

MEEBO BAR V10

The new Meebo Bar contains five key performance improvements.

1. Just-in-time code delivery

Code gets delivered just when it is needed instead of all at once. For example, the initial JavaScript payload is just 1/3 of its previous size. The result is a faster bar.

2. True persistent caching

All JavaScript, is cached in local storage, giving a near-zero download experience on subsequent visits.

3. Bundled image loading

Images are intelligently packaged into existing payloads, which means fewer network requests and faster image loading.

4. Non-blocking code

Loading JavaScript into a page can often block the rendering of the page. The new Meebo Bar completely eliminates this possibility.

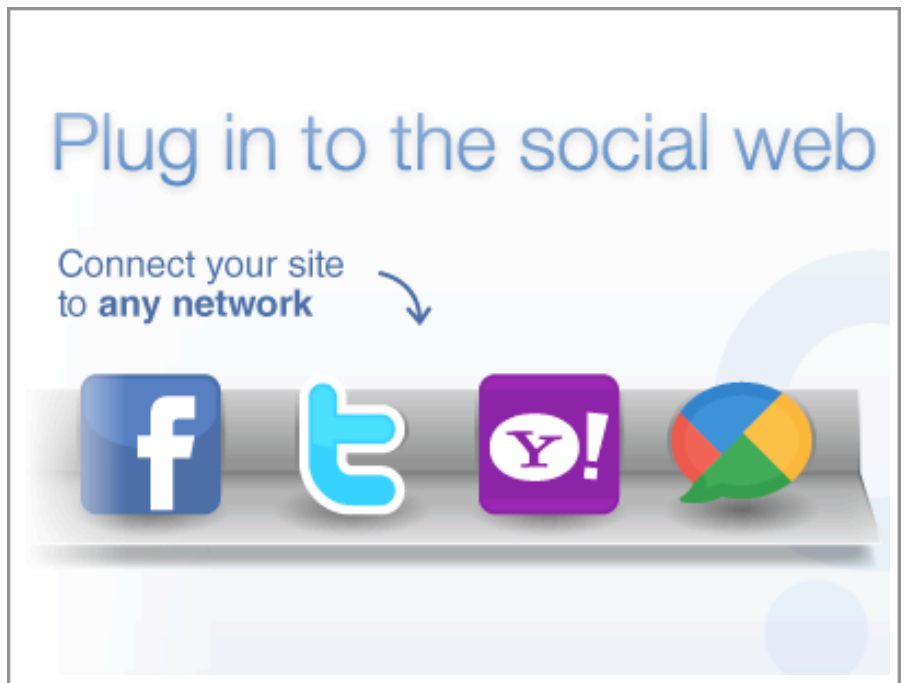
5. Faster rendering

The Meebo Bar renders structural graphics as vectors, resulting in fewer image downloads and faster drawing of the bar.

WANT TO LEARN MORE?

Two of our engineers recently gave a lecture with all the gory technical details at the Velocity 2010 conference on web performance. Check it out at:

<http://mee.bo/meebotalk>



A new architecture for a faster bar

Version 10 of the Meebo Bar delivers a modular architecture. The main benefits of this release are key performance improvements for publishers and a secure environment for user data and messaging.

1. Just-in-time minimal code delivery

One secret to making third-party plugins faster is to download the smallest amount of code needed at just the right time it is needed. A smaller initial payload means the bar can start up much faster. And deferred loading of functionality means faster download times on startup.

We deliver the minimal code just-in-time through a series of steps. First we segment our code into delivery packets that are organized around user functionality. For ads the code is deferred until the time that ads are rendered. For messaging the code for IM is downloaded shortly after the page finishes loading or when the user clicks to connect (whichever comes first). Second, when we build the bar we run our segmented code through Google's closure compiler to make the JavaScript code more efficient and reduce what is left to its most minimal size. Finally, we deliver our JavaScript code in a compressed format (gzip) decreasing the download size even further.

Using these techniques the initial JavaScript code is now 1/3 of its former size (57k) and the overall initial payload (JavaScript + CSS + embedded images) is less than half of its former size (<90k). What this means to you is the bar loads faster with less latency.

2. True persistent caching

What if we could take the code that gets loaded by the user when they visit a site with the bar and store all of that code locally? And what if the next time they visited **any** site with the Meebo Bar we could just pull the bar out of the browser instead of having to download it again? This is exactly what we have done with our *true* caching mechanism.

By utilizing local storage (an advanced feature in modern browsers) we can make all subsequent loads of the JavaScript for the bar be near zero in download and display

time. While browsers have long supported a caching mechanism, this functionality has deficiencies. Browser cache is much smaller than local storage, so items stored in browser cache get quickly cycled out of memory. This means that subsequent visits require code to be downloaded again.

With local storage, the user brings the bar with them to your site across all of your pages. You end up paying for the cost of the just-in-time delivery only once (if they visit your site first) or never (if they visited another site with the bar first).

3. Bundled image loading

The number of requests a plugin makes to load and render itself is an important factor in its overall performance. Some plugins have small download sizes but then require 40 or more requests to fetch CSS, button images, and graphical assets to render themselves. Here's the problem: only a finite number of requests can be served in parallel and each request carries with it network latency.

In version 10 of the bar we have employed a more advanced technique for loading images. All of our images (e.g., for buttons) are now bundled

within our CSS stylesheet as embedded data. Images piggyback on the existing request keeping the total number of requests low and fixed regardless of the number of images required.

4. Non-blocking code

Normally when you add JavaScript code to a page the browser will block the rest of the page from rendering while it downloads and processes the JavaScript. Other plugins have developed advanced techniques to allow JavaScript code to download without blocking page rendering. Unfortunately in some instances some browsers can still block the rendering of the page.

To solve this problem, our engineers went to work and pioneered a new technique that ensures that in all cases the browser will not block when loading the Meebo Bar code. We have shared this approach with the rest of the industry and are evangelizing it to other third-party plugins.

5. Faster rendering

In previous versions of the bar we took the standard approach of using images to create gradients, drop shadows, rounded corners, and other effects to

render the bar. The downside is images are downloaded one by one or in a harder to customize sprite format.

In this version of the bar we take a different approach. By rendering the supporting graphics with vector drawing technology (like SVG) we can draw the bar faster without additional HTTP requests and with a greater flexibility to customize the look and feel of the bar.

Security Sandbox

In addition to all of the performance improvements just outlined, the architecture of version 10 creates a security sandbox for the user's personal information and conversations to live within.

This means that whenever a user logs in to the bar all personal information and messaging is isolated from the site's code and vice versa. While not as demonstrable as performance, security is a key feature in this version of the bar.

The new Meebo Bar is a key step forward in terms of the quality it delivers to the user's site experience. It delivers all of the previous Meebo Bar functionality with even greater security and with a huge boost in speed.

by the numbers	payload size	# requests	later visits
	57k initial JavaScript payload	5 initial requests to load the bar	2 requests needed on later visits
	87k total initial payload	3 delayed requests after loading	0 amount of JavaScript needed on later visits
	1/3 smaller initial javascript than previous versions	0 additional requests for button images or graphical assets	