



Browser Frontend + Erlang Backend → Webapp Anywhere



Yusheng Li

Qiyun Software

<http://qyapp.com>






Browser is Evolving... into an OS Abstraction Layer

-  Normal HTML + CSS is good enough to bring up a complex GUI App
-  HTML5 adds Canvas, WebGL, Local Storage, manifest, web socket, Audio, Video, and more...

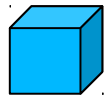


Browser is Evolving... into an OS Abstraction Layer

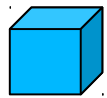
-  Javascript Engine is fast enough to run complex logic locally
-  Can manipulate device hardware through Javascript API
-  Highly standardized, smoothly across browsers, OS platforms, Devices...



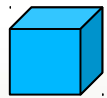
HTML5+CSS3+Javascript → Rich Client App



All necessary program logics are implemented in client javascript



All GUI changes are implemented in client javascript



All device hardware manipulations are implemented in client javascript



HTML5+CSS3+Javascript → Rich Client App

We now have a complete client app that can act as native app.

What do we lack here?

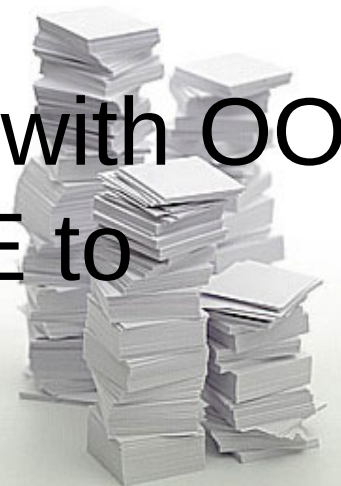
```
//export your function so that it can be called by ns.remote.fun(p1,p2);  
function export_getUsers(name,address){  
  if(RpcUser.sessionid() == undefined){  
    return "please login first";  
  }  
  var users = MyDB.table('users').where({name:name,address:address});  
  return users;  
}
```



Server Side Logic and Data!

A Brand New Way to Create Webapp

- Webapps are organized in projects.
- Each project is a single unit for debug, deploy and reuse.
- Inside each project, use `DEFINE_CLASS` to organize logic with OO design; use `DEFINE_NAMESPACE` to form a module.



A Brand New Way to Create Webapp

- Use `var mod = using("username/projname")` to reference a module defined in the project.
- Server Side Logic can be implemented in Javascript, Erlang, etc
- `mod.remote` and `mod.remote_async` has all RPC functions that can be used to call server side functions.








