OpenQM

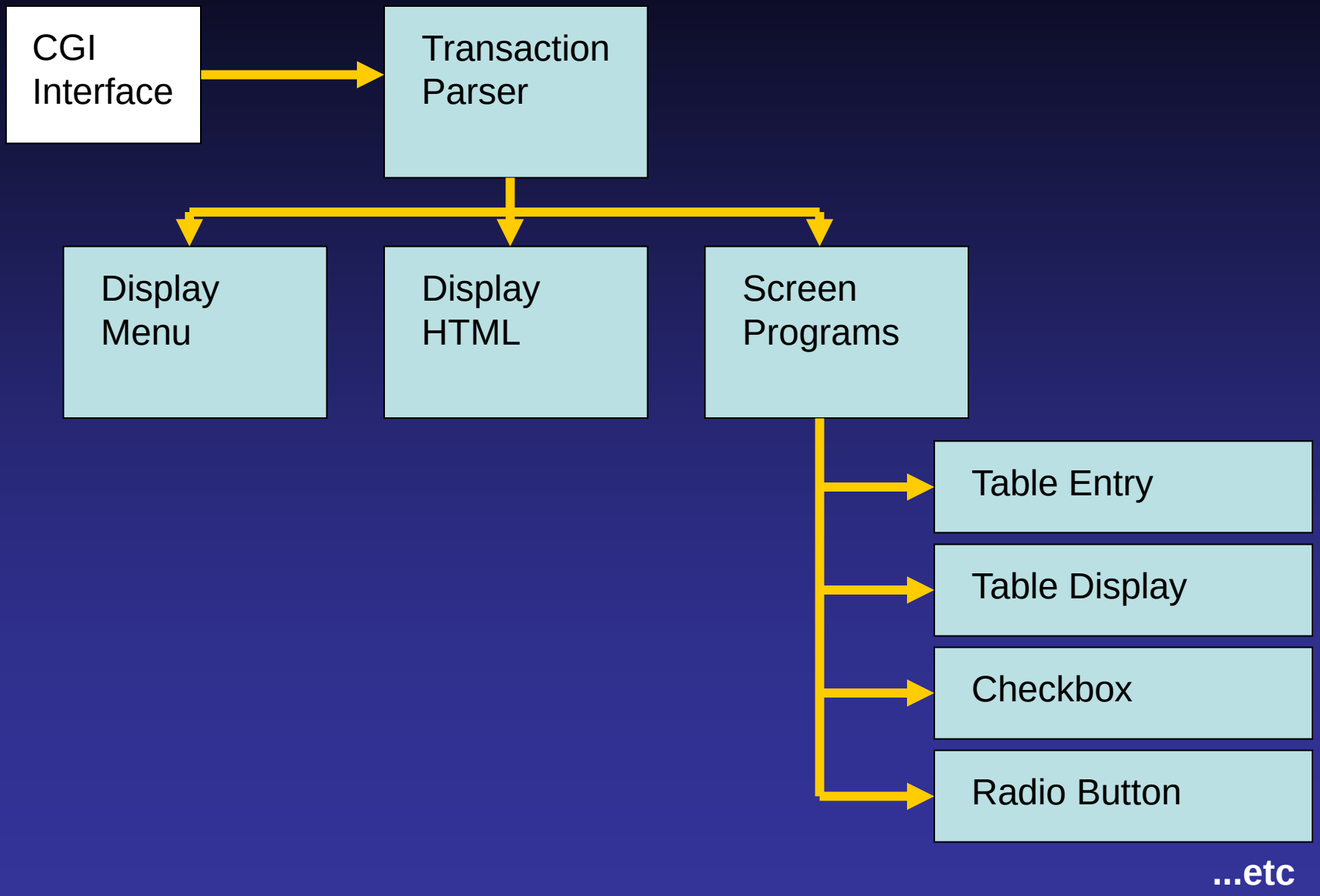
Building a CGI Web Server

