Overview:
Wakanda Architecture

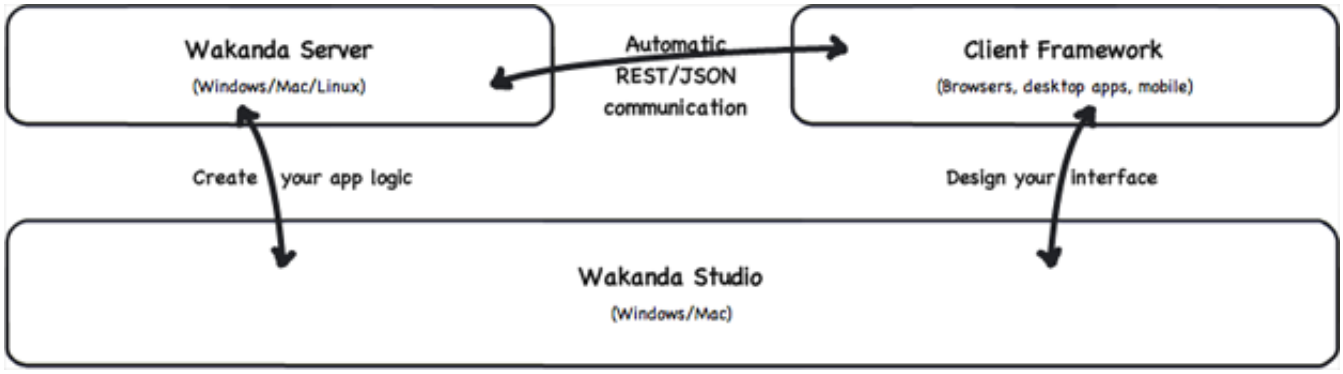
This preliminary documentation does not reflect the final content and design.

Wakanda Architecture

Wakanda is made up of three main components: Wakanda Server, Wakanda Studio, and the Wakanda Client Framework. (For a more general overview of Wakanda, please see **What is Wakanda?**)

Wakanda is so easy to use that you can get it up and running right away. Understanding its architecture will help you further develop high performance web applications for your organization. This document will review Wakanda's three parts, their primary components, and how they relate to one another.

Let's quickly look at a schema to put the platform in context.



For the moment, try to think of the Wakanda platform in an abstract way:

You aren't necessarily looking at multiple machines, but various layers in an application stack. These layers can even co-exist on a single machine, which is the way you will most likely begin using Wakanda.

Your development studio, test server, and the browser(s) with which you access your application can all be in one place. (Naturally, once you deploy your solution, you can also use dedicated machines and devices.)

Wakanda Server

Wakanda Server is made up of several parts:

- the datastore – housing all your application data
- the datastore engine – containing the datastore classes you've defined and JavaScript you've written for the application's business logic
- HTTP server – communicating with the outside world via JSON/REST

One of the key points of Wakanda's datastore is that it's very, very fast. In our tests, we have found that an item can be queried from among a billion objects in a matter of milliseconds. This efficiency comes from the architecture of the datastore itself: It contains all the data as well as the model, describing the datastore classes that define how the data is structured.

You can access these classes and data via the Wakanda datastore. Wakanda Server makes the datastore available as a series of JavaScript objects and functions, thus allowing you to easily manage datastore classes, entities and collections of entities.

To fully understand how to use Wakanda Server, you can read the **Datastore** documentation and the **Global Application** documentation as well as the documentation on **Using Datastore Class Events**.

The HTTP server portion of Wakanda is not only a powerful web server, but it also allows you to serve numerous simultaneous client connections. In addition to serving up web pages (in the form of HTML, CSS, JavaScript and so on), the server sends and receives application data via JSON/REST. This efficient means of communication is less verbose because less data is moving across the network. The HTTP server has the added advantage of being recognized by all

browsers on different devices – desktop, mobile or tablet – without any plug-ins.

Because the web server is so closely connected to the datastore and the server-side application logic, interactions are optimized and ultimately more intelligent, loading data only when necessary.

One of the reasons the server reacts so quickly is its support for HTTP request handlers (see documentation) and the RESTful architecture. Without the need to load the entire framework on the client side, the browser transmits a single request to the server to access the business logic and returns the corresponding data.