Demo

<https://qyapp.com/liyusheng/feedback>

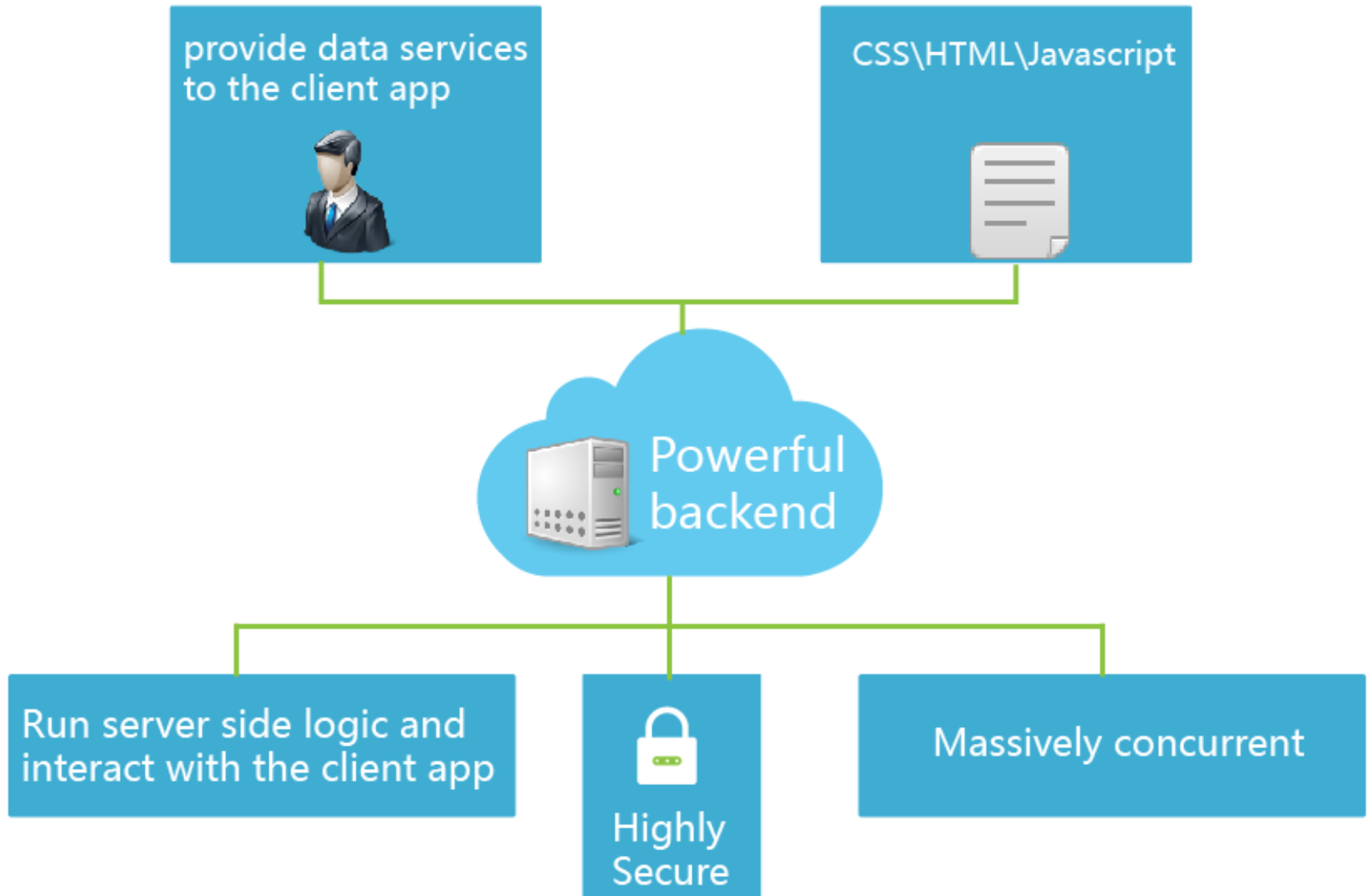


We need a powerful backend

-  as a web server to serve HTML, Javascript, CSS files.
-  provide data services to the client app
-  Run server side logic and interact with the client app
-  Highly secure, isolated for each client app
-  Massively concurrent



We need a powerful backend

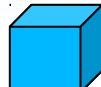
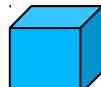
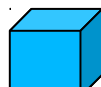
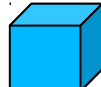
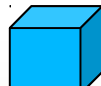


We need a powerful backend

Can we do all these?



Erlang to the rescue

-  Mature web server implementation (Yaws)
-  Inherently concurrent
-  Strong isolations between processes
-  Ideal at network protocol implementation
-  OTP behavior architecture



Erlang to the rescue

 Built-in lexical and syntactical parser

 Highly mature language and runtime, battle tested for centuries (in computer age)



Erlang Backend



Serve HTML, CSS, Javascript files



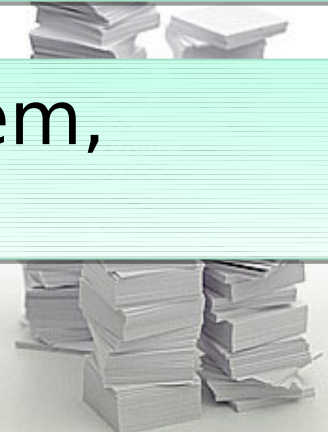
User Identification, Security Infrastructure



Server Side Javascript compilation and execution



Glue layer to databases, file system, version control system, etc...



