node-ffi

othree jsdc

@othree

othree.net

MozTW

F2E at HTC

PhD Candidate

Topics

r3

ffi

node-ffi

node-ffi
packages

node-r3

r3

- Super fast router lib written in C
- by c9s

https://github.com/c9s/r3

r3

- Based on DFA(確定有限狀態自動機)
- Precompile routes into prefix tree
- 1000 times faster than Journey

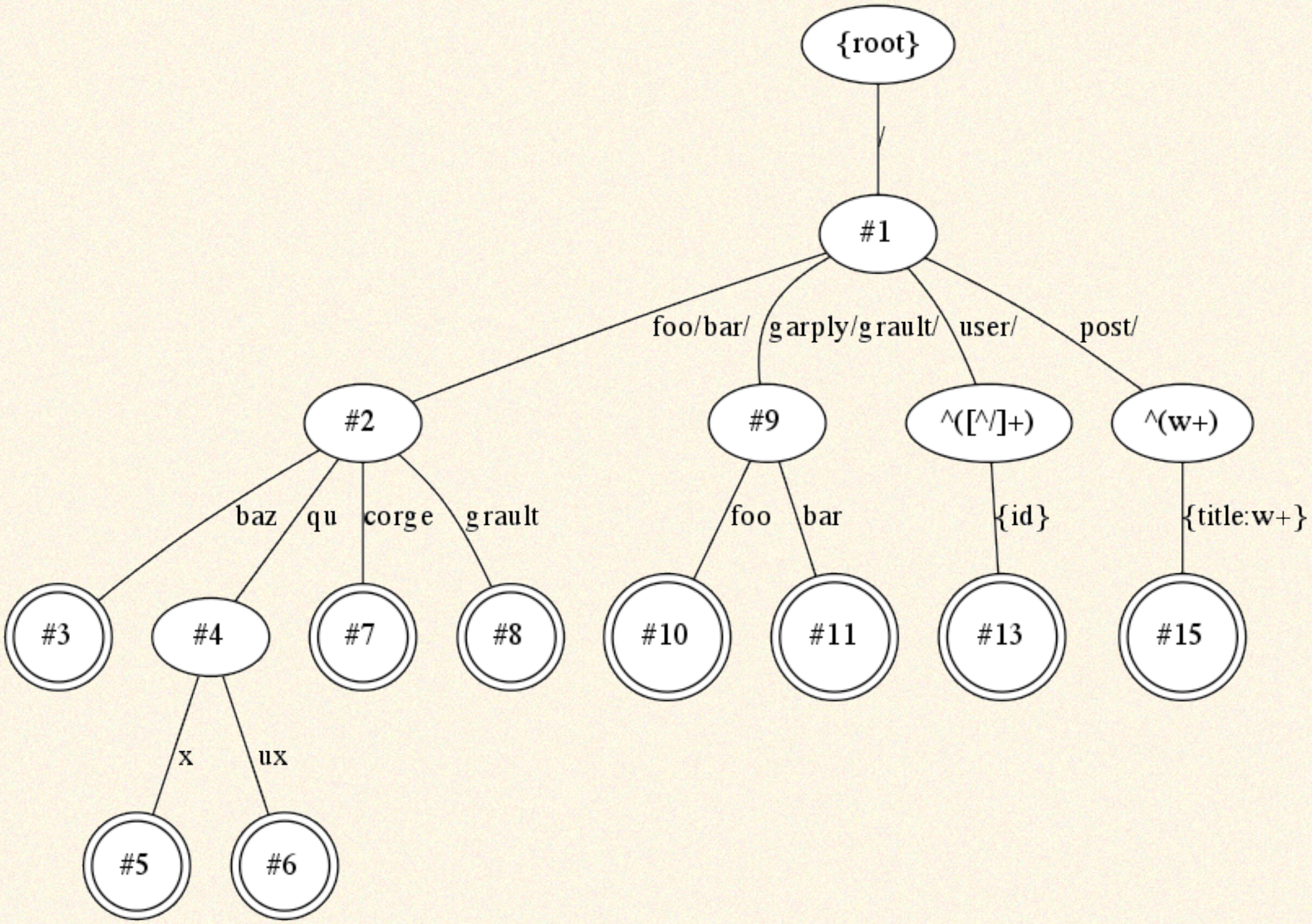
http://c9s.blogspot.tw/2014/05/r3-url-router-library.html

How to Use libr3

- Create tree
- Insert route with data
- Compile tree
- Match route and get the data