Added together with a very efficient cache, the eventual load on the client browser is very light, resulting in high-speed performance that rivals native applications.

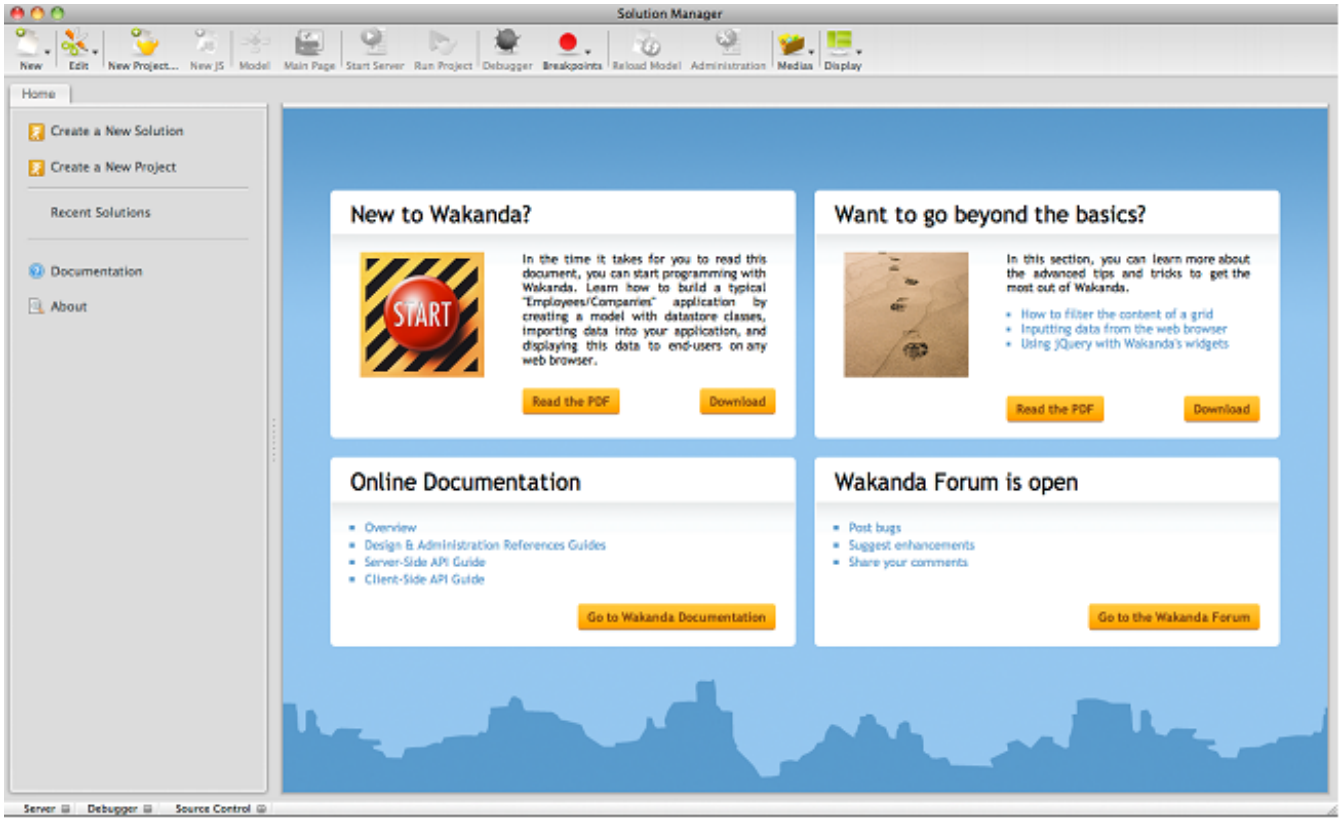
Wakanda Server itself is faceless, which means that maintenance is performed via browser – anytime, anywhere. You can directly access and manipulate models and data via JavaScript, largely eliminating the need for a dedicated datastore administrator and making associated languages like PHP strictly optional. Learn more about **Wakanda Server Administration** on the Wakanda Doc Center.

Wakanda Server's ease-of-use and high performance are tied to its implementation of JavaScript (based on WebKit's Squirrelfish just-in-time compiler) and the datastore class paradigm. For a deeper understanding of these concepts, take a look at the **Wakanda Server-side Concepts** documentation.

