The development and integration of the PaintWeb paint tool in Moodle

- Marius and Mihai Şucan
- ”Aurel Vlaicu” University, Arad, Romania

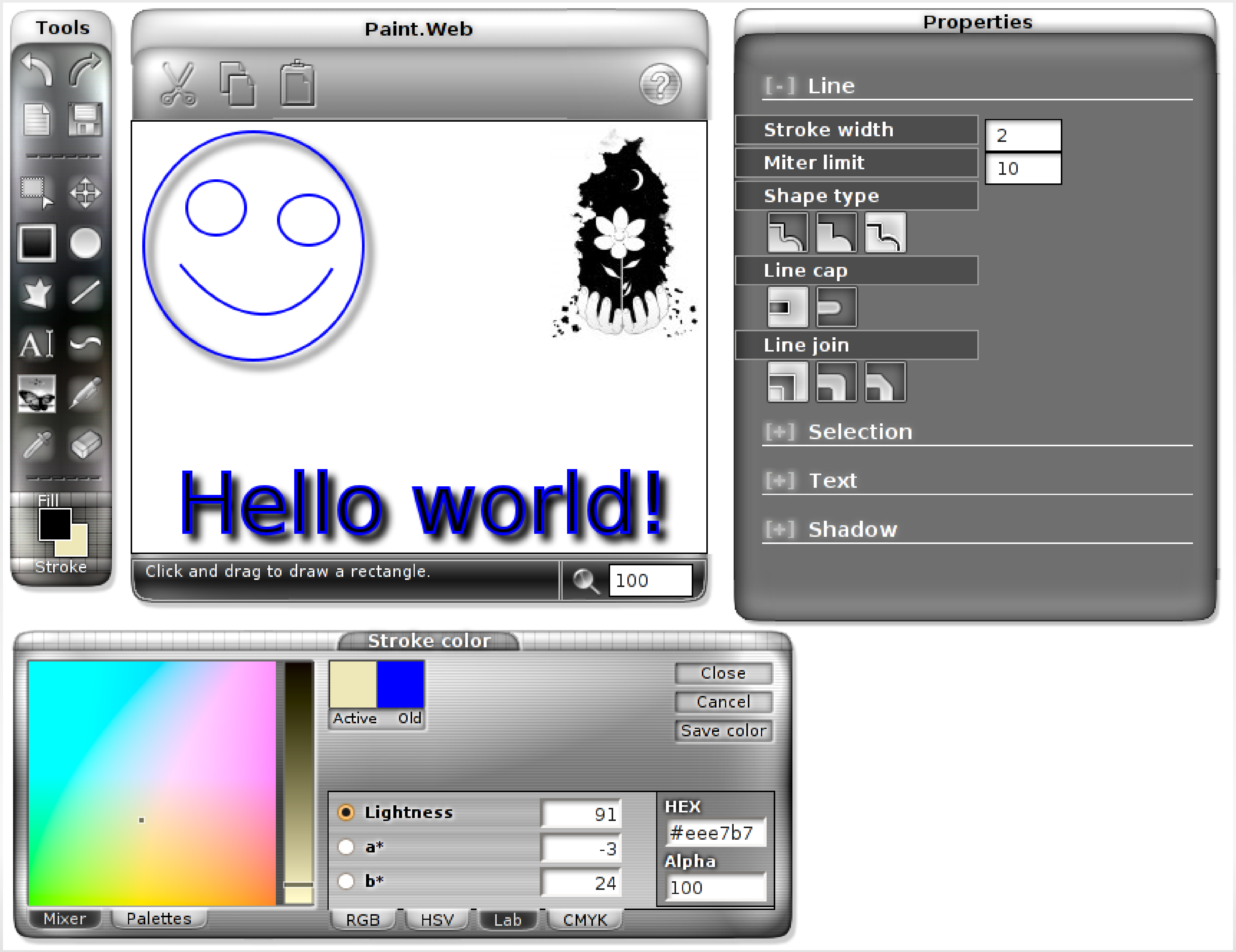
- A Google Summer of Code 2009 project
- Mentor: Martin Langhoff

August 2009

PaintWeb – The beginnings

- Last year I developed a Web application for online drawing. I published it as an open-source project, GPL v3, at Google Code.
- PaintWeb was a technical demo of the new HTML 5 Canvas API (the 2D context), together with some Web Forms 2.
- The result:
  - A single script of about 6000 lines of code, with no jsdoc-formatted comments.
  - Two tutorials on Canvas for Opera Software.

PaintWeb in november 2008



Click and drag to draw a rectangle.

