Take the "RISC"!

reduced instruction set client

A substantially different architecture for industry-stable HTML Web UIs.

***browser-compatible** by design*
***fast** by design **robust** by design*

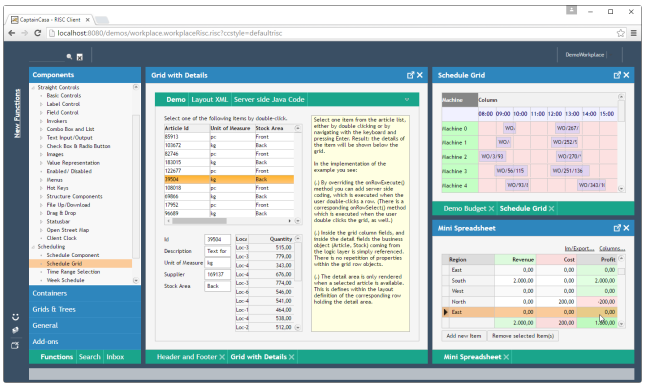
zero installation - zero maintenance

CaptainCasa Enterprise Client RISC

The „RISC.HTML method“ (reduced instruction set client): robust, compatible, fast Web UIs.

The RISC-HTML method is a new architectural paradigm for developing Web front-ends. It is designed to be used in the context of demanding, operationally used business applications with high expectations towards usability and long term robustness.

The RISC-HTML method on the one hand responds to requirements in the area of ergonomics, usage quality and performance and it grants a great degree of freedom for designing application dialogs. On the other hand this freedom comes with a new, concept-driven quality in the area of browser-/device-compatibility and in the area of supporting long lasting application life cycles.



Browser user interfaces mean “zero installation” for the end user. From software developer's perspective this “zero installation” is directly associated with “endless maintenance” on his/her side.

The RISC method overcomes this association and first time combines “zero installation” on end user side with “zero maintenance” on software developer's side.

MOTIVATION

Developing front-ends with HTML/JavaScript traditionally means a loss of control – especially when coming from native or Java-based frontend environments:

- **Browser-/Device compatibility**: the end user is dictating the browser to be used, your software is expected to properly run on “any” browser – and especially is expected to run on all future versions of browsers.

  In case of compatibility problems you are the one to solve them. The end user (especially within big companies) will never update his/her browser just to make your application work.

- **Layout flexibility**: you see a lot of really nice screens that are built with HTML - but still HTML does not provide powerful layout functions. Typical requirements like “I have several variable vertical parts – and at same time several fix-sized vertical parts” still mean some headache. The main layout philosophy of HTML still is based on text flow – and not on screen size.

- **Complexity:** HTML provides a rich set of elements, each element providing a high number of parameters and each element providing a high number of style attributes. Result: the complexity of implementations is high. - Example: in many scenarios it is nearly impossible to efficiently update the CSS style sheet definition, without causing side effects that no one is really aware of.

From a software developer's perspective this loss of control is applied from outside – and as consequence is a permanent risk. You may have invested quite some effort into your web frontend implementation – and it can happen, that you have to re-invest again into the frontend in short term future due to browser or framework updates.

Traditional frameworks of course try to hide the browser complexity and compatibility issues. But, due to the fact that they are built on top of HTML, they are not able to substantially overcome the problems and have a reached a high level of complexity themselves. In case of bugs you as software developer are responsible for solving the bug – and no one is interested if it is a framework bug or some browser issue.

THE RISC-METHOD

The RISC method means a change of paradigms: the RISC method does NOT intend to shrink-wrap HTML behind some abstracted API and to equalize browsers behind this API. - But: the RISC method goes back to the fundamentals by asking: which are the very basic elements that a UI framework requires for building a rich set of components?

So the RISC method identifies these “primitive elements” that a UI technology has to provide, so that any “normal” control can be built on top of them.

The primitive elements identified are:

- **„Rectangles“**: the UI technology needs to draw and manage rectangular areas. Rectangles may have a defined background and/or may show some text as content.

- **„Input fields“**: the UI technology needs to provide single line/ multi line text input controls.

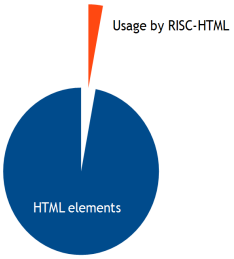
From layout perspective the UI technology must only provide a very basic way of arranging the primitive elements: it must be able to absolutely position them by passing explicit coordinates (x,y,width,height,(z)).

In short words: having rectangles and having text input fields, and having the possibility to draw them at a dedicated position - that's all you need for building all the nice components like buttons, combo boxes, grids, dialogs, layout manager on top!

