

# Sun Labs Lively Kernel

Release Notes  
Version 0.7  
October 4, 2007

## Overview

Sun Labs Lively Kernel is a new web programming environment developed at Sun Microsystems Laboratories. The Lively Kernel supports desktop-style applications with rich graphics and direct manipulation capabilities, but without the installation or upgrade hassles that conventional desktop applications have. The system is written entirely in the JavaScript programming language – a language supported by all the web browsers – with the intent that the system can run in commercial web browsers without installation or any plug-in components. The system leverages the dynamic characteristics of the JavaScript language to make it possible to create, modify and deploy applications on the fly, using tools built into the system itself. In addition to its application execution capabilities, the Lively Kernel system can also function as an integrated development environment (IDE), making the whole system self-sufficient and able to improve and extend itself dynamically.

## Release History

<u>Version</u>	<u>Description</u>
0.7	The first public open source release of the Lively Kernel.

## Contents of this ZIP file

In this ZIP file you find all the source code files of the Lively Kernel.  
A quick summary of the files is provided below.

<u>File</u>	<u>Description</u>
<i>index.xhtml</i>	The XHTML file that launches and Lively Kernel and loads all the other files.
<i>prototype.js</i>	The Prototype library that we use as part of the system ( <a href="http://www.prototypejs.org/">http://www.prototypejs.org/</a> ).
<i>fontinfo.js</i>	Contains code for mapping fonts onto the underlying web browser.
<i>svgtext-compat.js</i>	Contains code for mapping text support onto the underlying SVG graphics engine.
<i>Core.js</i>	Contains the core Lively Kernel features, low-level porting interface and the core Morhic UI framework.
<i>Text.js</i>	All the text-related functionality of the Lively Kernel.
<i>Widgets.js</i>	All the widget definitions for the Morhic UI framework.
<i>Network.js</i>	All the network-related functionality of the Lively Kernel.
<i>Storage.js</i>	All the storage-related functionality of the Lively Kernel.
<i>Tools.js</i>	Contains various tools such as the class browser, object inspector, style editor, etc.
<i>Examples.js</i>	Contains all the sample applications.
<i>Main.js</i>	Contains the JavaScript code that initializes the sample applications.
<i>definitions.svg</i>	Contains embedded resource definitions that are included in the system when it starts.
<i>defaultconfig.js</i>	Defines the default system configuration options for the Lively Kernel.

In addition to the files listed above, the user can provide a '*localconfig.js*' file that may override the default system configuration options defined in '*defaultconfig.js*'.

**Note.** The Lively Kernel is a “zero-installation” system, i.e., the user does not actually install anything or deal with any of the aforementioned files directly. Instead, the necessary files are loaded automatically by the web browser when the Lively Kernel is started. The loading of the files is performed by the '*index.xhtml*' file.

**Important.** Do not assume that the file structure will remain the same in the future. The future releases of the Lively Kernel may use a different file structure.

## Supported APIs

Since the Lively Kernel runs in an ordinary web browser and leverages the JavaScript virtual machine that is already part of the web browser, the system allows the developers to use all those JavaScript APIs that are provided by the underlying JavaScript environment.

Generally speaking, the Lively Kernel API set consists of the following three APIs:

- Core JavaScript APIs (see [http://developer.mozilla.org/en/docs/Core\\_JavaScript\\_1.5\\_Reference](http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference))
- Asynchronous HTTP networking via XMLHttpRequest and the Prototype library
- The Lively Kernel APIs

The Core JavaScript APIs and asynchronous HTTP support is provided by the web browser. The APIs specific to the Lively Kernel are described below.

## Application Programming Interfaces (APIs) Specific to the Lively Kernel

The majority of JavaScript classes defined by the Lively Kernel are related to a user interface framework called *Morphic*. Morphic is a UI framework that supports composable graphical objects, along with the machinery required to display and animate these objects, handle user inputs, and manage underlying system resources such as displays, fonts and color maps. A key goal of Morphic is to make it easy to construct and edit interactive graphical objects, both by direct manipulation and from within programs. The Morphic user interface was developed originally for the Self system (<http://research.sun.com/self>), but it became popular later also as part of the Squeak system (<http://www.squeak.org>).

From the programmer's viewpoint, every visual object in the Lively Kernel is a *morph*. A morph is a visual structure that contains functionality for managing the coordinates, boundaries and other visual characteristics of an object, as well as the necessary functionality for responding to events and external requests such as moving, grabbing, resizing, rotating, and so on.

All the morphs in the Lively Kernel live in a *world*. A Morphic world is a visual container – sort of like a fully interactive, graphical web page or Wiki – that can be manipulated directly by the users. Before a morph becomes active visible, it needs to be added to a world.

In the current version of the Lively Kernel, we support the following types of user-level morphs:

<u>Class Name</u>	<u>Description</u>
<i>Morph</i>	The Morph base class
<i>ButtonMorph</i>	Simple button
<i>ImageButtonMorph</i>	Button with an associated image
<i>ImageMorph</i>	Image object
<i>IconMorph</i>	Image object representing an icon
<i>TextMorph</i>	An object representing formatted text
<i>CheapListMorph</i>	Simple textual list supporting selection, etc.
<i>MenuMorph</i>	Popup menu
<i>TextBox</i>	Simple text box
<i>ClipMorph</i>	Clipping view
<i>WindowMorph</i>	Full-fledged window object
<i>TabbedPanelMorph</i>	Tabbed window/panel
<i>TitleBarMorph</i>	An object representing a window title bar
<i>PanelMorph</i>	Simple panel
<i>ScrollPane</i>	Scrollable panel
<i>ListPane</i>	A scrollpane containing a textual list ( <i>CheapListMorph</i> )
<i>TextPane</i>	A scrollable text pane
<i>SliderMorph</i>	Scroll bar object
<i>ColorPickerMorph</i>	Color picker panel

<u>Class Name</u>	<u>Description</u>
<i>SelectionMorph</i>	Selection tray object (for selecting multiple objects)
<i>HandMorph</i>	An object representing a hand (cursor) in a Morphic world
<i>PasteUpMorph</i>	A visual container onto which other Morphs can be pasted
<i>WorldMorph</i>	A Morphic world/subworld
<i>LinkMorph</i>	Hyperlink between Morphic worlds

In addition to the Morph classes summarized above, the following classes are frequently needed by the Lively Kernel programmers:

<u>Class Name</u>	<u>Description</u>
<i>Point</i>	An object that represents a two-dimensional point
<i>Rectangle</i>	An object that represents a rectangle
<i>Color</i>	An object that represents a color value
<i>LinearGradient</i>	An object representing a linear gradient (“sliding”) color
<i>RadialGradient</i>	An object representing a radial gradient (“sliding”) color
<i>StipplePattern</i>	An object representing a stipple pattern
<i>Transform</i>	Matrix transformations for object scaling, rotation, etc.
<i>DisplayObject</i>	The low-level interface between the Morphic UI framework and the underlying browser graphics engine
<i>Shape</i>	A base class for defining the underlying shape of a morph
<i>RectShape</i>	Rectangle shape
<i>EllipseShape</i>	Ellipse/circle shape
<i>PolygonShape</i>	Polygon shape
<i>PolylineShape</i>	Polyline shape
<i>PathShape</i>	Path shape

For further information on the APIs, refer to the Lively Kernel web site:

<http://research.sun.com/projects/lively>

More documentation will be available on the web site later.

**Important.** The APIs summarized above are still under development and may change significantly in the future.