Zoom: 100

[-] Line

Stroke width: 2

Miter limit: 10

Shape type

Line cap

Line join

[+] Selection

[+] Text

[+] Shadow

Lightness: 91

a*: -3

b*: 24

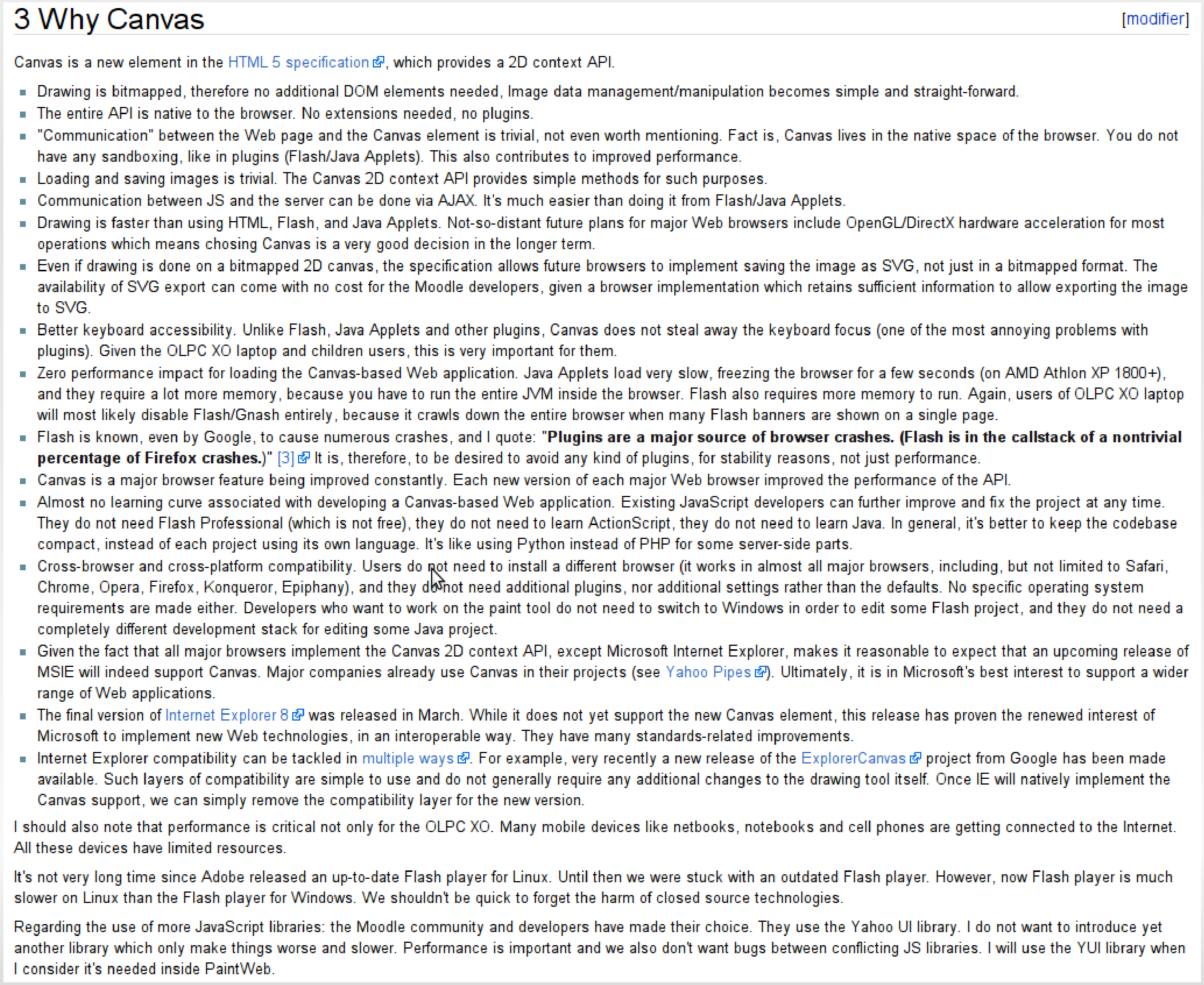
HEX: #eee7b7

Alpha: 100

Why PaintWeb?

- It began as a tech demo for tutorials on HTML 5 for Opera.
- It evolved beyond initial expectations.
  - I decided open-sourcing the code, in the hope of further improvements in the future.
- While it may not be too obvious, PaintWeb already surpasses the classical MS Paint in some ways.
- Originality: PaintWeb is the only open-source project which provides a Web application for online drawing in the form of a component, which uses HTML 5 Canvas.
  - There are attempts with Flash, Java, or even Canvas demos, but none as evolved as PaintWeb, and none having the same technologies.

The GSOC project proposal



3 Why Canvas

[modifier]

Canvas is a new element in the HTML 5 specification, which provides a 2D context API.

- Drawing is bitmapped, therefore no additional DOM elements needed, Image data management/manipulation becomes simple and straight-forward.
- The entire API is native to the browser. No extensions needed, no plugins.
- "Communication" between the Web page and the Canvas element is trivial, not even worth mentioning. Fact is, Canvas lives in the native space of the browser. You do not have any sandboxing, like in plugins (Flash/Java Applets). This also contributes to improved performance.
- Loading and saving images is trivial. The Canvas 2D context API provides simple methods for such purposes.
- Communication between JS and the server can be done via AJAX. It's much easier than doing it from Flash/Java Applets.
- Drawing is faster than using HTML, Flash, and Java Applets. Not-so-distant future plans for major Web browsers include OpenGL/DirectX hardware acceleration for most operations which means chosing Canvas is a very good decision in the longer term.
- Even if drawing is done on a bitmapped 2D canvas, the specification allows future browsers to implement saving the image as SVG, not just in a bitmapped format. The availability of SVG export can come with no cost for the Moodle developers, given a browser implementation which retains sufficient information to allow exporting the image to SVG.
- Better keyboard accessibility. Unlike Flash, Java Applets and other plugins, Canvas does not steal away the keyboard focus (one of the most annoying problems with plugins). Given the OLPC XO laptop and children users, this is very important for them.
- Zero performance impact for loading the Canvas-based Web application. Java Applets load very slow, freezing the browser for a few seconds (on AMD Athlon XP 1800+), and they require a lot more memory, because you have to run the entire JVM inside the browser. Flash also requires more memory to run. Again, users of OLPC XO laptop will most likely disable Flash/Gnash entirely, because it crawls down the entire browser when many Flash banners are shown on a single page.
- Flash is known, even by Google, to cause numerous crashes, and I quote: "**Plugins are a major source of browser crashes. (Flash is in the callstack of a nontrivial percentage of Firefox crashes.)**" [3] It is, therefore, to be desired to avoid any kind of plugins, for stability reasons, not just performance.
- Canvas is a major browser feature being improved constantly. Each new version of each major Web browser improved the performance of the API.
- Almost no learning curve associated with developing a Canvas-based Web application. Existing JavaScript developers can further improve and fix the project at any time. They do not need Flash Professional (which is not free), they do not need to learn ActionScript, they do not need to learn Java. In general, it's better to keep the codebase compact, instead of each project using its own language. It's like using Python instead of PHP for some server-side parts.
- Cross-browser and cross-platform compatibility. Users do not need to install a different browser (it works in almost all major browsers, including, but not limited to Safari, Chrome, Opera, Firefox, Konqueror, Epiphany), and they do not need additional plugins, nor additional settings rather than the defaults. No specific operating system requirements are made either. Developers who want to work on the paint tool do not need to switch to Windows in order to edit some Flash project, and they do not need a completely different development stack for editing some Java project.
- Given the fact that all major browsers implement the Canvas 2D context API, except Microsoft Internet Explorer, makes it reasonable to expect that an upcoming release of MSIE will indeed support Canvas. Major companies already use Canvas in their projects (see Yahoo Pipes). Ultimately, it is in Microsoft's best interest to support a wider range of Web applications.
- The final version of Internet Explorer 8 was released in March. While it does not yet support the new Canvas element, this release has proven the renewed interest of Microsoft to implement new Web technologies, in an interoperable way. They have many standards-related improvements.
- Internet Explorer compatibility can be tackled in multiple ways. For example, very recently a new release of the ExplorerCanvas project from Google has been made available. Such layers of compatibility are simple to use and do not generally require any additional changes to the drawing tool itself. Once IE will natively implement the Canvas support, we can simply remove the compatibility layer for the new version.