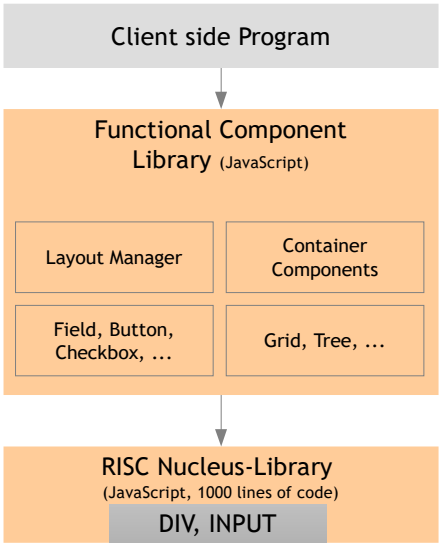
THE RISC WEB FRAMEWORK - „REDUCED INSTRUCTION SET CLIENT“

The RISC method is now transferred to HTML. The “rectangle” is represented by the DIV-element – the “input field” by the INPUT/TEXTAREA element

The browser in general allows the absolute positioning of elements. And: the browser provides a programming language JavaScript which can be used to on the one hand to encapsulate the layer of primitive elements and on the other hand to build functional controls (button,…) on top.



So, within the RISC method the browser is only used in a very thin and reduced way. The complete client side architecture looks as follows:



On the bottom layer, the two primitive elements are encapsulated by some “Nucleus Library”. This library provides an API that allows some access to the primitive elements.

On top of this nucleus library there are the control implementations (“functional component library”) - including simple controls (button, field, combo box, …), complex grid and tree controls, layout container controls and dialog controls.

The browser's role now is reduced to executing the rendering - by drawing DIV and INPUT elements. The position of the elements is not defined by the browser - but is defined on functional control layer.

A surprising “side effect” of applying the RISC method to the browser is, that the result is very fast. Browsers seem to “love” rendering of rectangular areas by shifting most of the rendering work to the graphics hardware. - And JavaScript, due to JIT compilation, is a quite fast runtime in the meantime.

SUBSTANTIAL SOLUTION AND GAIN OF CONTROL

The RISC method is a substantial, architecture-driven solution for the problems identified within the introduction of this document:

- **Browser-/Device compatibility**: the browser is only used in a very limited way: absolute positioning of DIV and INPUT components is all the browser has to do. These functions are such basic, that any browser supports them. In addition there is a “Nucleus Library” of very limited size that encapsulates these primitive elements and functions. In case of problems with a browser, the problems have to be solved on level of the “Nucleus Library” - all the functional components on top are not affected.
  In short words: the effort you have to spend for browser compatibility issues is drastically reduced!
- **Layout flexibility**: Layout management is part of the functional components - they decide where to exactly draw and arrange what. Layout management is NOT done by corresponding HTML elements (TABLE,...) with known limitations.
- **Complexity**: the complexity is not somehow distributed onto various levels (HTML, framework, CSS, …) but is clearly managed on functional component level.

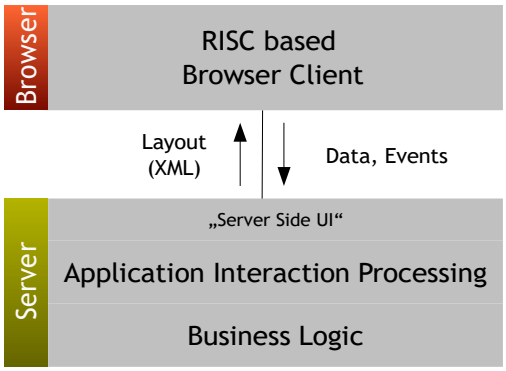
The RISC method ensures a regain of control - both from short term and from long term point of view. It's not the outside browsers that dictates the rhythm of your frontend development anymore - control is back in your hands!

CAPTAINCASA ENTERPRISE CLIENT RISC

CaptainCasa Enterprise Client is a rich client framework for developing and running user interfaces for server-based business and enterprise applications.

SERVER CENTRIC UI

CaptainCasa Enterprise Client follows the so called principle of server-centric UI-processing.

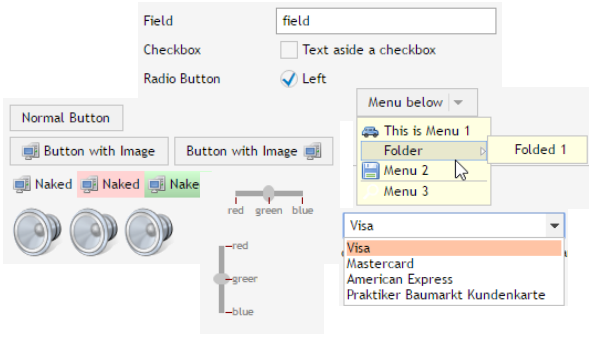


The development and interaction processing of dialogs is located on server side. Result: the server side interaction has direct access to the server side business logic. - The front-end client is a generic rendering engine, receiving form descriptions (XML) from the server side, rendering them and passing events and data updates back to the server side.

