

Webstrates for the Future Web?

Kristian B Antonsen, Michel Beaudouin-Lafon, James Eagan, Clemens Nylandsted Klokmose, Wendy Mackay, Roman Rädle

▶ **To cite this version:**

Kristian B Antonsen, Michel Beaudouin-Lafon, James Eagan, Clemens Nylandsted Klokmose, Wendy Mackay, et al.. Webstrates for the Future Web?. ProWeb 2017 - Programming Technology for the Future Web, Apr 2017, Brussels, Belgium. pp.1. hal-01614236

HAL Id: hal-01614236
https://inria.hal.science/hal-01614236

Submitted on 11 Oct 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Webstrates for the future web?

Kristian B. Antonsen[1], Michel Beaudouin-Lafon[2, 3, 4, 7], James Eagan[5, 6, 7],
Clemens N. Klokmose[1], Wendy Mackay[4, 2, 3, 7], Roman Rädle[1]
Aarhus University[1], Univ. Paris-Sud[2], CNRS[3], INRIA[4], LTCI[5], Telecom ParisTech[6], Université Paris-Saclay[7]

Webstrates[1] [2] presents an alternative take on the future of web use and development. In Webstrates, changes to the Document Object Model (DOM) of webpages (called webstrates) are persisted across reloads and synchronized to other clients of the same webstrate. This includes changes to embedded JavaScript and CSS.

What previously only ran as native desktop applications are increasingly implemented as pure web applications (e.g., Microsoft Office 365 or Spotify). They provide a user experience that is similar to or even indistinguishable from that of their native counterparts, but with the benefits of being accessible from anywhere in the world.

Despite the open standards of HTML, CSS, and JavaScript, as well as users being able to inspect the DOM and application logic in the developer tools of a webbrowser, changes to either of the two are ephemeral and gone on webpage reload or subsequent webpage visits.

In Webstrates, there is technically no difference between the use, authoring and development of web content; it all happens within the context of the web browser by manipulating the DOM through JavaScript or even directly through the developer tools.

Webstrates was developed to prototype a reiteration of Kay and Goldberg's vision of interaction with computers as being interaction with personal dynamic media [1], but with an emphasis on shareability, thus we've referred to *shareable dynamic media* (SDM). The core principles of SDM are *"Malleability: users can appropriate their tools and documents in personal and idiosyncratic ways; Shareability: users can collaborate seamlessly on multiple types of data within a document, using their own personalized views and tools; and Distributability: tools and documents can move easily across different devices and platforms."* [2]

Webstrates can compose other webstrates via transclusion, using `iframes` and hereby allowing for application/document-like relationships between webstrates. We have used this to develop sophisticated collaborative applications where users can interact with the same document using personalized, asymmetric editors, and extend the functionality of these editors collaboratively at runtime (more details in the original paper). Programming in Webstrates can happen directly through the developer tools of the browser, through a dedicated code editor webstrate that enable editing scripts of a another webstrate using transclusion, through a legacy file-system integration with an extension that integrates with Visual Studio Code, and recently we have taken a literate programming based approach to development inspired by e.g. iPython Notebook. Since the original publication Webstrates has gone from being a proof-of-concept to a robust web framework used in multiple research projects internationally and by the paper authors for their daily activities (e.g. note taking, lectures, presentations, teaching material, and rapid prototyping).

[1] http://www.webstrates.net

The basic implementation of Webstrates consists of a server (Node.js) and client, both written in JavaScript. The DOM synchronization is implemented using OT over the server. The server persists ops in a MongoDB database, allowing a client to load the latest version of a webstrate or to restore from any previous version. Webstrates' version-control allows replay of the history of a webstrate in real-time or sequentially with a fixed time between each version down to single character input. Webstrates also includes a number of pragmatic features required to make it a usable web framework, which includes primitive authentication, an asset API, signaling, and a custom `transient` HTML element for non-persistent content.

Authentication uses the OAuth standard, i.e. supporting GitHub as auth provider. Users can be given individual read or write access, or no access at all, to a particular webstrate.

Webstrates implements an asset API that allows a client to upload files to the server. Thereby, images and other kinds of HTML5 supported media can be uploaded and linked as resource in the DOM. Assets are also version-controlled and restoring a particular version of webstrate will also restore assets in their proper version.

In addition to persistent DOM synchronization, Webstrates provides support for sending and subscribing to signals on DOM elements, allowing for clients to share volatile data that does not need to be persisted, like data with a high update frequency (sensors) or ephemeral data like cursor positions.

The custom `transient` HTML element will not be synchronized with other clients, including all of its descendant elements. The transient element is useful in scenarios where it is desired to avoid synchronization of particular DOM elements or subtrees of the DOM. For example, to create transient user interface elements like context menus, which should only be visible in the browser showing the menu, or rapidly changing visualizations.

With Webstrates we demonstrate one possible direction for the future web that we hope can inspire and foster discussion at the ProWeb 2017 workshop. We will gladly demonstrate Webstrates at the workshop and illustrate best programming practices for it. Further, we would like to discuss open issues (e.g. programming language support, collaborative programming, fine-grained permissions, handling large datasets, and other advances in the future of web programming) from which we believe future versions of Webstrates and its community will greatly benefit.

REFERENCES

[1] A. Kay and A. Goldberg. 1977. Personal Dynamic Media. *Computer* 10, 3 (March 1977), 31–41. DOI: http://dx.doi.org/10.1109/C-M.1977.217672

[2] Clemens N. Klokmose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: Shareable Dynamic Media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 280–290. DOI: http://dx.doi.org/10.1145/2807442.2807446