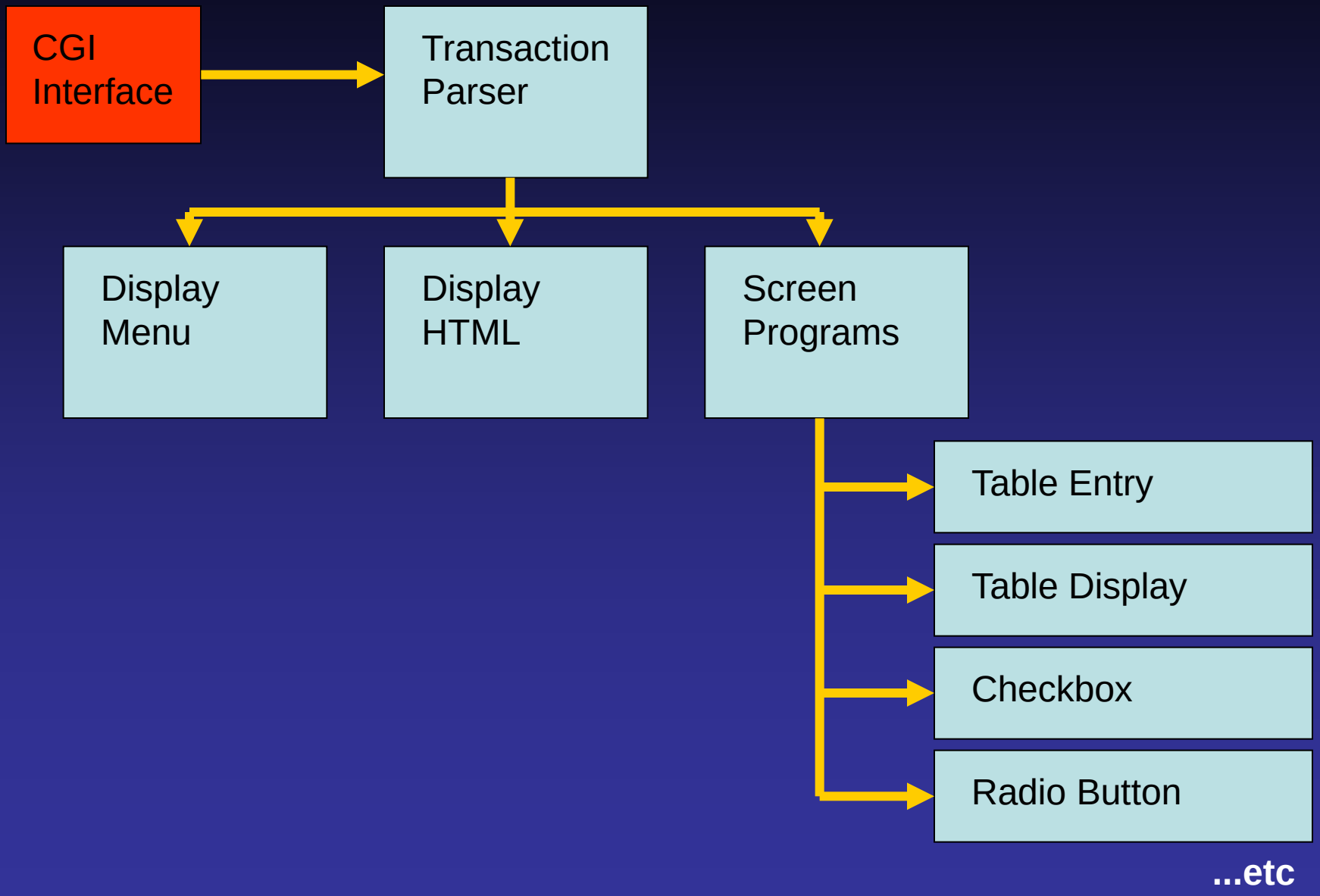
Martin Phillips
Ladybridge Systems Ltd

