Comet)

SSE is just a download of a text file. You can use any server that can slowly generate a text file. You don't need special servers and complicated libraries for it! Since it's regular HTTP, it works with all proxies just fine. Even connection drops are not a problem, because it can gracefully reconnect.

```
res.writeHead(200, {"Content-Type":  
                    "text/event-stream"});  
  
res.write("data: " +  
          JSON.stringify(stuff) +  
          "\n\n");  
res.flush();
```


On the server it is very simple as well. Set content type and print lines. Note that I'm not including any special library and I'm not doing any handshake.

Bonus tip here is to use `JSON.stringify` and `JSON.parse`, so you can use JS end-to-end. Flush is needed to actually send data, instead of it sitting somewhere in buffers.

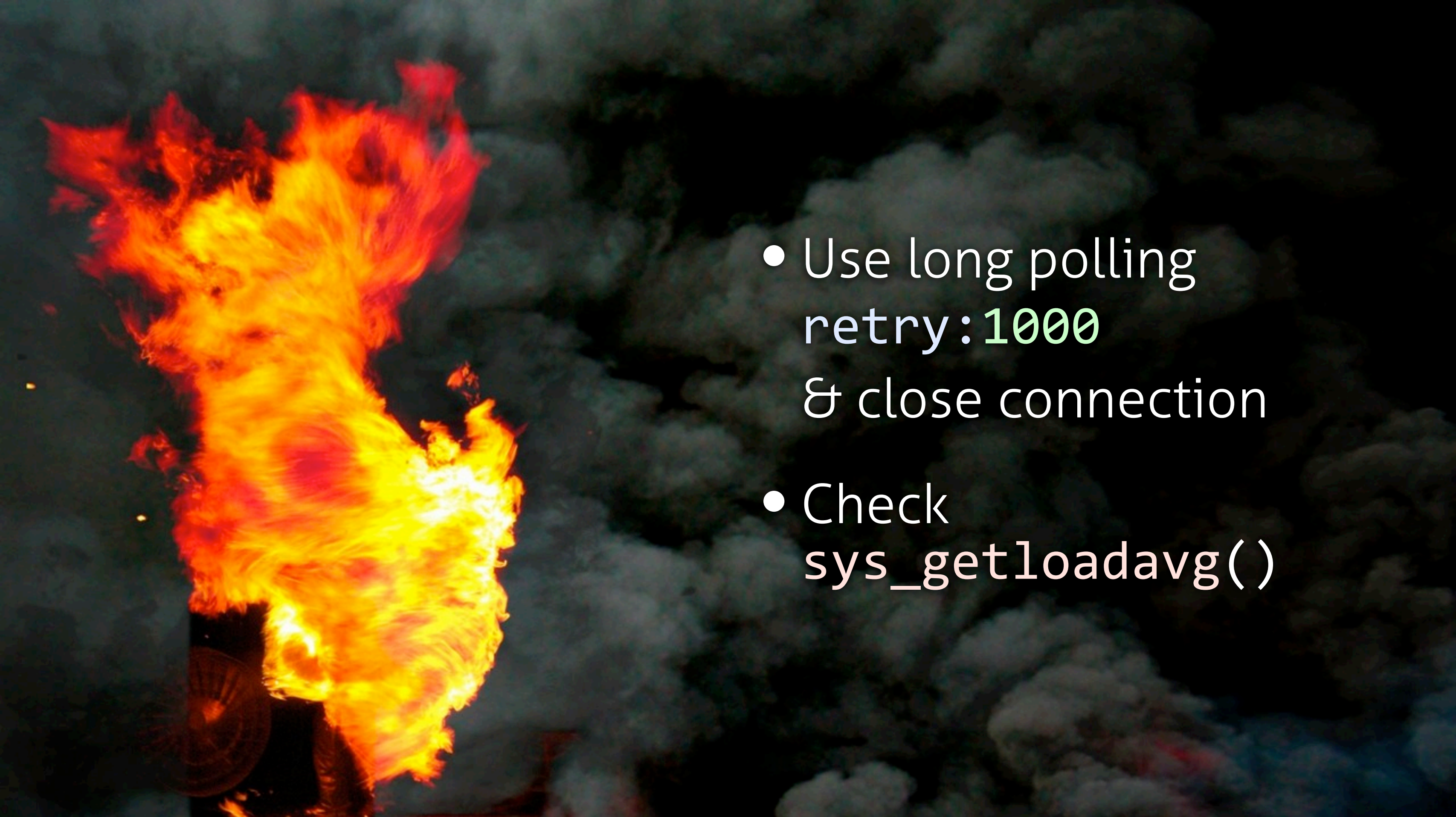

```
header("Content-Type: text/event-stream");

do {
    echo "data :".json_encode($stuff)."\n\n";
    flush();
    sleep(1);
} while(!connection_aborted());
```