Erlang Backend

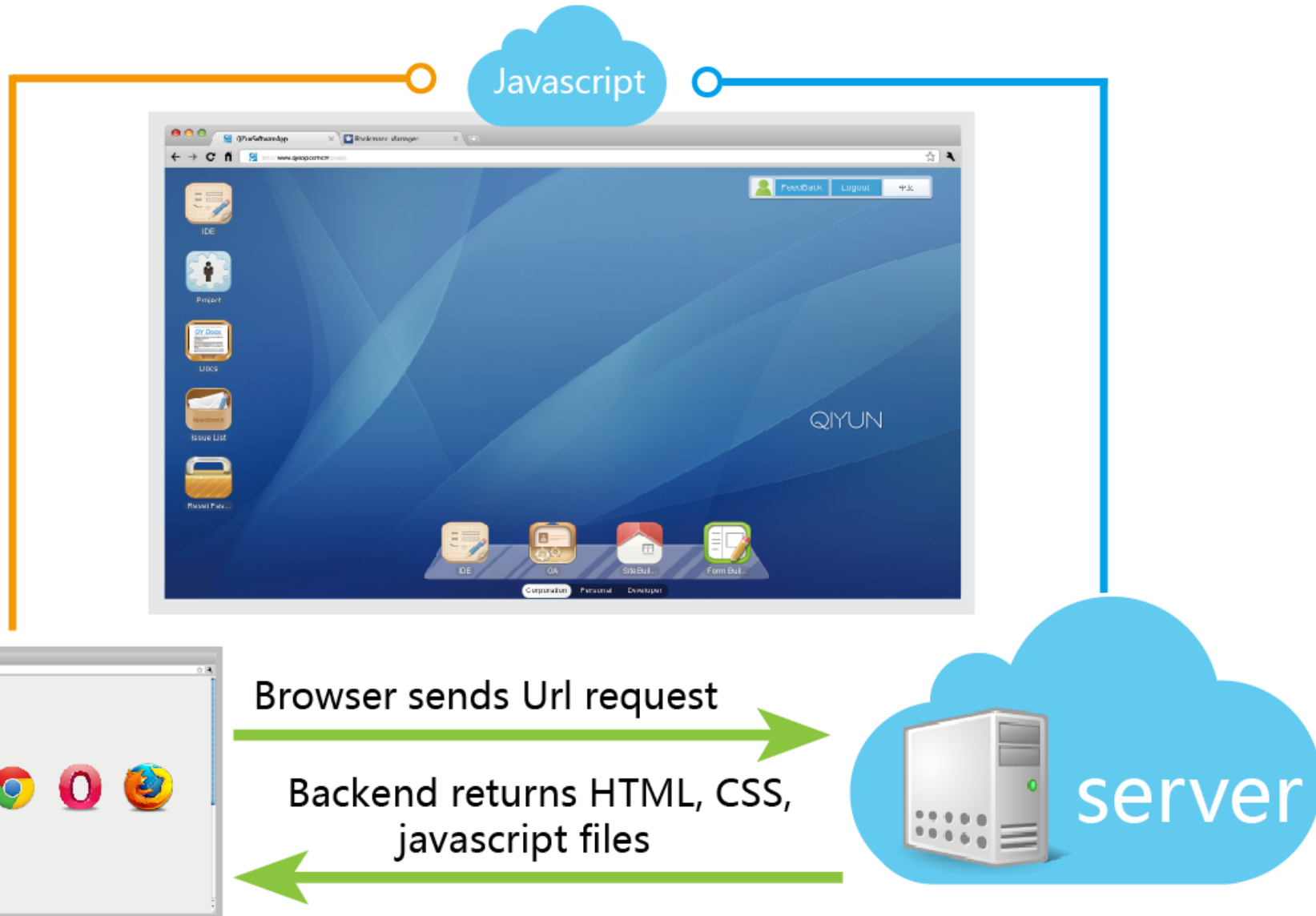


Provide Remote Procedure Calls for Client Apps

- To call server side functions
- To exchange web socket messages
- To manipulate database
- To interact with backend file system



Browser Frontend + Erlang Backend runtime architecture




```
1  if(!ns.remote || !ns.remote.getAllUsers){  
2      showMessage("No Method");  
3      return;  
4  }  
5  var allUsers;  
6  var ret = ns.remote.getAllUsers();  
7  if(ret.success && !ret.result.error){  
8      allUsers = ret.result;  
9      showMessage("OK");  
10 }else{  
11     showMessage(ret.result.error);  
12 }
```

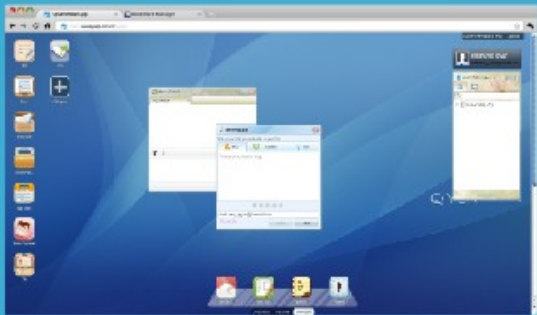
Javascript calls server side functions



```
1  function export_getALLUsers() {  
2      return MyDB.table("users").select();  
3  }
```

Backend runs the code and returns the result

Javascript updates the GUI with the result

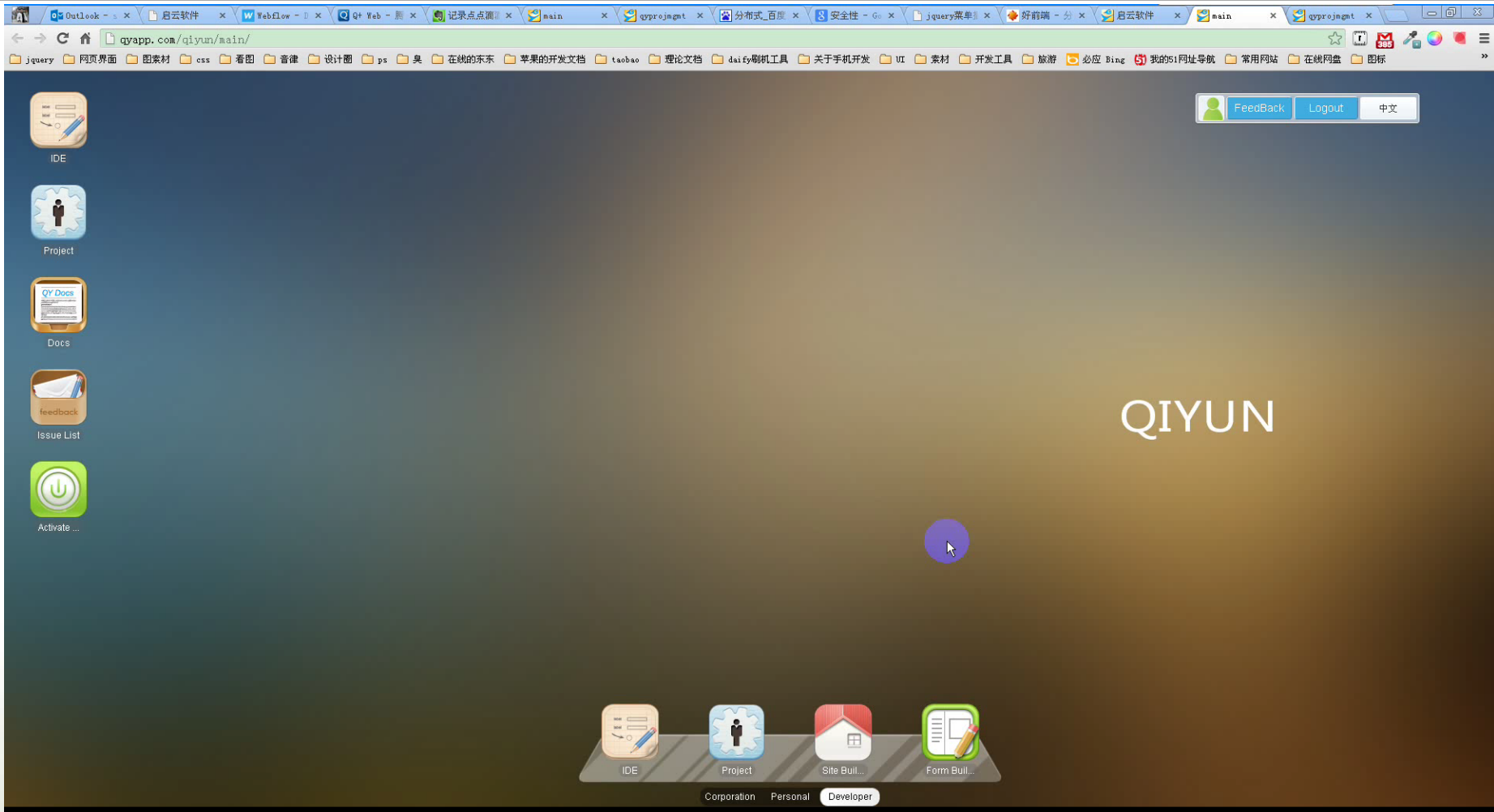


Javascript

Javascript sends web socket messages



Browser Frontend + Erlang Backend development time architecture



We are going to **open source!**

Frontend and Backend Runtime!