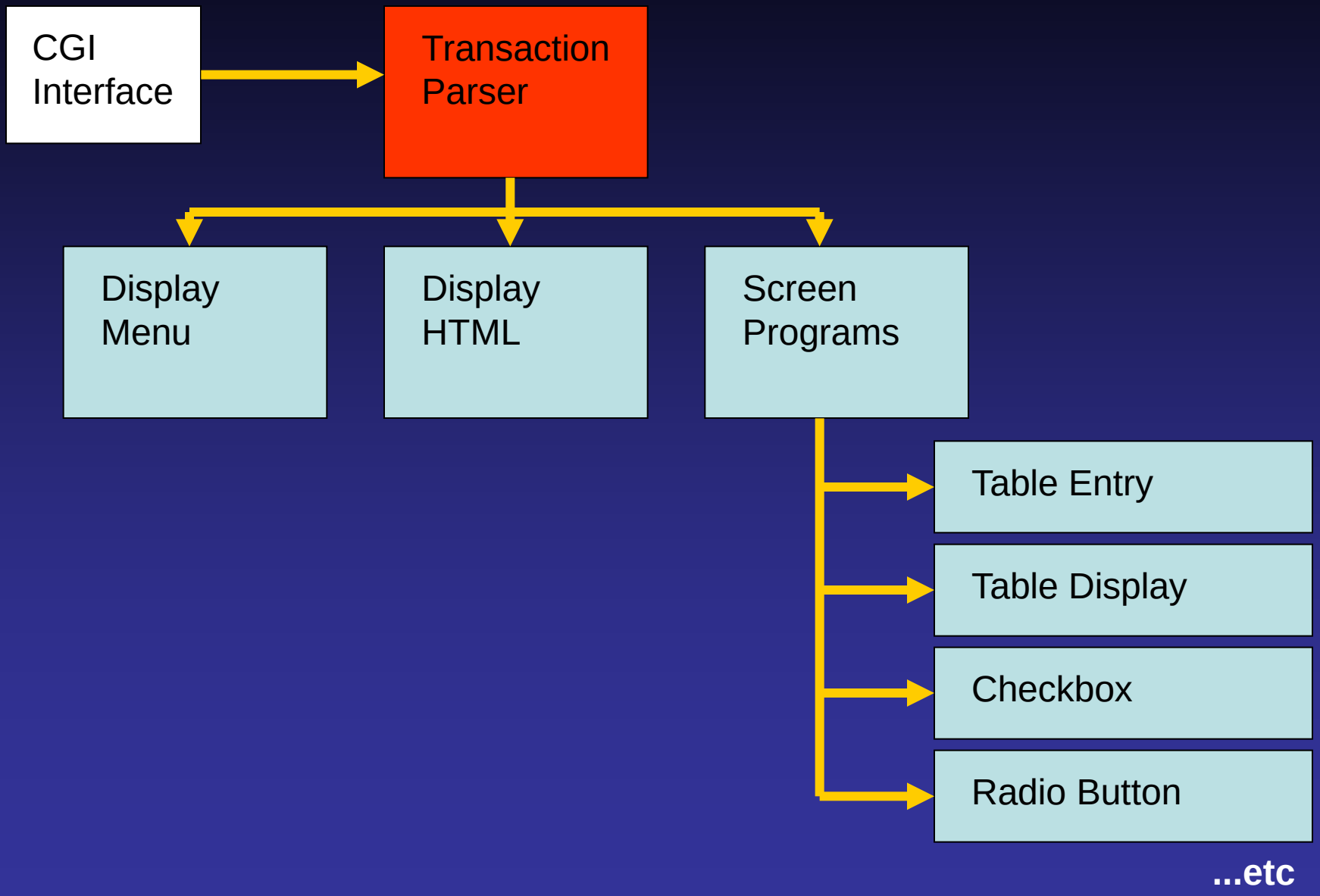
What We Are Going To See Today....