Sample 1/2

```
n = r3_tree_create(10);
int route_data = 3;

r3_tree_insert_routel(n, METHOD_GET, "/blog/post",
  sizeof("/blog/post") - 1, &route_data );

int err = r3_tree_compile(n, &errstr);
```


{root}
/
#1
foo/bar/
garply/grault/
user/
post/
#2
#9
^([^/]+)
^(w+)
baz
qu
corge
grault
foo
bar
{id}
{title:w+}
#3
#4
#7
#8
#10
#11
#13
#15
x
ux
#5
#6

Sample 2/2

```
match_entry * entry = match_entry_create("/blog/post");
entry->request_method = METHOD_GET;

route *matched_route = r3_tree_match_route(n, entry);
matched_route->data; // The Data

match_entry_free(entry);
r3_tree_free(n);
```

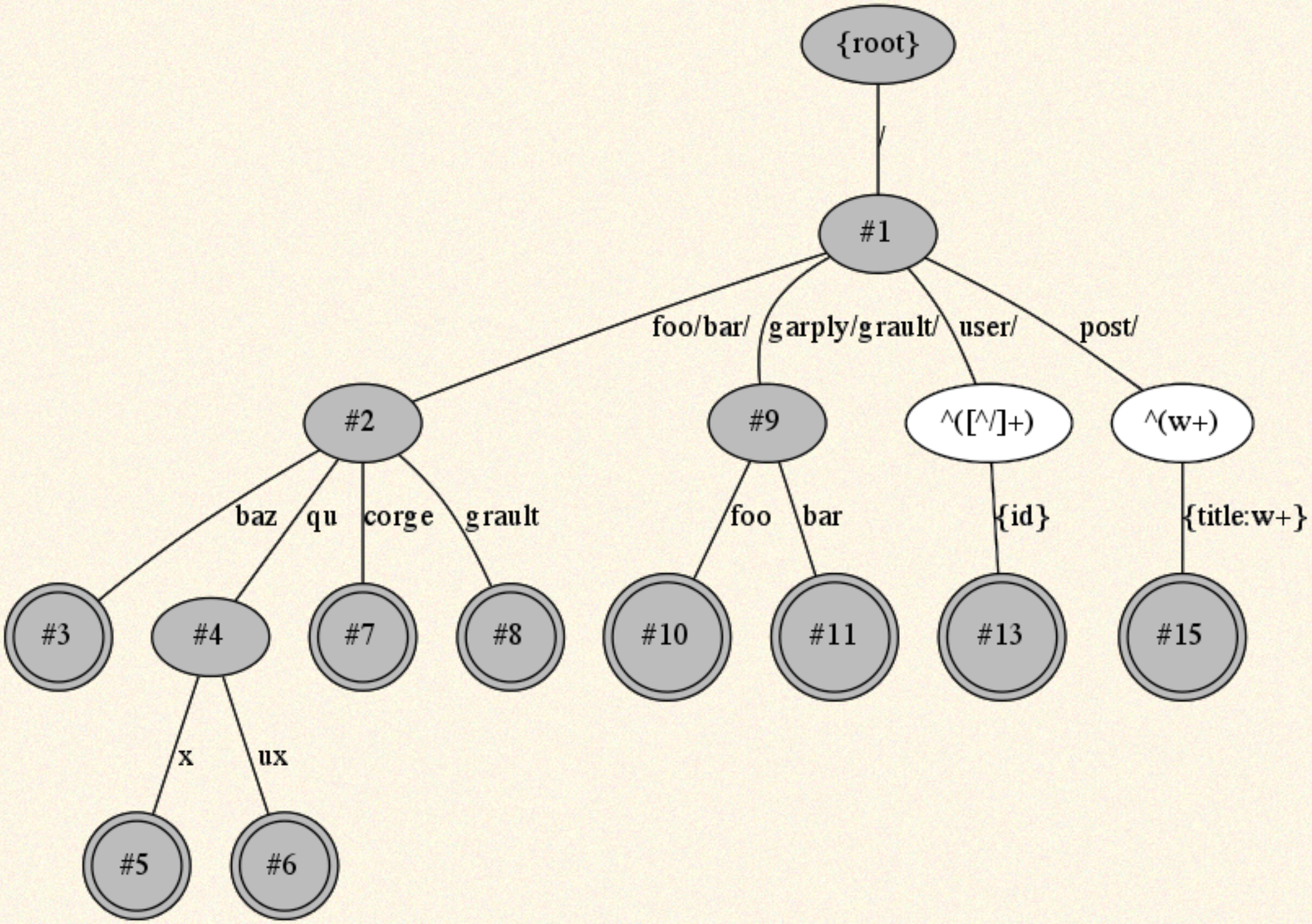
match_entry

- Used for capture variables
- Add method condition
- Future:
  - IP mask
  - Port range