Both the communication of layouts from the server to the client and the communication of user input and events from the client to the server is optimized for performance and data volume. Only changes are transferred into both directions - so that round-trips between client server are no heavy-weight but light-weight round-trips.

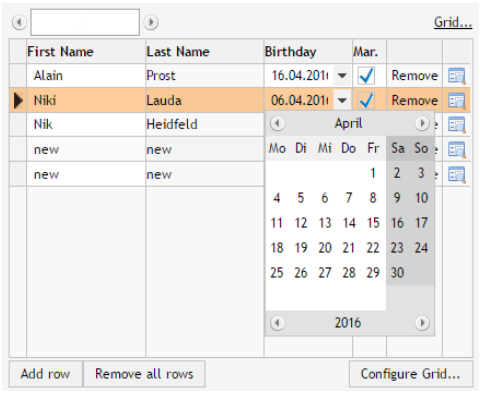
RISC BASED BROWSER CLIENT

The client part of CaptainCasa Enterprise Client is a pure HTML client which is built using the RISC method. The client provides a large number of functional components:



The HTML-RISC based library contains: classical components

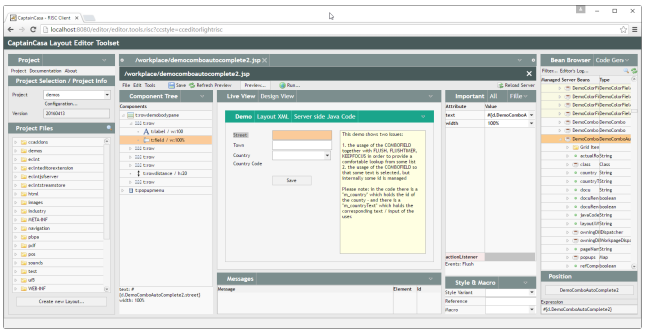
(button, field, combo box, …), flexible grids (up to Pivot processing), flexible trees, various layout containers, adaptive layout managers, file upload/download components – and many more.



DEVELOPMENT PROCESS WITH CAPTAINCASA ENTEPRISE CLIENT

The development of dialogs is done on server side: the layout is either defined as XML file within a comfortable wysiwyg-editor – or is dynamically created based on application concepts.

Each dialog is represented by a corresponding Java bean implementation that represents the server side view model.



The tool environment of CaptainCasa Enterprise Client was developed using CaptainCasa Enterprise Client itself – and as result is 100% running within the browser. It serves as nice example for the speed and interaction quality of the RISC-HTML architecture.

CAPTAINCASA COMMUNITY

CaptainCasa is an open community of independent software vendors from Germany, Switzerland, Austria, the Netherlands and Belgium.

The community was founded in 2007 – those days using a Java Swing based client, later on switching to JavaFX, and now switching to RISC-HTML. Due to the server side architecture of CaptainCasa Enterprise Client the protocol between the generic rendering client and the server side processing was kept stable. – The transfer from e.g. Java Swing client to RISC-HTML client is very simple as consequence.

The community is internally communicating through an online forum and meets one time per year for a Community Meeting in Heidelberg, Germany.

CaptainCasa GmbH is the legal entity behind the community and drives software development, services and sales of CaptainCasa Enterprise Client.

AVAILABILITY

CaptainCasa Enterprise Client RISC was released for public usage on 07th of July 2016 – after going through a community beta phase, in which community members transferred their application from Java Swing/ JavaFX based client processing to RISC-HTML based client processing.

Finally... - one last issue:

WHAT'S THE BACKGROUND FOR THE NAME „RISC-HTML“?

The name “RISC-HTML method” is a reminiscence to the “CISC” vs. “RISC” processor discussion in the 80s/90s. There are a lot of similarities between this discussion about hardware and the discussion about problems in the browser area.

When looking back on processor architectures, then in the beginning there was the attitude to add more and more instructions to the processor's instruction set. The thinking was based on the paradigm: every command running inside the silicium of the processor is a “good command”. The result: so called “CISC” processors (complex instruction set).

Of course each new command increased the complexity of the processor design. And: complex commands required more processing time than simple logical operations – and slowed down the processor. Result: a shift of paradigm happened:

So called “RISC” processors (reduced instruction set) only provided very basic commands anymore – resulting in a much cleaner and simpler design of the processor. Complex operations were not executed by the processor itself but were part of the program that runs through the processor. In other words: algorithmic complexity was taken out of the hardware and was shifted into the software.

The similarities between the processor discussion and the browser discussion are obvious: the browser started as text rendering engine and gained more and more elements, attributes, CSS parameters – now having reached a huge level of complexity. The RISC-HTML method takes the complexity out of the inner browser processing and shifts it into a dynamic, JavaScript based processing in front of the core browser.

CaptainCasa GmbH
Hindemithweg 13
D- 69245 Bammental
http://www.CaptainCasa.com
info@CaptainCasa.com