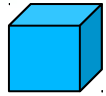
Can completely setup an environment to serve the webapps developed by our platform

Comments and contributions are welcome!

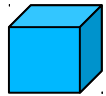
Please send email to yshli_qiyun@hotmail.com if you like to receive announcements.



Backend Workflow



When a GET request arrives, Check User ID and serve files

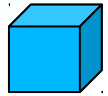


When a POST request arrives, Check User ID and get the RPC parameters

- BERT/JSON format are supported

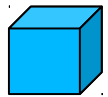


Backend Workflow



According to the RPC parameters, call built-in modules or user implemented modules

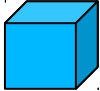
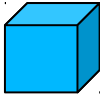
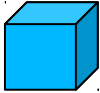
- Interact with file system, database, version control system etc through APIs
- Send back web socket messages if necessary



Return the result as BERT/JSON

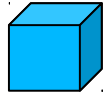


Server Side Javascript

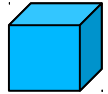
-  Lexer and parser are implemented in Erlang's xrl and yrl tools
-  Compiled into erlang modules
-  Built-in MyDB, MyFile, RpcUser, RpcGlobal Javascript objects with provided functions



Server Side Javascript



Runtime check ensure security, memory usage and storage usage.



Really light weight, utilizing Erlang's process mechanism



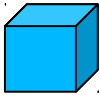
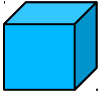
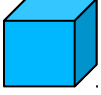
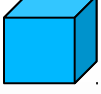
Erlang Language Support

```
1 -module(srcmodule).
2
3 -include_lib("kernel/include/file.hrl").
4 -define (PRINT(A), {ok,A}).
5 -define (MACO_WITH_PARAM(A), ?PRINT(A)).
6 -define (MACO_WITH_BIF(A), {list_to_binary(A),length(A)}).
7
8 -export([test_macro/1, testcase/2]).
9
10 test_macro(X) when is_list(X) -> ?MACO_WITH_BIF(X);
11 test_macro(X) when X == 0 -> #file_info{};
12 test_macro(X) when X > 0 -> ?MACO_WITH_PARAM(X);
13 test_macro(X) when X < 0 -> ?MACO_WITH_PARAM(-X);
14 -ifdef(debug).
15 test_macro(X) -> ?PRINT(X).
16 -else.
17 test_macro(_X) -> ok.
18 -endif.
19
20 testcase(X,Y) ->
21     SumFun = fun (A,B) -> abs(A) + abs(B) end,
22     case SumFun(X,Y) of
23         Value when Value > 100 -> test_macro(Value);
24         _ -> lists:map(fun(A)-> A+X+Y end,lists:seq(1,10))
25     end.
```

```
1 -module(destmodule).
2
3 -export([qyfunmap/0, qyfun1/1, qyfun2/2]).
4
5 -record(file_info,
6     {size, type, a1, a2, a3, a4, mode, links, a5,
7     a6, a7, a8, a9}).
8
9 -record(file_descriptor, {module, data}).
10
11 qyfun1(X) when is_list(X) ->
12     {list_to_binary(X), length(X)};
13 qyfun1(X) when X == 0 -> #file_info{};
14 qyfun1(X) when X > 0 -> {ok, X};
15 qyfun1(X) when X < 0 -> {ok, -X};
16 qyfun1(_Var1) -> ok.
17
18 qyfun2(X, Y) ->
19     Var3 = fun (Var1, Var2)
20         -> abs(Var1) + abs(Var2) end,
21     case Var3(X, Y) of
22         Var4 when Var4 > 100 -> qyfun1(Var4);
23         _ ->
24             lists:map(fun (Var1) -> Var1 + X + Y end,
25                 lists:seq(1, 10))
26     end.
27
28 qyfunmap() -> [{{"testcase">>,2},qyfun2},
29     [{"test_macro">>,1},qyfun1]].
```




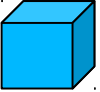


Erlang Language Support

-  Lexer: `qyerl_scan.erl` (`erl_scan.erl`), generate token sequence
-  Erlang Pre-Processor: `aleppo`, all macro tokens are reset
-  Parser: `qyerl_parser.yrl` (`erl_parser.yrl`), generate abstract forms
-  `qyerl_compiler`: convert abstract forms to source file

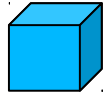


Erlang Language Support

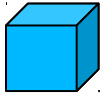
-  No new atoms are generated. Var1, Var2..., and Fun1, Fun2..., are used.
-  All Erlang function calls are verified to ensure security checks.
-  Memory Checks will be added in function calls
-  Inter-op with functions written in Server-Side Javascript



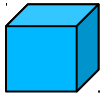
More languages are coming



May integrate Robert Virding's luerl to provide lua support (mainly for webapp game development)



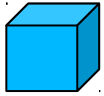
As long as we can write the parser for a new language and transform its semantics into erlang module



Selected languages should be suitable for server side programming



What We like Erlang Most?