Captures

- Use `{variable_name}`

- Can use regular expression


{root}
/
#1
foo/bar/
garply/grault/
user/
post/
#2
#9
^([^/]+)
^(w+)
baz
qu
corge
grault
foo
bar
{id}
{title:w+}
#3
#4
#7
#8
#10
#11
#13
#15
x
ux
#5
#6

```
match_entry * entry = match_entry_create(“/blog/post/1/2/3“);

entry->request_method = METHOD_GET; // Method Condition

route *matched_route = r3_tree_match_route(n, entry);


matched_route->data; // The Data

entry->vars; //Captured Vars, a string array
```

May I Use r3 in node

Yes.

How

- Node.js extension
  - Write C/C++, caasi/node-r3
- Foreign Function Interface (FFI)
  - Use dynamic lib

Why

- It might faster than current solution
- 因為我可以（Because I can）
- I am writing my PhD thesis, I am willing to do anything except writing thesis.

Foreign Function Interface

> A foreign function interface (FFI) is a mechanism by which a program written in one programming language can call routines or make use of services written in another.

https://en.wikipedia.org/wiki/Foreign_function_interface

- By understood calling convention
- You can execute foreign functions, send argument, and read return value
- In assembly language level
- Call C dynamic lib in most cases

__cdecl

```
val = sumExample(2,3)
```

```
; // push arguments to the stack, from right to left
push        3
push        2

; // call the function
call        _sumExample

; // cleanup the stack by adding the size of the arguments to
; // ESP register
add         esp,8

; // copy the return value from EAX to a local variable (int c)
mov         dword ptr [c],eax
```

node-ffi

> node-ffi is a Node.js addon for loading and calling dynamic libraries using pure JavaScript. It can be used to create bindings to native libraries without writing any C++ code.

https://github.com/node-ffi/node-ffi

So node-ffi Knows

- How to execute function in C/C++ dynamic lib
- How to convert JavaScript data to C/C++ data
- How to send parameter
- How to grab return value

- By **@TooTallNate**

@TooTallNate

- **Nathan Rajlich**
- Node.js core committer
- One author of **Node.js in Action**
- Lots of ffi related package
- NodObjC, Node.js ⇆ Objective-C bridge

ffi Packages

ffi

ref

ref-struct

ref-array

ffi	Foreign Function Call
ref	Reference Tools
ref-struct	Struct Helper
ref-array	Array Helper

Example

```
var ffi = require('ffi');

var libm = ffi.Library('libm', {
  'ceil': [ 'double', [ 'double' ] ]
});

libm.ceil(1.5); // 2
```

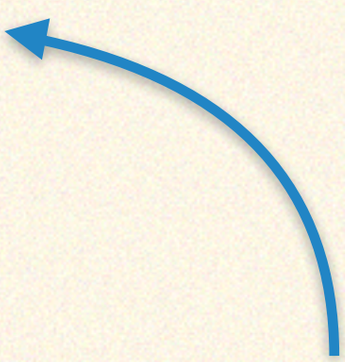
Example

```
var ffi = require('ffi');

var libm = ffi.Library('libm', {
  'ceil': [ 'double', [ 'double' ] ]
});

libm.ceil(1.5); // 2
```

Lib name
Use dlopen

Example

```
var ffi = require('ffi');

var libm = ffi.Library('libm', {
  'ceil': [ 'double', [ 'double' ] ]
});

libm.ceil(1.5); // 2
```



A function named ‘ceil’ return double
Take one double parameter

Example

```
var ffi = require('ffi');

var libm = ffi.Library('libm', {
  'ceil': [ 'double', [ 'double' ] ]
});

libm.ceil(1.5); // 2
```

return type

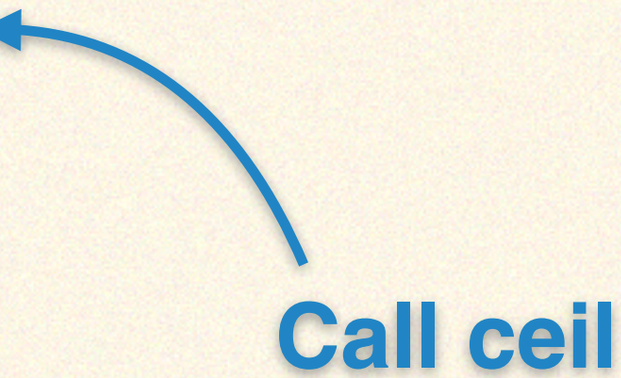
Example

```
var ffi = require('ffi');

var libm = ffi.Library('libm', {
  'ceil': [ 'double', [ 'double' ] ]
});

libm.ceil(1.5); // 2
```

arguments type

Example

```
var ffi = require('ffi');

var libm = ffi.Library('libm', {
  'ceil': [ 'double', [ 'double' ] ]
});

libm.ceil(1.5); // 2
```