I should also note that performance is critical not only for the OLPC XO. Many mobile devices like netbooks, notebooks and cell phones are getting connected to the Internet. All these devices have limited resources.

It's not very long time since Adobe released an up-to-date Flash player for Linux. Until then we were stuck with an outdated Flash player. However, now Flash player is much slower on Linux than the Flash player for Windows. We shouldn't be quick to forget the harm of closed source technologies.

Regarding the use of more JavaScript libraries: the Moodle community and developers have made their choice. They use the Yahoo UI library. I do not want to introduce yet another library which only make things worse and slower. Performance is important and we also don't want bugs between conflicting JS libraries. I will use the YUI library when I consider it's needed inside PaintWeb.

Why Canvas?

- A single element: <canvas>.
  - The API is in the DOM, native to the Web browser, no extensions/plugins needed.
- Fast bitmap drawing on a 2D surface, without any DOM elements for each path or shape.
- Better keyboard accessibility.
  - It is well known that Flash, Java and plugins in general have issues with the window focus.
- Almost no impact for page load performance.
- Much better browser stability.
  - Plugins (especially Flash) cause the highest number of crashes in Web browsers.

PaintWeb – What I did this summer

- Complete code reorganization.
- Performance optimisations for the OLPC XO.
- Defined an API for PaintWeb extensions.
- General improvements for features.
- A new interface.
- Packaging.
- TinyMCE 3 integration using a plugin.
- Moodle 1.9 and 2.0 integration.
- Documentation.