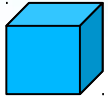


Pattern Matching

--really mind blowing, changed our way of programming



What We like Erlang Most?



Mature, Battle Tested

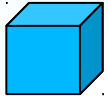
--almost all corner cases taken care of, e.g.,
`run_erl, to_erl` for running as service then
attached to the Erlang console

--debug and hot code loading in live
production system

--all features combined to form a consistent,
powerful system, just genius piece of work



What We like Erlang Most?



OTP Behaviour Design

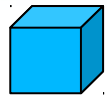
--qyserver_db,qyserver_filecache,qyserver_rpc,....
all implemented in similar ways.

--all work smoothly with no glitch.

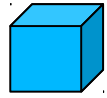
--ETS to hold states



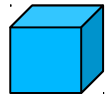
What We Want More?



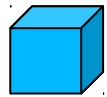
More Powerful Built-in Logging



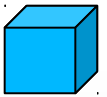
Simpler Tracing



Simpler Performance Measurement



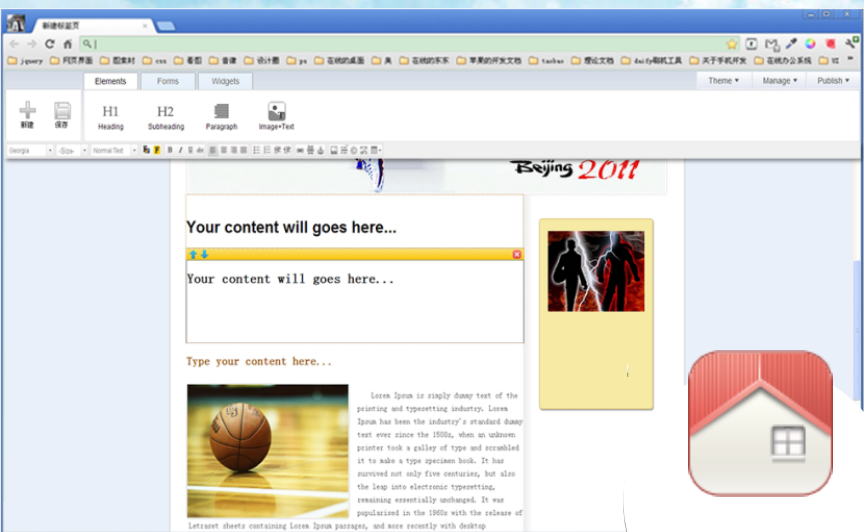
Built-in String Type



Local Variable



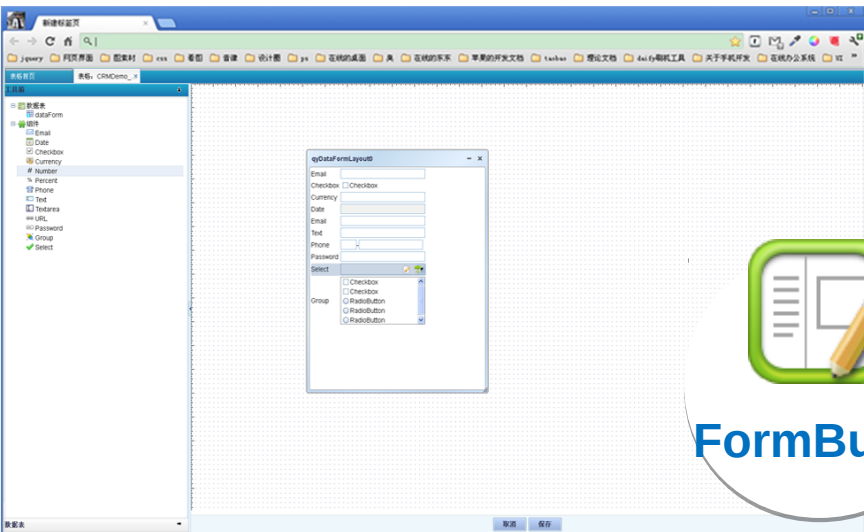
What We Build So Far?



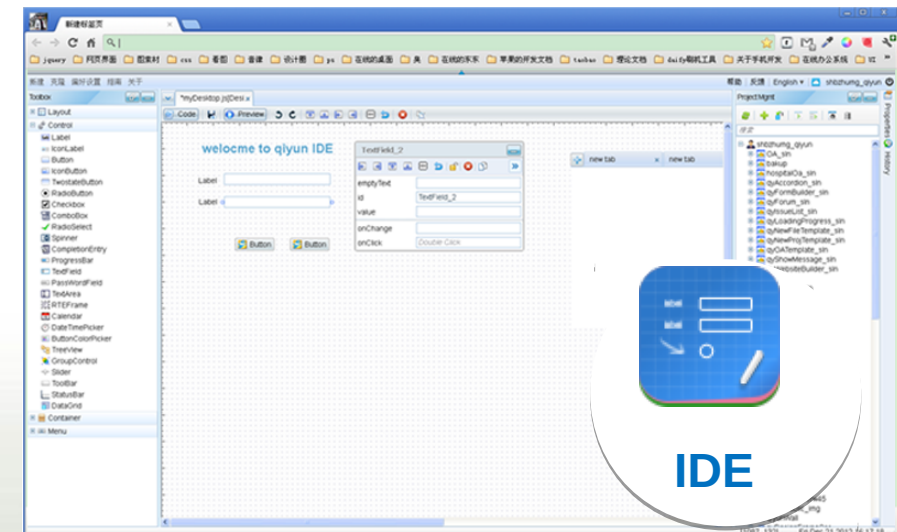
SiteBuilder



ProjectMgmt



FormBuilder



IDE

What Apps We Build So Far?