Wakanda Studio

Wakanda Studio contains elements that will be familiar to anyone who has used an IDE or a web development package; however, it also includes some new concepts. Wakanda Studio is comprised of:

- Solution Manager
- Datastore Model Designer
- GUI Designer
- Code Editor



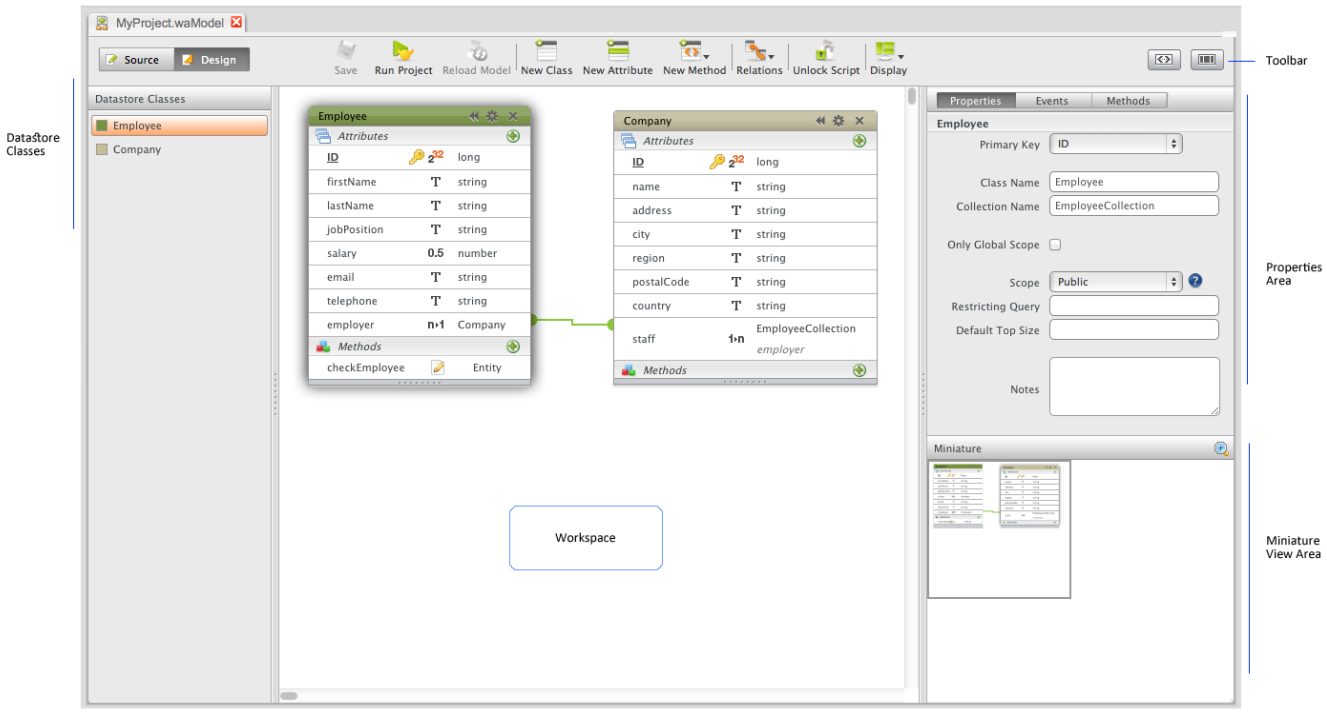
The Solution Manager is the gateway to managing all your Wakanda projects, each one being a web app. Via the solution explorer you can quickly navigate all your web apps' components (HTML, CSS, image, and associated files). You can have multiple solutions, inside which you have

multiple projects. The workspace pane adapts to the type of file you open: GUI Designer for HTML, Datastore Model Designer for your model, and Code Editor for JavaScript.

One of the more novel aspects of Wakanda Studio is the Datastore Model Designer where you visually draw your data models.

- Click, drag, and draw relations between classes.
- Modify properties, associated events, and methods via the panel.
- Zoom in and out, show or hide classes, or simply navigate your model.