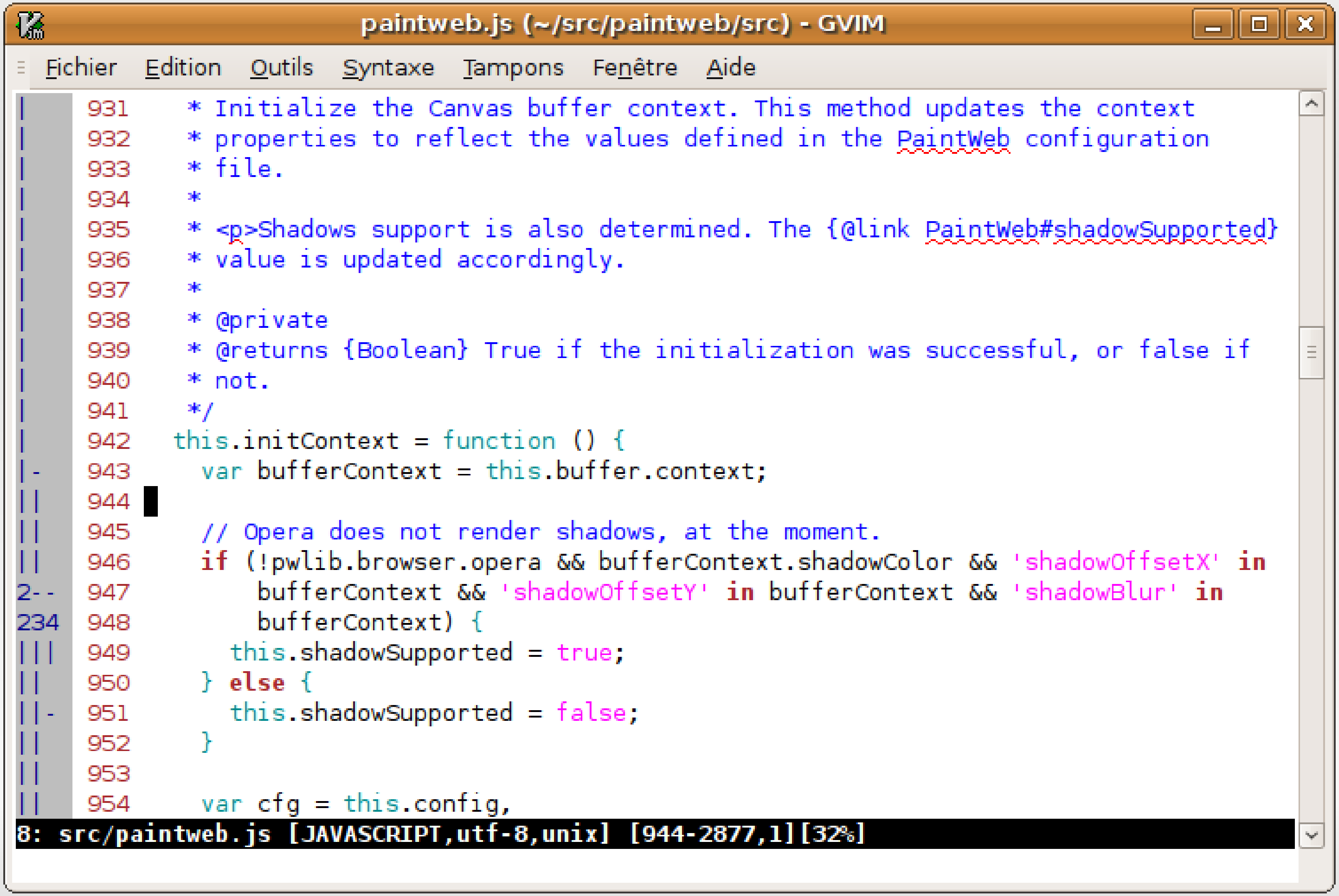
PaintWeb – Code reorganization

- Separate files for each drawing tool, extension and language.
- New configuration file.
- More formal code, stricter guidelines, with more consistency in the naming of functions and variables.
- Jsdoc-formatted comments for all of the methods and properties defined in PaintWeb.
- Separate files for the user interface: JavaScript, CSS and XHTML.

PaintWeb – Package structure

- **build/** - The packaged PaintWeb build.
- **demos/** - PaintWeb integration demos.
- **docs/** - documentation.
- **ext/** - scripts for integration into external projects.
- **scripts/** - scripts used for packaging.
- **src/** - the PaintWeb source code:
  - **extensions/**, **includes/**, **interfaces/**, **lang/** şi **tools/**;
  - **paintweb.js** – the main script.
- **tests/** - various tests performed for PaintWeb.

PaintWeb – Source code sample


paintweb.js (~/src/paintweb/src) - GVIM
Fichier Edition Outils Syntaxe Tampons Fenêtre Aide

```
     * Initialize the Canvas buffer context. This method updates the context
     * properties to reflect the values defined in the PaintWeb configuration
     * file.
     *
     * <p>Shadows support is also determined. The {@link PaintWeb#shadowSupported}
     * value is updated accordingly.
     *
     * @private
     * @returns {Boolean} True if the initialization was successful, or false if
     * not.
     */
    this.initContext = function () {
      var bufferContext = this.buffer.context;

      // Opera does not render shadows, at the moment.
      if (!pwlib.browser.opera && bufferContext.shadowColor && 'shadowOffsetX' in
          bufferContext && 'shadowOffsetY' in bufferContext && 'shadowBlur' in
          bufferContext) {
        this.shadowSupported = true;
      } else {
        this.shadowSupported = false;
      }

      var cfg = this.config,
8: src/paintweb.js [JAVASCRIPT,utf-8,unix] [944-2877,1][32%]
```

PaintWeb – About the OLPC XO

- The One Laptop Per Child School Server project includes Moodle.
  - PaintWeb must run fine on the OLPC XO-1.
- Technical details about the OLPC XO-1 laptop:
  - AMD processor, 433 Mhz, x86 compatible, 64 KB L1 cache, 128 KB L2 cache;
  - DRAM memory, 256 MB (dual DDR333 166 Mhz);
  - Storage: 1 GB or 2 GB SLC NAND flash;
  - A special dual-layer display 7.5 inch, TFT.
- The XO runs Linux, based on Fedora 9.
  - The default Web browser is Python-based and it embeds Gecko 1.9.0.

PaintWeb – Performance optimisations

- **The problem:** Images and texts are scaled up on the OLPC XO.
- In `about:config` the `layout.css.dpi` value is set to 134, instead of 96 (the default on PCs).
- **The solution:** cancel Canvas scaling.
- Say the DPI is 200, say `canvas.width=200` px, which is rendered at 400 px. We cancel scaling with `canvas.style.width=”100px”`.
- The OLPC XO can be detected with browser sniffing. This makes PaintWeb run much faster.
  - CSS 3 Media queries are used with Gecko 1.9.1+.

PaintWeb – On performance

- Myth: JavaScript is fundamental to the performance of common Web applications.
- Web applications are first affected by the browser rendering performance.
  - CSS+HTML, transparent images, etc.
- PaintWeb is slowed down by the rendering of CSS+HTML and that of Canvas itself.
  - PaintWeb is not (yet) affected by JS perf.
- As said, the OLPC XO performance was affected by Canvas scaling and by a Canvas background image.

PaintWeb – New API

- Register extensions.
  - `PaintWeb.extensionRegister('id').`
- Register drawing tools.
  - `PaintWeb.toolRegister('id').`
- Register commands.
  - `PaintWeb.commandRegister('undo', undoFn).`
- Add application event listeners.
  - `PaintWeb.events.add('imageSave', evFn).`
- … and more (see the documentation).