CooChart

Team-R-Us

Dimensions
Width: 683 Height: 302

Margin
Top: 40 Bottom: 40
Left: 40 Right: 40

Background
Type: Solid
Color: []

Border
Width: 0
Color: []

Sales Report

Month	Color
Jan	Blue
Feb	Orange
Mar	Green
Apr	Pink
May	Red

Board
Welcome to Team.R.Us!
We share the board and exchange information.
Invite your team to the board.
Use a board to hold a big project.
Keep your own board for private.
Go to other boards.

List
Here are four lists already.
Create a list. It's easy!
Different lists represent for different demands or status.
Try to move the list! Interesting?
Achieve a list.....
.....and then it will disappear from the board.....
you can restore it from the achieved lists.

Card
This is a card.
A card represents for a task.
Do you want to comment?
Click a card to see details.
Assign tasks?
Add a deadline to a task.
Segmentation a task.
Check the subtasks after it's finished.

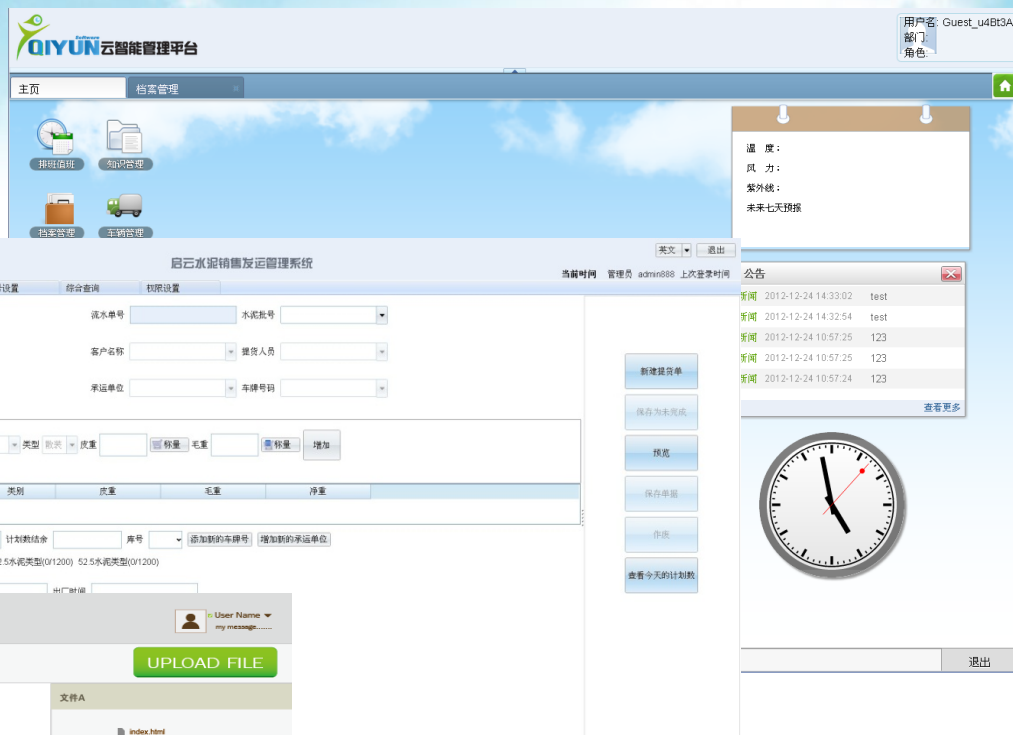
Tips
You can discuss freely!
Click your name above to see what happens.
Show personalized by using an avatar.
Need some help?
Add a card...

Members
Add Members...
Activity View more...
liiung_qiyun move cardand then it will disappear from the board.....to list:Listposition5
liiung_qiyun move card Use a board to hold a big project to list:Boardposition4
liiung_qiyun move card Use a board to hold a big project to list:Listposition5
liiung_qiyun move card Keep your own board for private to list:Boardposition5



What Apps We Build So Far?