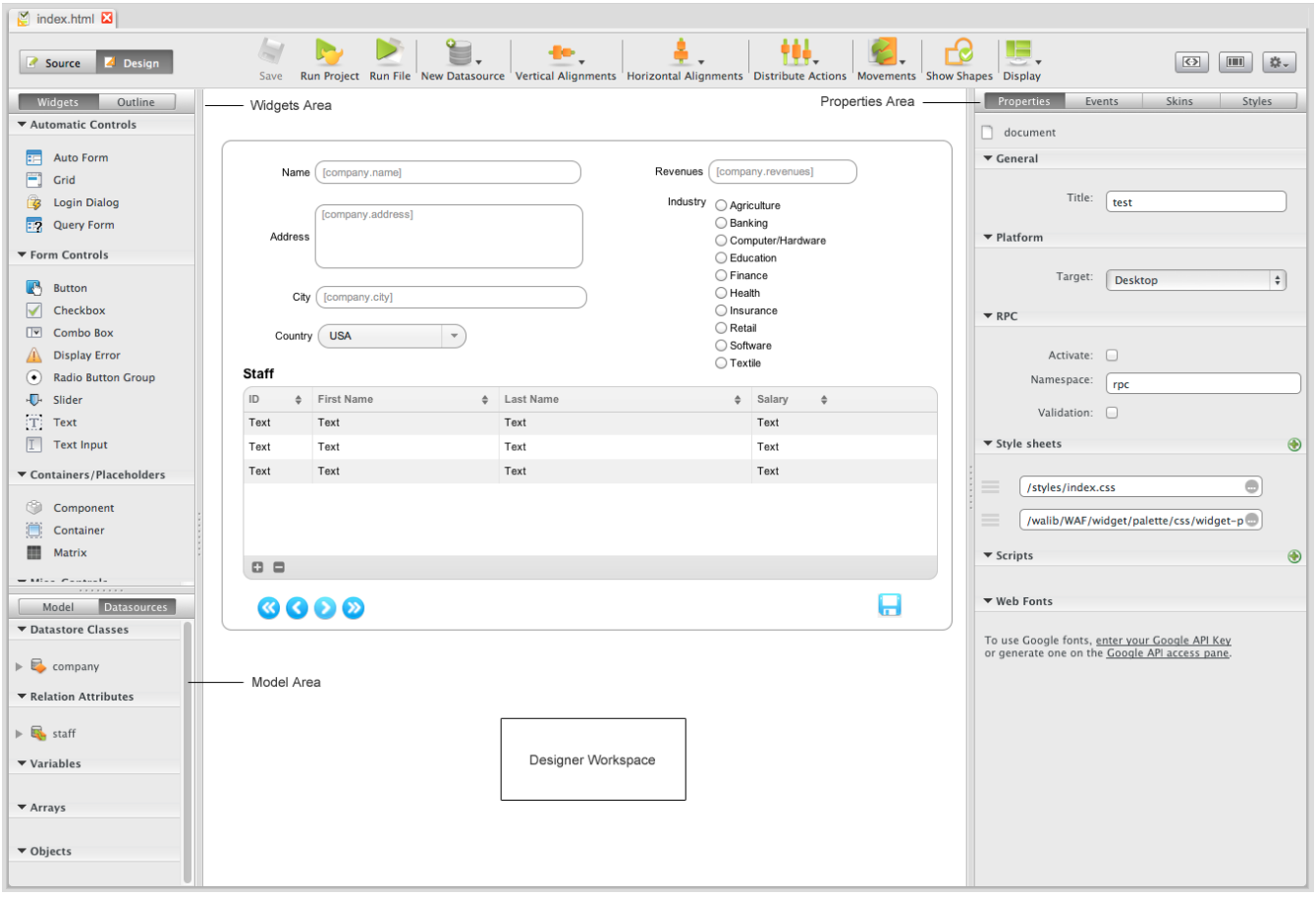
With Wakanda, even the most complex data structures are manageable.