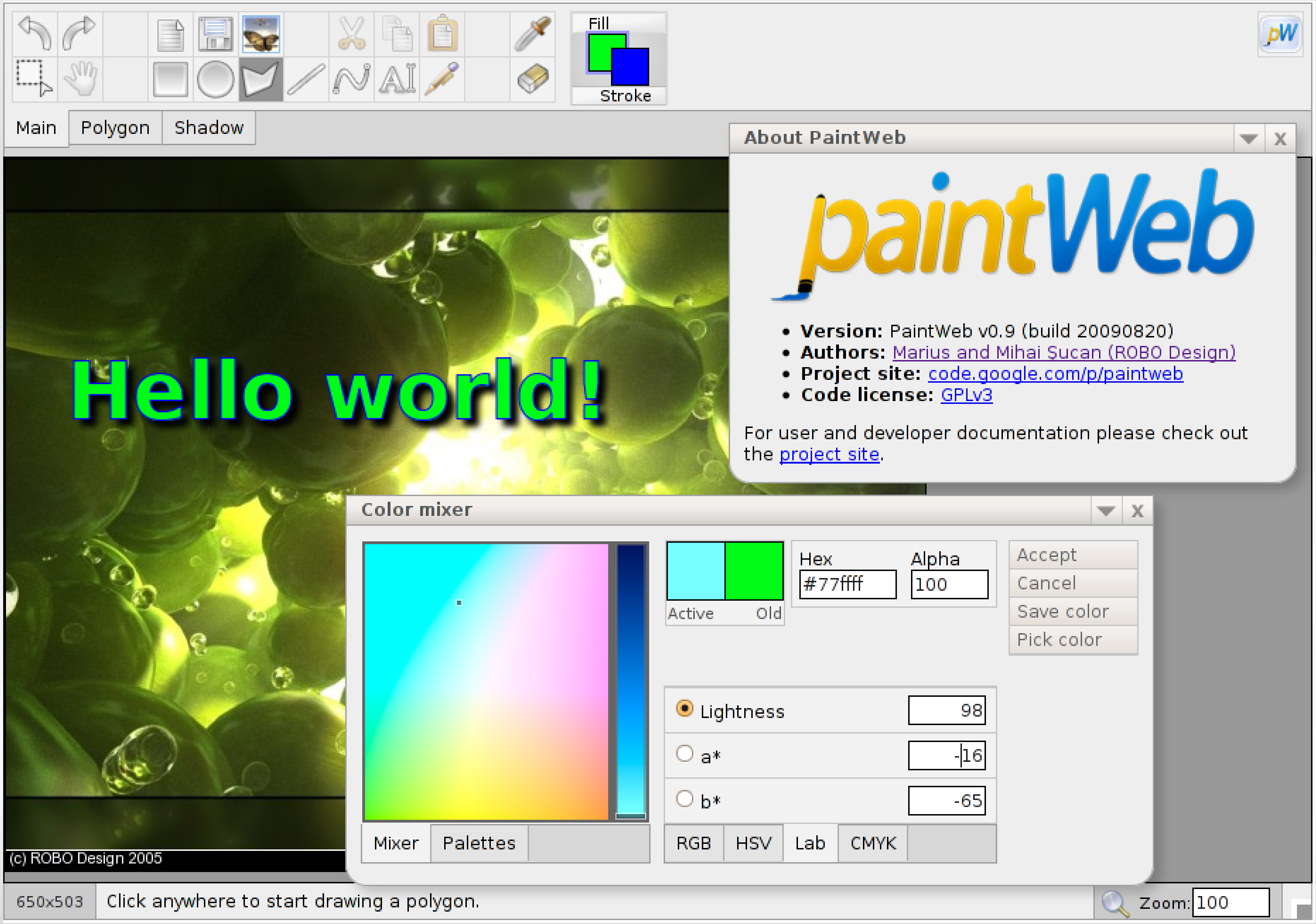
PaintWeb – Improved functionality

- A new extension: MouseKeys. Users can draw without a mouse.
- Reimplemented the support for keyboard shortcuts, which now enables much better browser compatibility.
- Many improvements and fixes for the selection tool.
- A new “Hand” tool for dragging the image inside the viewport.
- Improvements for the “Eraser” tool.
- Better support for big images, say 6000 x 6000 px.
- Improved browser compatibility for the text tool.

PaintWeb – The new interface

- Easy to modify, only three files: a CSS, an XHTML and a JavaScript.
- Dynamically loaded, on user request.
- The HTML code includes proprietary attributes:
  - `<p data-pwCommand="imageSave">Save image</p>`
  - `<p data-pwTool="selection">Selection</p>`
  - `<input data-pwConfig="line.lineWidth" type="number" min="1" max="100" value="1">`
- Much better keyboard accessibility.
- Contextual user interface: options show up dynamically, when needed.

PaintWeb – Screenshot



About PaintWeb

- **Version:** PaintWeb v0.9 (build 20090820)
- **Authors:** Marius and Mihai Şucan (ROBO Design)
- **Project site:** code.google.com/p/paintweb
- **Code license:** GPLv3

For user and developer documentation please check out the project site.

Color mixer

Active | Old

Hex: #77ffff
Alpha: 100

Accept
Cancel
Save color
Pick color

[✓ Lightness] 98
a* -16
b* -65

Mixer | Palettes

RGB | HSV | Lab | CMYK

650x503 Click anywhere to start drawing a polygon.