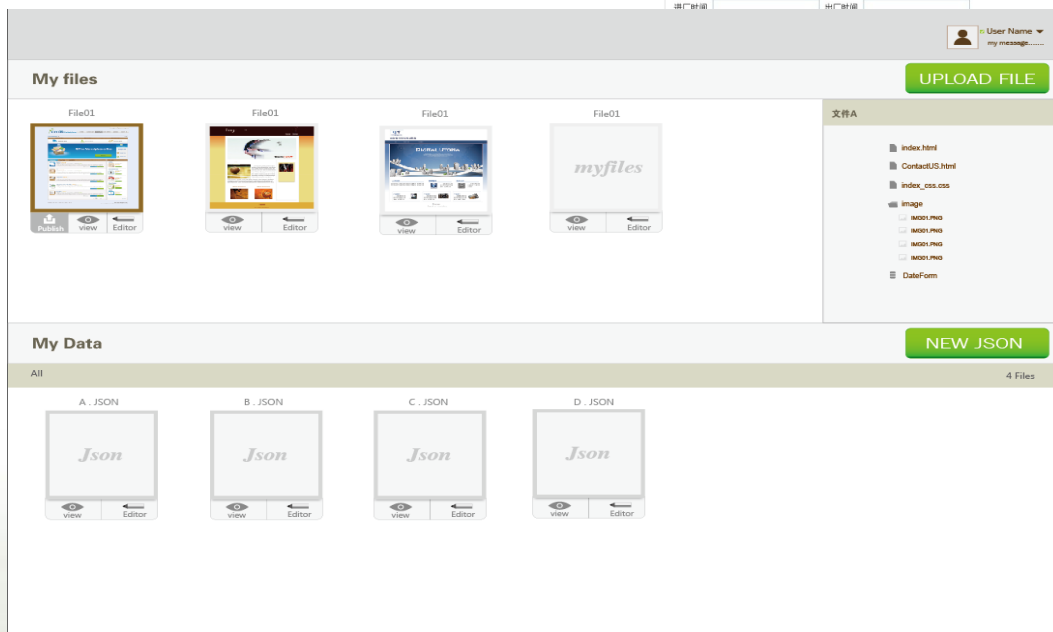
OA System



ERP

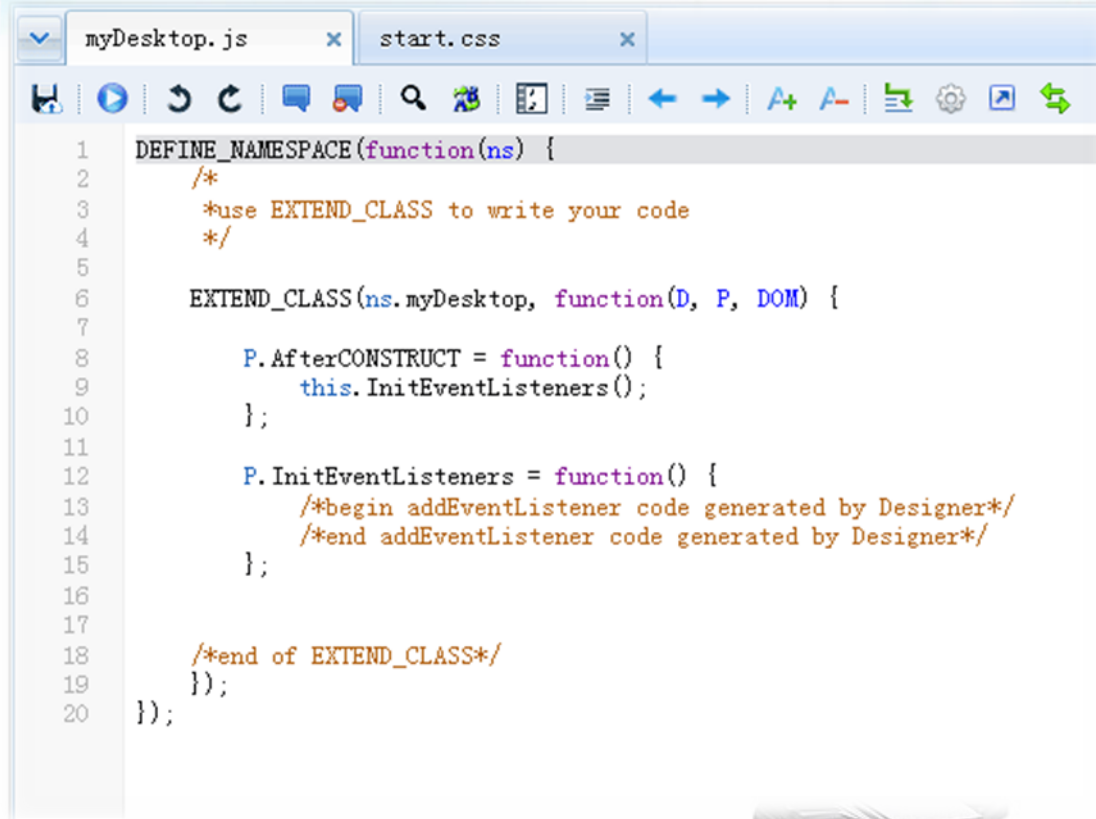


Bean2Cloud



Qiyun Editor

QiYun Editor is an online code editor, which can support more than 20 programming languages, such as erlang, javascript, html, CSS, SQL and diff, etc.