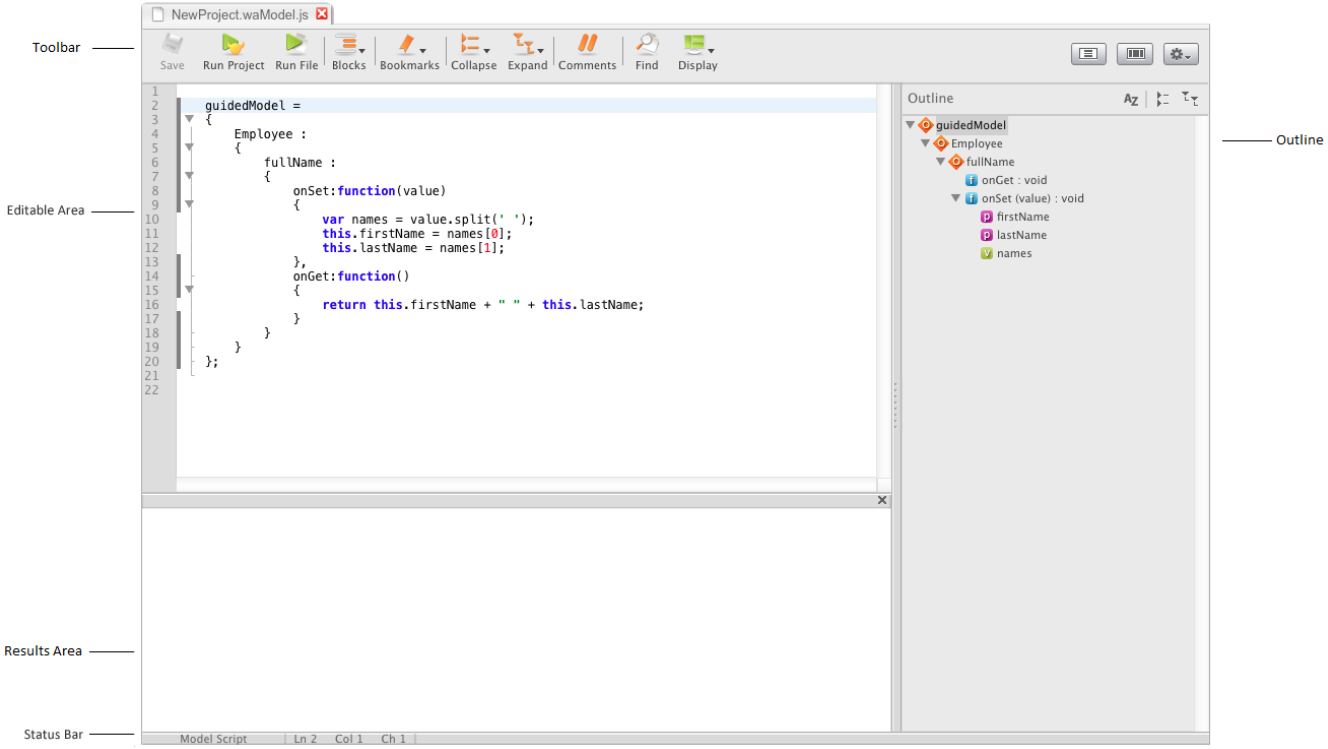
The GUI Designer looks like a typical WYSIWYG web page editor; however, it is much more powerful. Each of Wakanda's interface elements – known as widgets – can be bound to data elements and functions on the server.

Defining the color, form, shape and overall appearance of widgets automatically updates your Page's CSS file. Use the catalog panel to bind the widget to all of the functionality available to your application without writing any code. (Unless you want to add any, of course.) If it exists on the server, it's available in the panel.