Zoom: 100

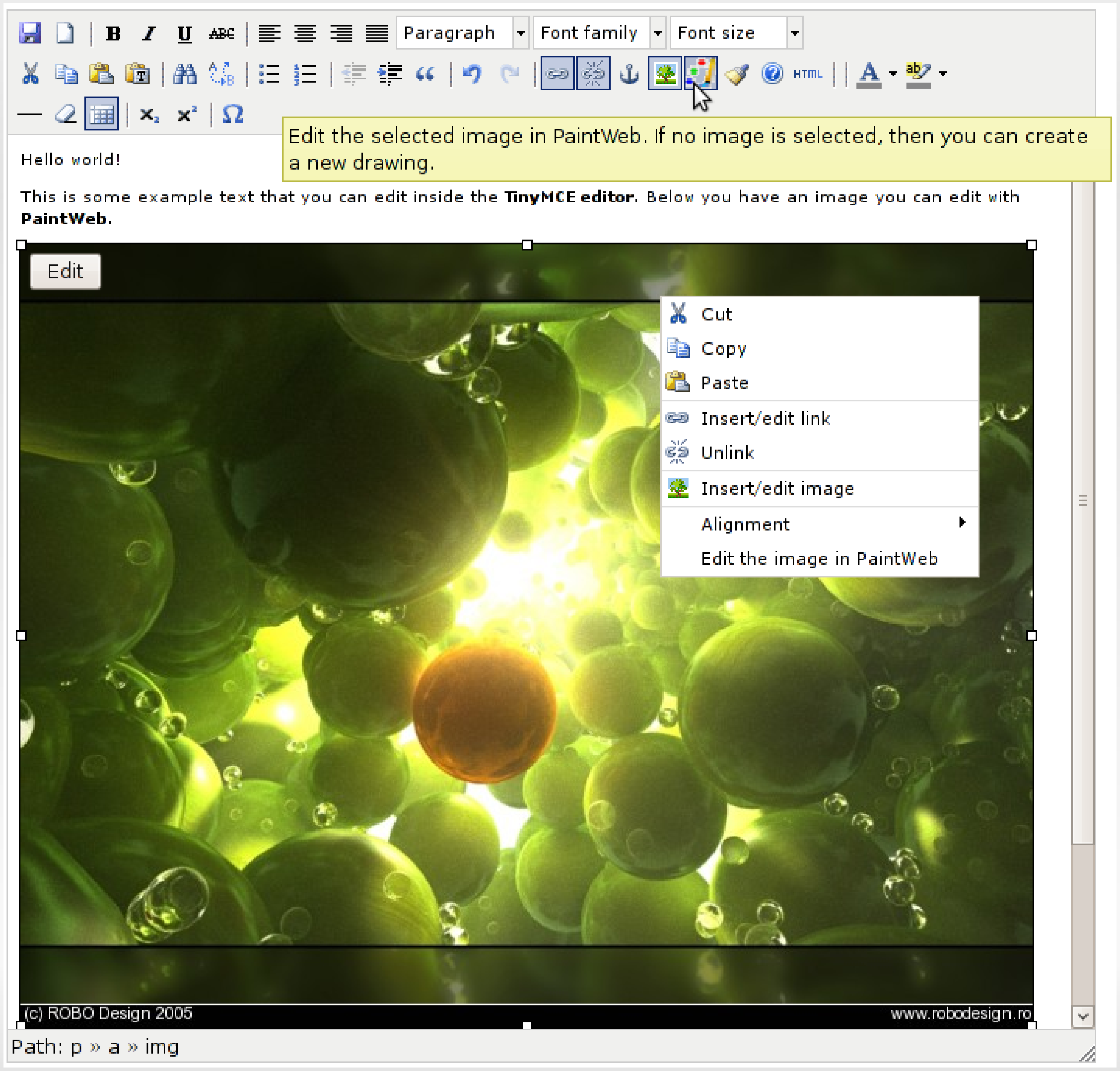
PaintWeb – Packaging

- Code packaging is driven by a Makefile and some scripts written in java, bash and PHP.
- YUICompressor is used to compress JS and CSS.
- I wrote a PHP script which does image inlining using data URLs inside CSS.
- Jsdoc-toolkit is used to generate the API reference.
- Results:
  - From 70 files down to only 6 files.
  - From 700 KB to 460 KB or even 140 KB (with gzipping).
  - Initialization is under a second with a broadband connection.

PaintWeb – TinyMCE integration

- One plugin, easy to install (copy/paste folder).
- A button on the tool bar:
  - If no image is selected: a new image is created and then edited with PaintWeb.
  - If an image is selected: PaintWeb allows you to edit the image.
- An “Edit” button dynamically shown on top of the selected image.
  - You can also double-click on the image to edit it.
- A context menu (right click) item which allows you to start PaintWeb.

PaintWeb in TinyMCE: screenshot



Edit the selected image in PaintWeb. If no image is selected, then you can create a new drawing.

Hello world!