To Use node-ffi

- Read header file(.h)
- Write `ffi.Library` definition
- Use it

math.h

```
double ceil (double x);
```

```
var libm = ffi.Library('libm', {
  'ceil': [ 'double', [ 'double' ] ]
});
```

http://www.cplusplus.com/reference/cmath/ceil/

Types

void bool byte pointer

(u)short (u)int (u)int16 (u)int32

(u)int64 (u)longlong size_t float

(u)char string

r3.h

```
     */
    void * data;
};

#define r3_node_edge_pattern(node,i) node->edges[i]->pattern
#define r3_node_edge_pattern_len(node,i) node->edges[i]->pattern_len

struct _edge {
    char * pattern; // 8 bytes
    node * child; // 8 bytes
    unsigned char  pattern_len; // 1 byte
    unsigned char  opcode:4; // 4 bit
    unsigned char  has_slug:1; // 1 bit
};

struct _route {
    char * path;
    int    path_len;

    int    request_method; // can be (GET || POST)

    char * host; // required host name
    int    host_len;

    void * data;

    char * remote_addr_pattern;
    int    remote_addr_pattern_len;
};

typedef struct {
    str_array * vars;
    const char * path; // current path to dispatch
    int    path_len; // the length of the current path
    int    request_method;  // current request method
```

```
     */
    void * data;
};

#define r3_node_edge_pattern(node,i) node->edges[i]->pattern
#define r3_node_edge_pattern_len(node,i) node->edges[i]->pattern_len

struct _edge {
    char * pattern; // 8 bytes
    node * child; // 8 bytes
    unsigned char  pattern_len; // 1 byte
    unsigned char  opcode:4; // 4 bit
    unsigned char  has_slug:1; // 1 bit
};

struct _route {
    char * path;
    int    path_len;

    int    request_method; // can be (GET || POST)

    char * host; // required host name
    int    host_len;

    void * data;

    char * remote_addr_pattern;
    int    remote_addr_pattern_len;
};

typedef struct {
    str_array * vars;
    const char * path; // current path to dispatch
    int    path_len; // the length of the current path
    int    request_method;  // current request method
```

Struct?

Struct

- A struct is actually a pointer
- Ok to use pointer
- Not ok to use pointer if want to load its property
- Use ref-struct

```c
struct _edge {
    char * pattern; // 8 bytes
    node * child; // 8 bytes
    unsigned char pattern_len; // 1 byte
    unsigned char opcode:4; // 4 bit
    unsigned char has_slug:1; // 1 bit
};
```

```c
var StructType = require('ref-struct');

var edge = StructType({
    pattern: "string",
    child: "pointer",
    pattern_len: "ushort",
    opcode: "uchar",
    has_slug: "uchar"
});
```

ref-struct

- Create ABI-compliant "struct" instances on top of Buffers

- ABI(Application Binary Interface):
  - the sizes, layout, and alignment of data types
  - ...

https://en.wikipedia.org/wiki/Application_binary_interface

Itanium C++ ABI

Contents

Acknowledgements

This document was developed jointly by an informal industry coalition consisting of (in alphabetical order) CodeSourcery, Compaq, EDG, HP, IBM, Intel, Red Hat, and SGI. Additional contributions were provided by a variety of individuals.