If you don't have node.js I'm not sure what you're doing on londonjs meetup, but you can generate event stream even with clunky old PHP. You might be thinking – if I unleash this on millions of visitors, is my server going to look like...

- 
- Admin interfaces
 - Watch server logs/
traffic in real-time
 - Moderators'
notifications

is my server going to look like this? Yes, yes it will! Two ways around it: limit number of users. Use it for yourself in admin, give it to few special users. Or, if you're brave, you can do it on larger scale:

- 
- Use long polling
retry: 1000
& close connection
 - Check
sys_getloadavg()

Instead of keeping connections open all the time, just close them from time to time. Browser will reconnect without fuss. You control length of connection and control delay between reconnection, so you can have full spectrum between persistent connection and polling. You can even be smart about this and watch server load – if it's low, then use persistent, when it's too high, switch to polling and increase back-off time. So, SSE can work with most servers. What about browsers?



11

(9)

6

5

4

6

beta

Peachy! Firefox has been lagging behind, but finally SSE is almost there.

There is something missing here, isn't it? A certain browser built-in into an outdated OS of an American corporation...



Android of course! The other one doesn't work either. But fear not! Fallback is easy. Since SSE is just a text file, syntax is trivial to parse and it can fall back to polling, you can read it with XHR. It's easy to write yourself, you can find polyfill for this too. I'm pretty sure that Pusher guys have infiltrated the room by now, and they have dealt with some server and client problems. Speaking of problems, there are a couple:

- Same-origin limitation
(even port has to be the same!)
- CORS in the future
- No binary data (UTF-8 only)

You must use host and port for serving the page itself and the event stream. It's annoying if you want to use Node.js for the stream and have something else for regular pages. You can work around it by rewriting+proxying just the stream URL with nginx or Varnish (this is another place where proxy compatibility pays off!)



History

First version was on Hixie's blog in 2004 in response to "wouldn't it be cool if server could fire any event?" And it was really any event—you could remotely control page by firing mouse clicks and keyboard events!



<event-source>

You could just hook stream with <event-source> element and the server could play with the page. Unsurprisingly, it turned out to be difficult implement. Opera released first implementation in 2006, but Firefox had a patch for it a bit earlier. However, it took years for the patch to be reviewed, fixed, reviewed, updated, etc. In 2009 the spec has been simplified to only allow custom events, having element for JS-only API felt stupid. This broke Firefox's patch. Updated patch missed Fx4 deadline. Fx5 wasn't making big changes. Finally Fx6 has it. I'd go mad if my patch took so long to land, but Wellington Fernando de Macedo—who wrote most of it—in the meantime wrote WebSockets patch too!

Future

- SMS
- Push notifications

Mozilla hesitated adding SSE, which is a bit redundant with XHR multipart/replace hack and WebSockets, but eventually decided to do it, hoping for the future: SSE is protocol-agnostic. It doesn't have to work over HTTP. It might work over SMS or Apple-style push notifications. An incoming event could launch Firefox mobile, which would open appropriate tab and fire event there—a real push! It's still far off thought, there isn't even spec for it yet. In short term, EventSource is coming to SharedWorkers, so you'll be able to e.g. fire notification only in one tab that user is looking at.

ⓘ <http://pornel.net/sse>

© CC-BY-SA

- http://www.flickr.com/photos/gauri_lama/2672901420/
- <http://www.flickr.com/photos/wheatfields/116783486/>
- <http://arrioch.deviantart.com/>