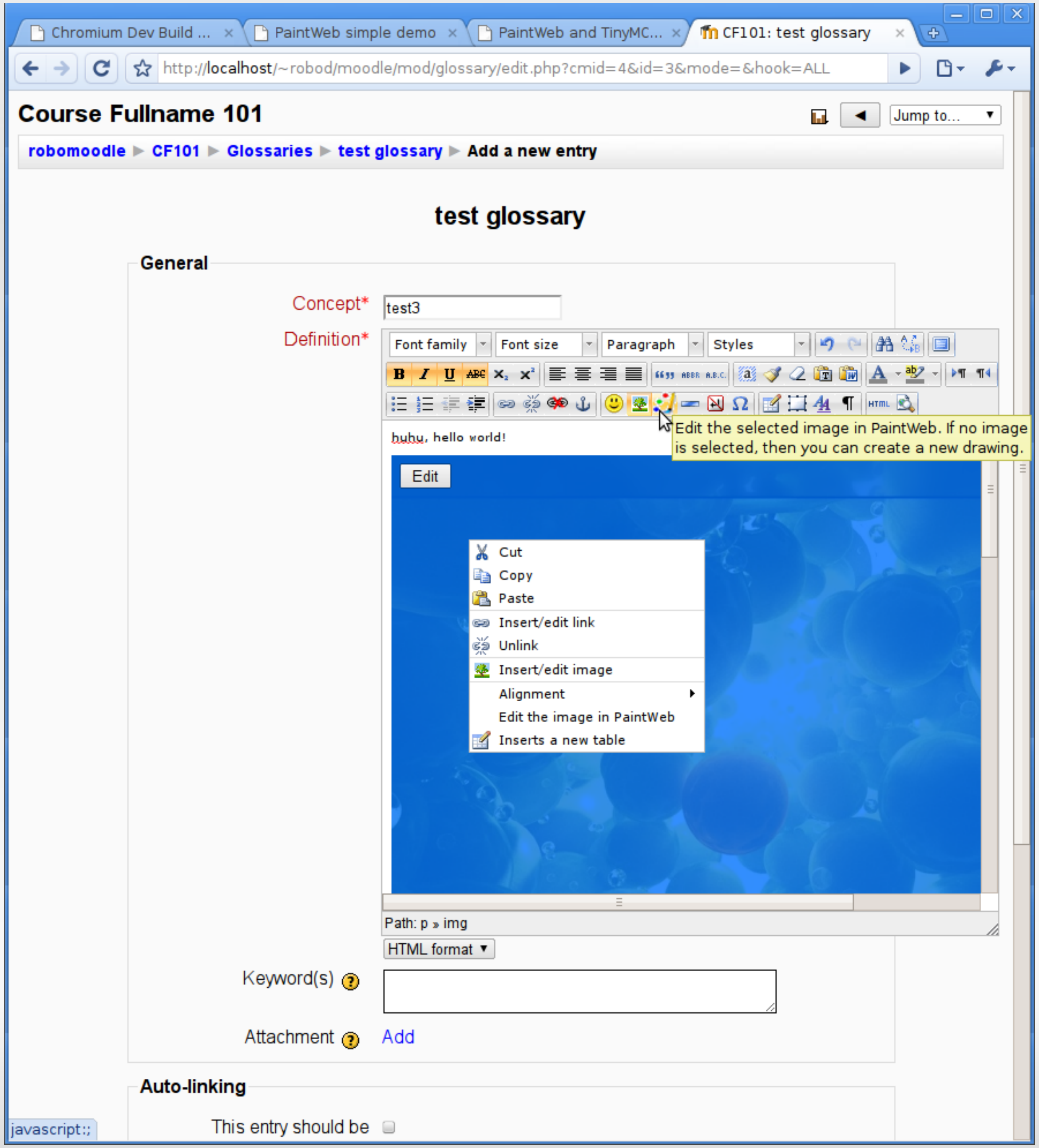
This is some example text that you can edit inside the **TinyMCE editor**. Below you have an image you can edit with **PaintWeb**.

Path: p » a » img

PaintWeb – Moodle 2.0 integration

- Moodle 2.0 is under heavy development.
- Lots of improvements, including a new File API.
- It uses TinyMCE 3 by default.
- PaintWeb is integrated into TinyMCE and images are saved using the new File API.
- PaintWeb has its own extension which deals with saving images inside Moodle.
- Files attached to `<textareas>` are stored in a user_draft file area. During form submission, files are moved to a permanent file area.

PaintWeb – TinyMCE 3 in Moodle 2



Course Fullname 101

robomoodle ▶ CF101 ▶ Glossaries ▶ test glossary ▶ Add a new entry

test glossary

General

Concept* test3

Definition*

Edit the selected image in PaintWeb. If no image is selected, then you can create a new drawing.

huhu, hello world!