Chapter 1: Introduction

C++ ABI Summary

Revised 20 March 2001

Links

Status pages:

- http://www.codesourcery.com/cxx-abi/: External link to this page

- Full open issues list
- Full closed issues list

Mailing Lists:

- On-Line Archives

Drafts:

- Draft C++ ABI for Itanium
- Draft psABI exception handling specification
- ABI Examples code samples and tests

- Sample GCC code:
  - typeinfo: class type_info declaration
  - cxxabi.h: ABI support type_info classes and prototypes
  - vec.cc: ABI vector new and delete helpers
  - tinfo.h: runtime header for tinfo.cc & tinfo2.cc
  - tinfo.cc: ABI dynamic cast and catch matching routines
  - tinfo2.cc: ABI non-class type_info definitions

Proposals and auxiliary information:

- Proposals to the base ABI group for COMDAT sections and removing limitations on section indices.
- SGI Interface Section proposal.
- Vtable layout examples.
- WG21 argument destruction writeup (PDF)

Objectives

- Interoperable C++ compilation on Itanium: we want users to be able to build relocatable objects with different compilers and link them together, and if possible even to ship common DSOs. This objective implies agreement on:
  - Data representation
  - Object file representation

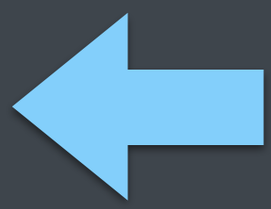
https://mentorembedded.github.io/cxx-abi/

Struct in Struct?

```c
typedef struct {
    str_array * vars;
    const char * path; // current path to dispatch
    int     path_len; // the length of the current path
    int     request_method; // current request method
    void * data; // route ptr
    char * host; // the request host

...
```

```
typedef struct {
    str_array * vars;
    const char * path; // current path to dispatch
    int     path_len; // the length of the current path
    int     request_method; // current request method
    void * data; // route ptr
    char * host; // the request host

...
```

```
var ref = require('ref');

var str_array = StructType({ ... });

var match_entry = StructType({
    vars: ref.refType(str_array),
    path: "string",
    path_len: "int",
    request_method: "int",
    data: "pointer",
    host: "pointer",
    ...
```



ref is a native addon for Node.js
that aids in doing C programming
in JavaScript, by extending the
built-in Buffer class

https://tootallnate.github.io/ref/

ref v0.2.0

Turn Buffer instances into "pointers"

What is `ref`?

`ref` is a native addon for Node.js that aids in doing C programming in JavaScript, by extending the built-in `Buffer` class with some fancy additions like:

- Getting the memory address of a Buffer
- Checking the endianness of the processor
- Checking if a Buffer represents the NULL pointer
- Reading and writing "pointers" with Buffers
- Reading and writing C Strings (NULL-terminated)
- Reading and writing JavaScript Object references
- Reading and writing **int64_t** and **uint64_t** values
- A "type" convention to define the contents of a Buffer

There is indeed a lot of *meat* to `ref`, but it all fits together in one way or another in the end. For simplicity, `ref`'s API can be broken down into 3 sections:

`ref` exports

All the static versions of `ref`'s functions and default "types" available on the exports returned from `require('ref')`.

"type" system

The "*type*" system allows you to define a "type" on any Buffer instance, and then use generic `ref()` and `deref()` functions to reference and dereference values.

`Buffer` extensions

`Buffer.prototype` gets extended with some convenience functions. These all just mirror their static counterpart, using the Buffer's `this` variable as the `buffer` variable.

ref.refType

- Get reference to data type

Returns a new clone of the given "type" object, with its indirection level incremented by 1.

https://tootallnate.github.io/ref/

ref.refType

- Get reference to data type

- `indirection` means the count of `*`

ref.refType

```
str_array   var1;
str_array * var2;
```

```
StructType({
    var1: str_array,
    var2: ref.refType(str_array)
});
```

ref.refType

```
str_array   var1;
str_array * var2;
```

```
StructType({
    var1: str_array,
    var2: ref.refType(str_array)
});
```

ref.refType

```
str_array   var1;
str_array * var2;
```

```
StructType({
    var1: str_array,
    var2: ref.refType(str_array)
});
```