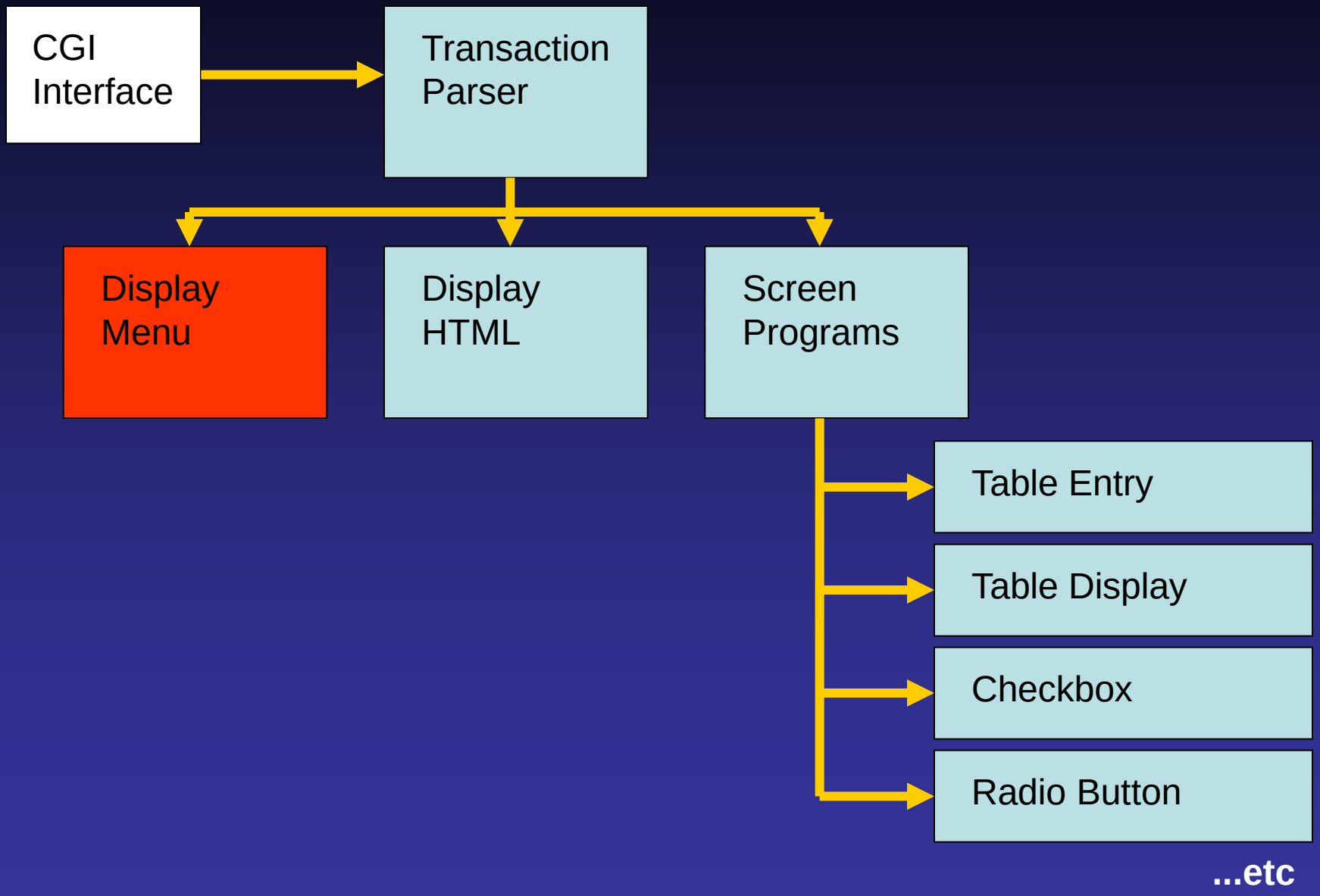
How to build a web server as a multi-value Basic application using no additional tools.