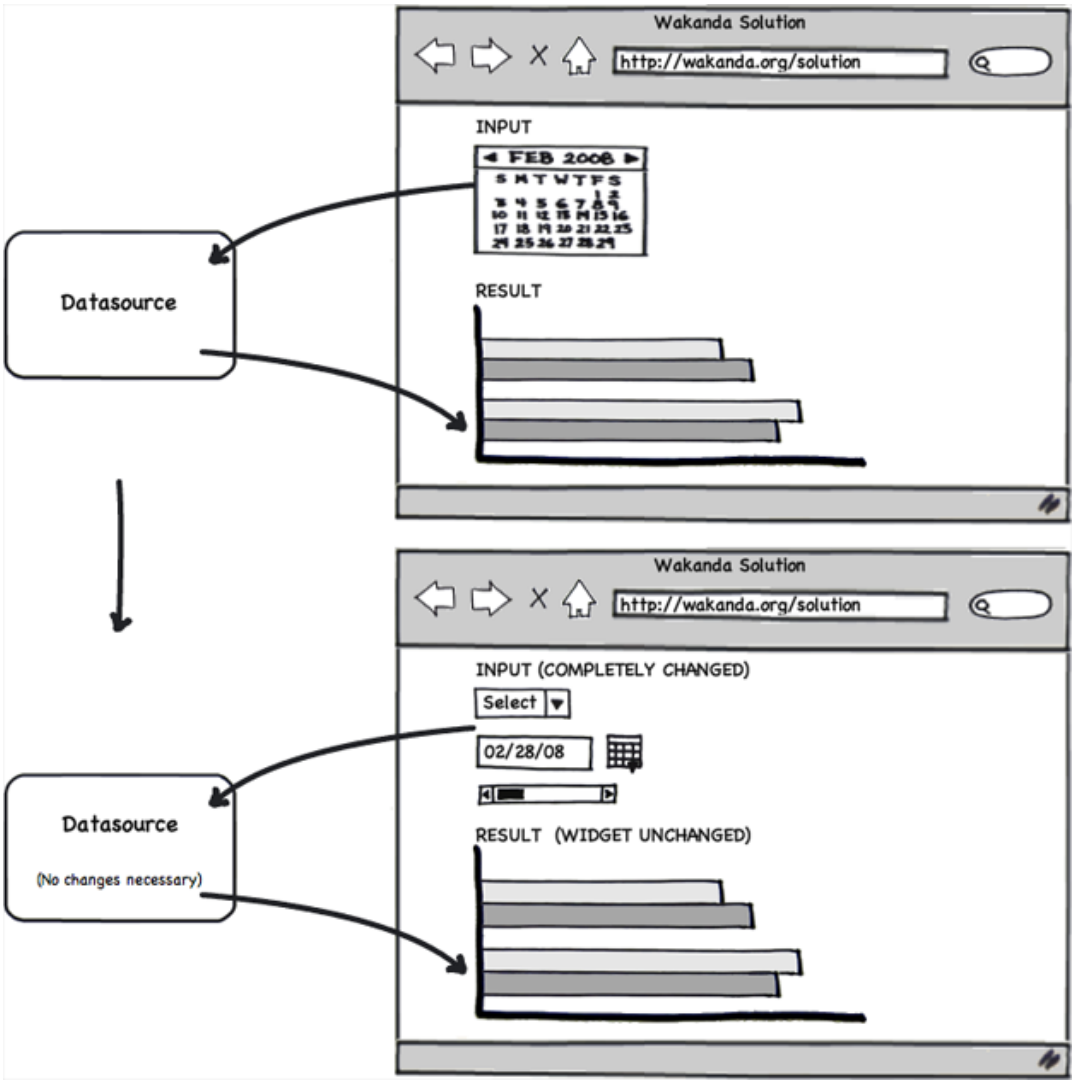
You are not, however, limited to the widgets that come with Wakanda. The GUI Designer also allows access to widgets you've created on your own (see **How to Create a Custom Widget**).



The Code Editor provides all the functionality one would expect from a typical IDE. However, Wakanda's implementation of auto-completion in the Code Editor not only provides type-ahead functionality for syntax, but for data and logic on your server, too. The Code Editor can auto-complete datastore class names, methods, JavaScript arrays, variables, and objects, saving you even more time and reducing the chance of errors.



The dynamic connection makes for the "unbreakable" nature of Wakanda development. Because everything is so closely tied to the model and the business logic – the datastore classes, the methods, etc. – each part moves with the other. If you change the behavior of a class or an attribute on the back end, it will automatically be reflected in the front end. Each connection carries all of its inherent logic with it, so when you update the back end, you won't need to modify your front end again, and vice versa.