Back to r3.h

```
node * r3_tree_create(int cap);

node * r3_node_create();

void r3_tree_free(node * tree);

edge * r3_node_connectl(node * n, const char * pat, int len, int strdup, node *child);

#define r3_node_connect(n, pat, child) r3_node_connectl(n, pat, strlen(pat), 0, child)

edge * r3_node_find_edge(const node * n, const char * pat, int pat_len);

void r3_node_append_edge(node *n, edge *child);


edge * r3_node_find_common_prefix(node *n, const char *path, int path_len, int *prefix_len, char **errstr);

node * r3_tree_insert_pathl(node *tree, const char *path, int path_len, void * data);

#define r3_tree_insert_pathl(tree, path, path_len, data) r3_tree_insert_pathl_ex(tree, path, path_len, NULL , data, NULL


route * r3_tree_insert_routel(node *tree, int method, const char *path, int path_len, void *data);

route * r3_tree_insert_routel_ex(node *tree, int method, const char *path, int path_len, void *data, char **errstr);

#define r3_tree_insert_routel(n, method, path, path_len, data) r3_tree_insert_routel_ex(n, method, path, path_len, data,

#define r3_tree_insert_path(n,p,d) r3_tree_insert_pathl_ex(n,p,strlen(p), NULL, d, NULL)

#define r3_tree_insert_route(n,method,path,data) r3_tree_insert_routel(n, method, path, strlen(path), data)


/**
```

```
110  node * r3_tree_create(int cap);
111
112  node * r3_node_create();
113
114  void r3_tree_free(node * tree);
115
116  edge * r3_node_connectl(node * n, const char * pat, int len, int strdup, node *child);
117
118  #define r3_node_connect(n, pat, child) r3_node_connectl(n, pat, strlen(pat), 0, child)
119
120  edge * r3_node_find_edge(const node * n, const char * pat, int pat_len);
121
122  void r3_node_append_edge(node *n, edge *child);
123
124
125  edge * r3_node_find_common_prefix(node *n, const char *path, int path_len, int *prefix_len, char **errstr);
126
127  node * r3_tree_insert_pathl(node *tree, const char *path, int path_len, void * data);
128
129  #define r3_tree_insert_pathl(tree, path, path_len, data) r3_tree_insert_pathl_ex(tree, path, path_len, NULL , data, NULL
130
131
132
133  route * r3_tree_insert_routel(node *tree, int method, const char *path, int path_len, void *data);
134
135  route * r3_tree_insert_routel_ex(node *tree, int method, const char *path, int path_len, void *data, char **errstr);
136
137  #define r3_tree_insert_routel(n, method, path, path_len, data) r3_tree_insert_routel_ex(n, method, path, path_len, data,
138
139  #define r3_tree_insert_path(n,p,d) r3_tree_insert_pathl_ex(n,p,strlen(p), NULL, d, NULL)
140
141  #define r3_tree_insert_route(n,method,path,data) r3_tree_insert_routel(n, method, path, strlen(path), data)
142
143
144  /**
```

#define

- C preprocessor
- Replace the string in source codes
- Process before compile
- Not real function

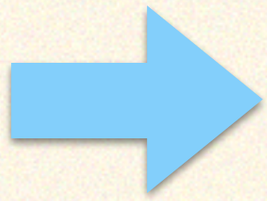
```
#define r3_tree_insert_path(n,p,d) r3_tree_insert_pathl_ex(n,p,s
```

```
var libr3 = ffi.Library('libr3', {
  "r3_tree_insert_pathl_ex": ["pointer", ["pointer", "string", "
});


var r3_tree_insert_path = function (tree, path, data) {
  return libr3.r3_tree_insert_pathl_ex(tree, path, path.length,
};
```

Back to r3.h

r3.h

- 200+ lines
- 4 structs
- 30+ methods



Lots of work

node-ffi-generate

- Automatic generate ffi definition from .h file
- Cons:
  - Result is too detail, too big
  - Might not suit for use

https://github.com/tjfontaine/node-ffi-generate

node-r3

- Only define necessary methods
- 10 methods
  - Tree create, compile, dump, match, free
  - Insert route, path
  - Match_entry create, free

Now We Can Write ffi.Library Define for r3

And Start Use r3 in Node.js

哪會這麼順利

Route Data

```
route * r3_tree_insert_routel(node *tree, int method,
    const char *path, int path_len, void *data);
```

```
route * r3_tree_insert_routel(node *tree, int method,
    const char *path, int path_len, void *data);
```

Route Data

- Route data is void pointer
- Can point to any type of data
- libr3 only store the pointer

Pointer?

Pointer in JS

- There is no real pointer in JavaScript

- But you can use Buffer

Buffer

- Node.js only
- Buffer is a chunk of memory in specified address
- Can be typeless, size can be 0
- node-ffi deal pointer as buffer

Example 1/2

```
var tree = libr3.r3_tree_create(10);

var data = new Buffer(data_str + '\u0000');

r3_tree_insert_route(tree, 'GET','/path', data);
```

Example 2/2

```
var node = libr3.r3_tree_match_route(tree, entry);

var data = ref.deref(node).data;
```

https://github.com/othree/node-r3/blob/46e29f72bbf68097ed66fd151f9ef42d5a3ede05/node-r3.js

deref

- Dereference
- Get value from buffer
- In this case, deref will get a struct.
  And its data attribute is what we want.