Path: p » img

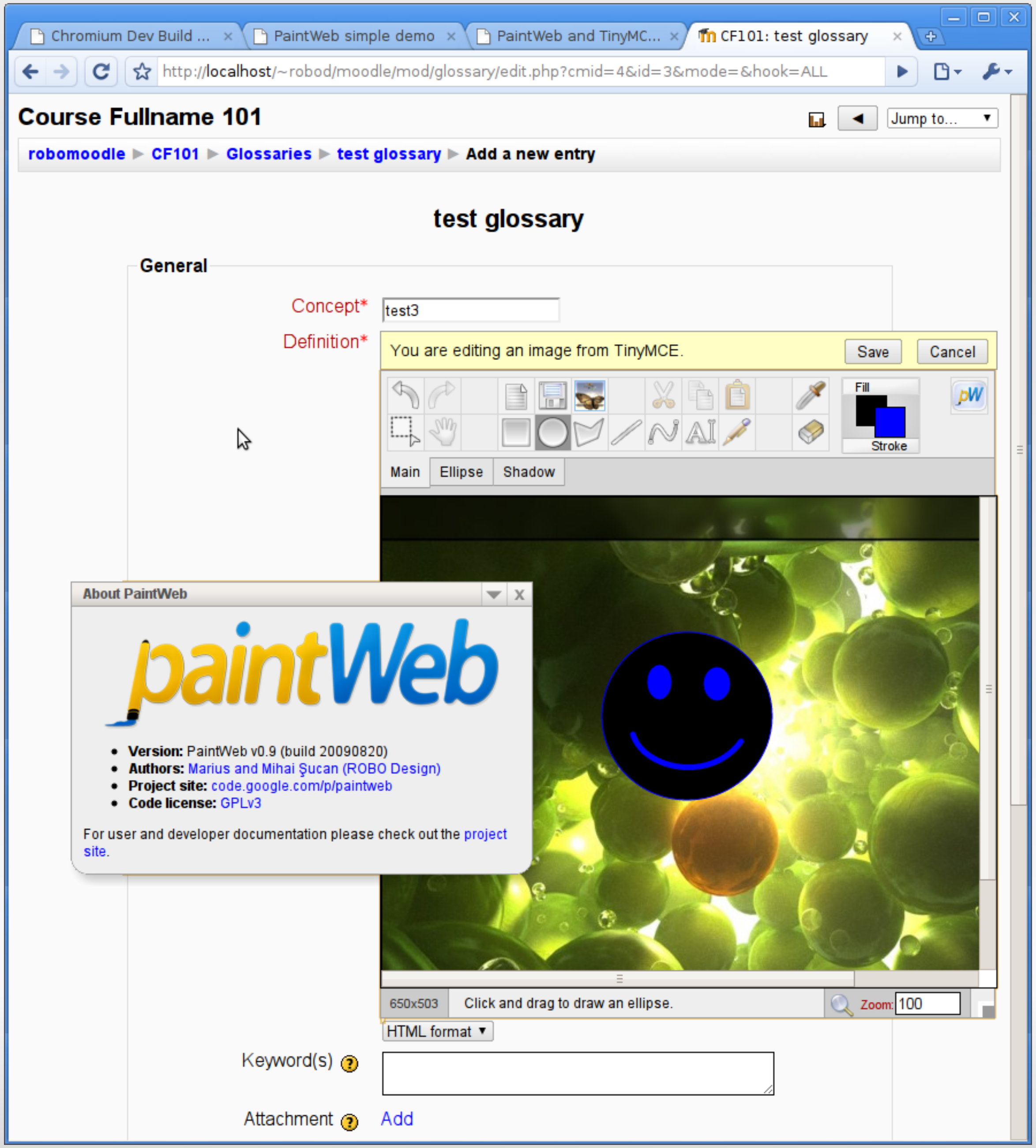
Keyword(s)

Attachment

Auto-linking

This entry should be

PaintWeb – PaintWeb in Moodle 2



Course Fullname 101

robomoodle ▶ CF101 ▶ Glossaries ▶ test glossary ▶ Add a new entry

test glossary

General

Concept* test3

Definition*

You are editing an image from TinyMCE.

About PaintWeb

- **Version:** PaintWeb v0.9 (build 20090820)
- **Authors:** Marius and Mihai Şucan (ROBO Design)
- **Project site:** code.google.com/p/paintweb
- **Code license:** GPLv3

For user and developer documentation please check out the project site.

650x503 Click and drag to draw an ellipse. Zoom: 100

Keyword(s)

Attachment

PaintWeb – Moodle 1.9 integration

- Moodle 1.9 is the current stable version. Only bug fixes are accepted.
- It lacks a sufficiently evolved File API.
- It uses HTMLArea by default, and it also includes TinyMCE 2 – both are quite outdated.
- Martin Langhoff published patches for TinyMCE 3 integration. I updated them to the latest version.
  - PaintWeb can now be used in TinyMCE 3.
- Image save implementation is not yet complete. We have to manually deal with obsolete images, and backup/restore courses.
- This work will be available as contrib-patches for anyone.
- The patches will be used by the OLPC school server.

PaintWeb – Documentation

- API references for the entire code, including the drawing tools, extensions, commands, GUI, and the application events.
- Usage in TinyMCE.
- How to build PaintWeb.
- How to extend PaintWeb.
- Some tutorials published by Opera:
  - HTML 5 Canvas - the basics.
  - Creating an HTML 5 canvas painting application.
  - Keyboard accessibility in Web applications.

PaintWeb – Compatibility

- Web browsers: Opera, Konqueror, Google Chrome, Firefox and Safari.
- Render engines: Presto, KHTML, Webkit and Gecko.
- Microsoft Internet Explorer does not support Canvas.
  - We will probably use excanvas for compatibility.
- The text tool does not work in Opera.
  - The attempt to use SVG:Text has failed.
- Shadows do not render in Opera.
  - Neither in Chrome, but they work in Safari and Firefox.
- There are many small and big bugs in Web browsers.

PaintWeb – Conclusions

- The Moodle community is very friendly and welcoming.
  - Thanks to Martin Langhoff (mentor), Helen Foster (GSOC admin at Moodle), Robert O'Callahan (Mozilla) şi Olli Savolainen (GSOC student at Moodle).
- GSOC was an experience from which I learned lots of new things and I met many great people.
- GSOC can be a “launch pad” for great projects, not just a summer job.
  - I will continue to collaborate with Moodle.
- This summer should be just the beginning.
  - … for a constantly evolving project: PaintWeb.

PaintWeb – What is next

- Polish PaintWeb.
  - I have a sufficiently big TODO list for this purpose.
- Polish Moodle 2.0 integration.
  - Complete Moodle 1.9 integration.
- Improve the user interface.
  - Make a simpler version for kids, for OLPC.
- Release a stable version of PaintWeb.
- The next PaintWeb version:
  - Filters, layers, maybe SVG and more.

PaintWeb – The plans from 2008

- Important features:
  - Filters, layers, gradients, patterns, “smart objects” and maybe SVG.
- Animated interface: CSS Animations.
- Integration into bigger Web applications.
- More HTML 5:
  - Offline Web application (Opera Unite, Adobe Air);
  - Client-side storage and database storage;
  - Drag and drop;
  - Server-sent events.

PaintWeb – Thanks

- Thank you for your time.
- To test the Web application go to:
  - http://code.google.com/p/paintweb

WE BRING YOU THE FUTURE

www.robodesign.ro