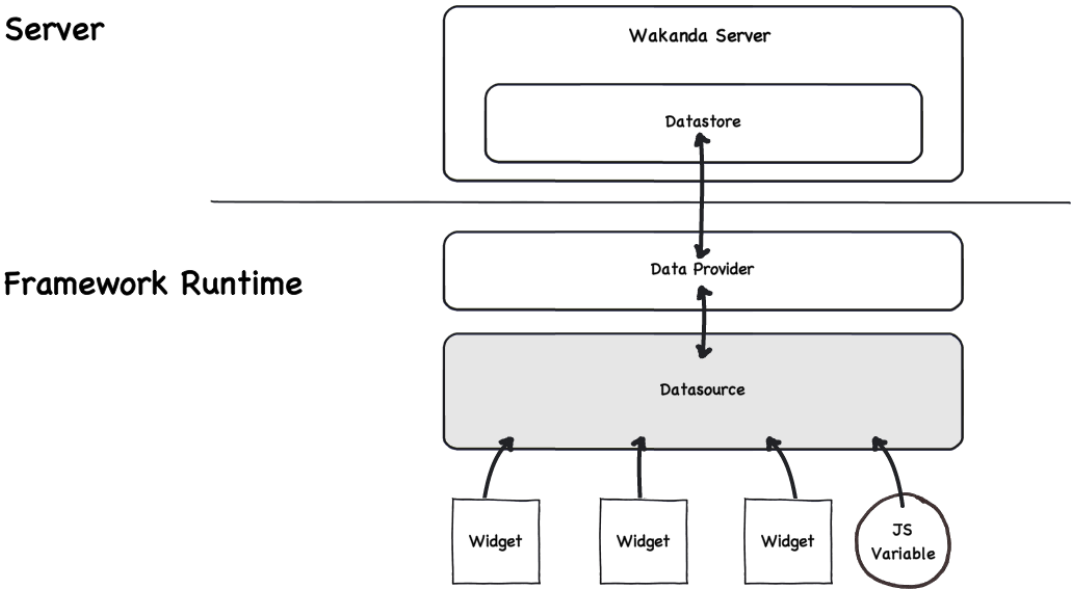
In this example, the calendar input widget is changed to a different type of date selector, but you don't need to modify the datasource or the graph, and no changes to the graph that shows the results.

For more information on how Wakanda Studio works, check out the **Wakanda Studio Reference Guide**, which covers everything from creating datastore classes to using widgets on your Pages.

Wakanda Client Framework

While Wakanda Server does much of the "heavy lifting" in order to provide fast, high-performance applications, it requires an advanced framework on the client-side to keep things running smoothly. The Wakanda Client Framework is made up of:

- A data provider to communicate with the server,
- widgets on the browser-based front end, and
- a datasource between them.



Wakanda's data provider communicates directly with the datastore on the server, typically via JSON and REST, assisted by JavaScript. The data provider handles the local cache and the parsing of data on the client, loading only the necessary data into memory. The Wakanda Doc Center discusses how to configure and use Wakanda's JSON-RPC services (see **Using JSON-RPC Services**).

Data is then passed on to the datasource. Using the inherent datastore class, the datasource sends and receives data to and from the appropriate interface widget. A datasource can also handle local JavaScript variables that aren't necessarily bound to Wakanda Server – and display them in widgets alongside server-based data.

Please review the **Dataprovider** and the **Datasource** for more details.

Widgets are the actual interface elements – based on HTML5, CSS3, and JavaScript – that end-users manipulate to send, receive, or manage data. While a variety of widgets are available by default, developers can customize widgets as much as they please, or create their own. The widgets run on top of a version of jQuery that ships with Wakanda, and by adding additional libraries, more GUI elements can be created and bound to Wakanda's logic.

Because the widgets are entirely standards-based and constructed of pure DOM elements, the final HTML page can be further customized and even animated using any other existing framework. It's up to you as the developer.

For instances in which you do not want to use a widget, you also have direct access to the dataprovider via JavaScript code, giving you a faceless way to interact with the data.

The key to Wakanda's high-performance front end is non-verbose data, resulting in fast, responsive clients that meet or surpass the performance of native applications. Using Wakanda solutions can be as easy and fun as developing them.

For more information this aspect of Wakanda, take a look at the **Introduction to Wakanda Client-side Development**.

Try Wakanda

Now that you have an understanding of how the pieces of Wakanda fit together, download and try it for yourself! A brief tutorial will show you how to Install Wakanda (**Installing Wakanda**) and just to make sure you start off on the right foot, use the **Quick Start** guide to get you up to speed with creating your first solution. Follow that up with more advanced concepts in the **How Do I** guide, and you'll be well on your way to developing like a pro.

And don't forget – you can interact with other developers like yourself as well as the Wakanda team on the forum.