The Data

- The data we got is another buffer instance
- We send pointer of this buffer to r3 when insert route
- We need to dereference it to get real data value

BUT

The Problem

- Dereference `data` will get nothing
- Because the size of `data` is 0 and the type is unknown

Solution 1

- Set data type and size
  - `buffer.type`
  - `ref.reinterpret`

reinterpret

- Resize buffer

- You need to know the size of your data

Cons

- Hard to implement various data type
- If you do it, might hurt performance
- Hard to design API

Solution 2

- Always use string

- `ref.readCString`

readCString

- Read string from buffer, until `00`

- Not stop when meet buffer end

```
var data = new Buffer(data_str + '\u0000');
```

```
var data = ref.readCString(ref.deref(node).data, 0);
```

Cons

- Route data is limited to string type

- After solve first issue
- We have first usable node-r3
- But the performance is very bad
- Slower than director

Performance

Spec

- Use the same tree in r3/tests/bench
- Run on iMac
  - 2.66 GHz Intel Core i5
  - 16 GB 1333 MHz DDR3

https://github.com/c9s/r3/blob/master/tests/bench.c

Route	req/s
director	~2200
node-r3 0.0.1	~~~1950~~

For unknown reason
I can not reproduce this number

What Takes Time

- Data transform
- Call foreign function

Transform String

- New buffer from given data string
- Read C string from pointer

```
data = new Buffer(data + '\u0000');
r3_tree_insert_path(this.tree, route, data);
```

```
var node = r3_tree_match(this.tree, path, entry);

var data = ref.readCString(node.deref().data, 0);
```

How to Skip Transform

- Store data in JavaScript side
- Use integer index as data

```
var i = this.i++;

this.data[i] = data;

libr3.r3_tree_insert_route(this.tree, method, route, i);
```

```
var node = libr3.r3_tree_match_route(this.tree, entry);

var index = node.deref().readUInt32LE(0);

var data = this.data[index];
```

Route	req/s
director	~2200
node-r3 0.0.1	~1900
node-r3 0.0.3	~5000

More Pros

- No more limitation on data type
- All JS native data type can be use

How About Foreign Calls

- What we used:
  - r3_tree_create, r3_tree_free
  - r3_tree_insert_path
  - r3_tree_compile
  - match_entry_create, match_entry_free
  - r3_tree_match_route

What is Omissible

- `match_entry_create`, `match_entry_free`
- Only required when:
  - Use HTTP method constraint
  - Capture argument

So

- I can omit these methods when not need it

Route	req/s
director	~2200
node-r3 0.0.1	~1900
node-r3 0.0.3	~5000
node-r3 0.1.0	7800~10600 **Not Yet**

- After solve second issue
- We have faster node-r3
- But it can not pass pressure test
- Data index becomes random number, and crash after more request

Crash…

2. [zsh] othreedeimac.htctaipei.htc.com.tw (zsh)

```
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
16806400

/Users/othree/os-project/node-r3/node-r3.js:210
      result[0].apply(this, [req, res, result[1]]);
                ^
TypeError: Cannot call method 'apply' of undefined
    at Server.<anonymous> (/Users/othree/os-project/node-r3/node-r3.js:210:17)
    at Server.EventEmitter.emit (events.js:98:17)
    at HTTPParser.parser.onIncoming (http.js:2108:12)
    at HTTPParser.parserOnHeadersComplete [as onHeadersComplete] (http.js:121:23)
    at Socket.socket.ondata (http.js:1966:22)
    at TCP.onread (net.js:527:27)
[/Users/othree/os-project/node-r3|git:c174f4d...] -othree- |1|>>> _                [ 8 ]
```

2. [zsh] othreedeimac.htctaipei.htc.com.tw (zsh)

```
    (o) regexp:^([^/]+) endpoint:0
      |-"{post_id}" opcode:3
      (o) endpoint:1 data:0x103003190


    |-"getme"
    (o) endpoint:0
      |-"2"
      (o) endpoint:1 data:0x103003220

      |-"1"
      (o) endpoint:1 data:0x1030032a0

      |-"3"
      (o) endpoint:1 data:0x103003320


    |-"ugetme"
    (o) endpoint:0
      |-"1"
      (o) endpoint:1 data:0x1030033a0

      |-"2"
      (o) endpoint:1 data:0x103003428

      |-"3"
      (o) endpoint:1 data:0x1030034b0



Listen on http://localhost:5000/
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 zsh: segmentation fault  no
de sample/http.js
[/Users/othree/os-project/node-r3|git:8d3cb8b...] -othree- |1|>>> _                                  [ 139 ]
```