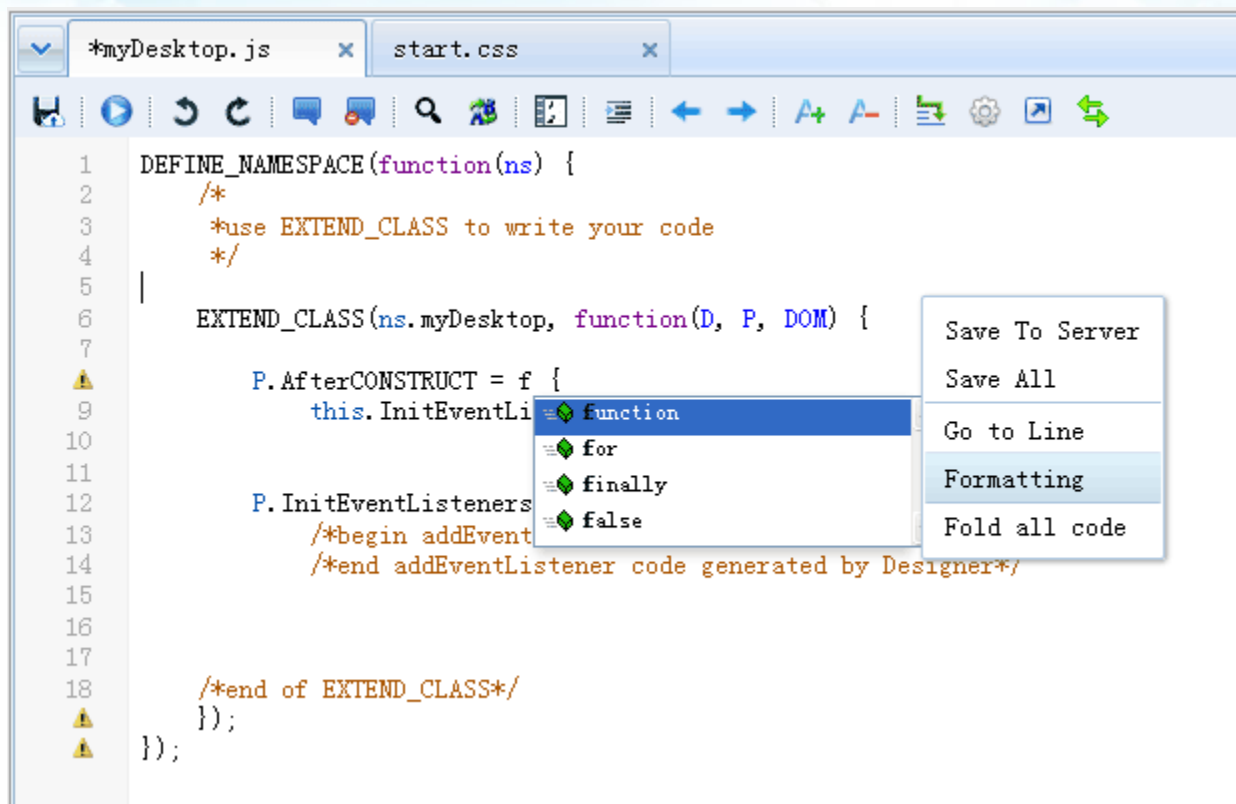
```
1  DEFINE_NAMESPACE(function(ns) {
2      /*
3       *use EXTEND_CLASS to write your code
4       */
5
6      EXTEND_CLASS(ns.myDesktop, function(D, P, DOM) {
7
8          P.AfterCONSTRUCT = function() {
9              this.InitEventListeners();
10         };
11
12         P.InitEventListeners = function() {
13             /*begin addEventListener code generated by Designer*/
14             /*end addEventListener code generated by Designer*/
15         };
16
17         /*end of EXTEND_CLASS*/
18     });
19
20 }
```



Qiyun Editor

Core Features

1. syntax check
(dynamic parsing)
2. auto complete
3. code format
4. theme customize



```
1 DEFINE_NAMESPACE(function(ns) {
2     /*
3     *use EXTEND_CLASS to write your code
4     */
5     |
6     EXTEND_CLASS(ns.myDesktop, function(D, P, DOM) {
7
8     P.AfterCONSTRUCT = f {
9         this.InitEventLi function
10
11
12     P.InitEventListeners
13         /*begin addEvent
14         /*end addEventListener code generated by Designer*/
15
16
17
18     /*end of EXTEND_CLASS*/
19     });
20 }
```



Team.R.Us

Team.R.Us is a Web-App for team collaboration and task management, which runs on Android, iPhone, iPad and Desktop.



Team.R.Us

Core Features

MVC architecture based on pure javascript, Complete synchronization by web socket (very fast), Instant messenger via web socket (high cocurrency)

The screenshot displays the Team.R.Us web application interface. At the top, there is a navigation bar with "Notifications" and "Boards" tabs, and a user profile for "LLiumg_qiyun". Below the navigation bar, there are several buttons: "Welcome board", "Options...", and "Add List". The main content area is divided into four columns:

- Board:** Contains a welcome message, a share information section, an invite team section, a project board section, a private board section, and a go to other boards section.
- List:** Contains a message about existing lists, a "Create a list" section, a "Different lists" section, a "Try to move the list" section, an "Achieve a list" section, and a section about restoring lists from archived lists.
- Card:** Contains a "This is a card" section, a "Do you want to comment?" section, a "Click a card to see details" section, an "Assign tasks?" section, an "Add a deadline to a task" section, a "Segmentation a task" section, and a "Check the subtasks" section.
- Tips:** Contains a "You can discuss freely!" section, a "Click your name" section, a "Show personalized" section, and a "Need some help?" section.

On the right side, there is a "Members" section with user avatars and an "Add Members..." button. Below that is an "Activity View more..." section showing a list of recent actions, such as "move card" and "Use a board to hold a big project". At the bottom right, there is a chat window with a message from "rwanerd_qiyun" and a "View More" button.

Bean2Cloud

BaaS. Design and build the online database by drag&drop. Auto create server side logic for access control and APIs to Insert, Delete, Modify and Query the data. Auto create client forms. Auto connect the frontend to the data.

The screenshot displays the Bean2Cloud web application interface. At the top right, there is a user profile section with a dropdown menu showing "User Name" and "my message.....". Below this, the "My files" section features a grid of four file thumbnails, each labeled "File01". The first thumbnail shows a complex web page with a "Publish" button, while the others show simpler layouts with "view" and "Editor" buttons. To the right of the file grid is a sidebar titled "文件A" (File A) containing a file tree with items like "index.html", "ContactUS.html", "index_css.css", "image", and "DateForm". A green "UPLOAD FILE" button is located in the top right of this section. Below the "My files" section is the "My Data" section, which includes a "NEW JSON" button and a grid of four JSON file thumbnails labeled "A . JSON", "B . JSON", "C . JSON", and "D . JSON". Each thumbnail has a "view" and "Editor" button. The bottom right corner of the "My Data" section indicates "4 Files".



<http://qyapp.com>

We have a team behind it





<http://qyapp.com>

We Like To Hear From You!
Any Questions?

yshli_qiyun@hotmail.com