Hard To Find The Bug

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCase()];
      if (!method) { throw new Error(route_frag[0]); }
      r3_tree_insert_route(this.tree, method, route, data);
    } else {
      r3_tree_insert_route(this.tree, 0, route, data);
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

variables

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCa
      if (!method) { throw new Error(route_fra
      r3_tree_insert_route(this.tree, method,
    } else {
      r3_tree_insert_route(this.tree, 0, route
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

create tree

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCa
      if (!method) { throw new Error(route_fra
      r3_tree_insert_route(this.tree, method,
    } else {
      r3_tree_insert_route(this.tree, 0, route
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

data storage

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCa
      if (!method) { throw new Error(route_fra
      r3_tree_insert_route(this.tree, method,
    } else {
      r3_tree_insert_route(this.tree, 0, route
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

parse route

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCa
      if (!method) { throw new Error(route_fra
      r3_tree_insert_route(this.tree, method,
    } else {
      r3_tree_insert_route(this.tree, 0, route
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

create index

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCa
      if (!method) { throw new Error(route_fra
      r3_tree_insert_route(this.tree, method,
    } else {
      r3_tree_insert_route(this.tree, 0, route
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

parse string

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCa
      if (!method) { throw new Error(route_fra
      r3_tree_insert_route(this.tree, method,
    } else {
      r3_tree_insert_route(this.tree, 0, route
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

method condition

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCa
      if (!method) { throw new Error(route_fra
      r3_tree_insert_route(this.tree, method,
    } else {
      r3_tree_insert_route(this.tree, 0, route
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

insert route

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCa
      if (!method) { throw new Error(route_fra
      r3_tree_insert_route(this.tree, method,
    } else {
      r3_tree_insert_route(this.tree, 0, route
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

compile tree

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCa
      if (!method) { throw new Error(route_fra
      r3_tree_insert_route(this.tree, method,
    } else {
      r3_tree_insert_route(this.tree, 0, route
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

Any Idea?

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCase()];
      if (!method) { throw new Error(route_frag[0]); }
      r3_tree_insert_route(this.tree, method, route, data);
    } else {
      r3_tree_insert_route(this.tree, 0, route, data);
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

The Bug

- `data` is local variable

- Will be free during Garbage Collection

```
var Router = function (routes) {
  var route, data, method, route_frag, i = 0;
  this.tree = libr3.r3_tree_create(10);
  this.data = [];
  this.index = [];
  for (route in routes) {
    this.data[i] = routes[route];
    data = ref.alloc('int', i).ref();
    this.index[i] = data; // prevent GC
    route = route.trim();
    route_frag = route.split(' ');
    if (route_frag.length > 1) {
      route = route_frag[1];
      method = METHODS[route_frag[0].toUpperCase()];
      if (!method) { throw new Error(route_frag[0]); }
      r3_tree_insert_route(this.tree, method, route, data);
    } else {
      r3_tree_insert_route(this.tree, 0, route, data);
    }
    i++;
  }
  libr3.r3_tree_compile(this.tree);
  return this;
};
```

So

- 自己的變數自己記
- 記得 JavaScript 有 GC

- After solve all 3 issues
- We finally have a really stable, high performance router lib
- And thats 0.0.3 release
- Now is 0.0.9

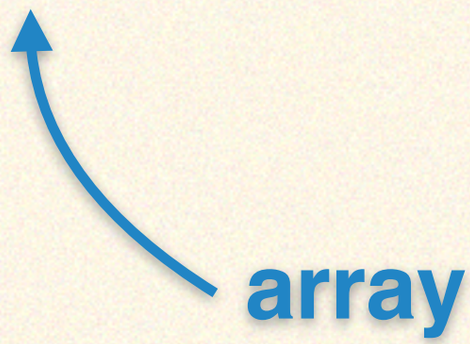
node-r3

```
var Router = require('node-r3').Router;

var router = new Router({
  "/foo": "data string",
  "/foo/bar": function () {},
  "/foo/bar/qoo": {obj: 1},
});

var dispatched = router.match("/foo");

router.free();
```

```
var router = new Router({
  "/": handler,
  "/foo": fooHandler,
  "/foo/{id}": fooHandler,
  "POST /me": postMeHandler,
  "GET /me": getMeHandler,
  "POST|GET /post": postHandler,
});

var server = http.createServer(
    router.httpHandler(notfound)
);
```

Handler Function

```
function (req, res, captures)
```



Error Handler Function

```
function (req, res)
```

Known Issues

- Memory leak?
- Performance improvement

Question?

Thanks For your Attention