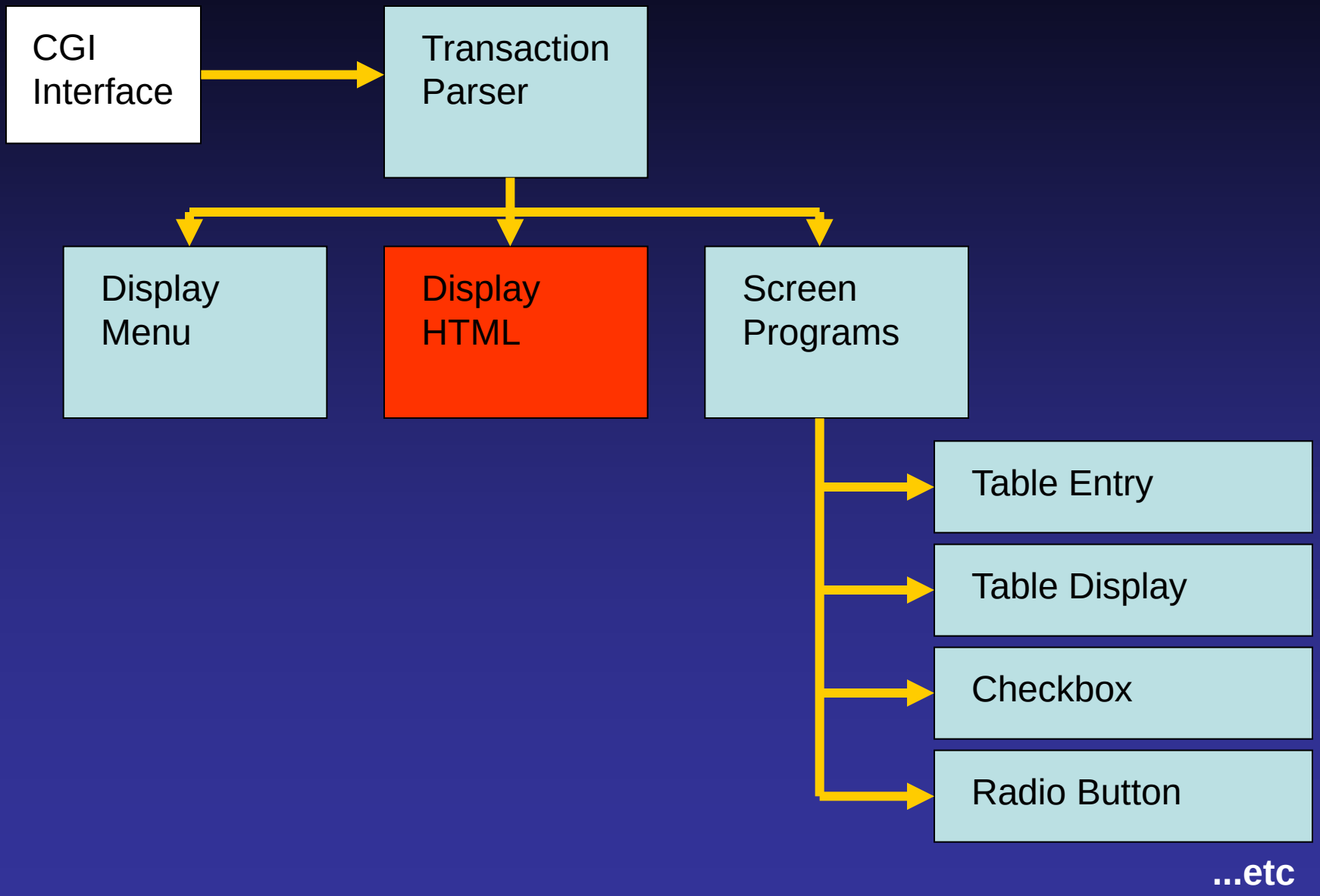
These examples are based on QM but can be adapted for other environments.

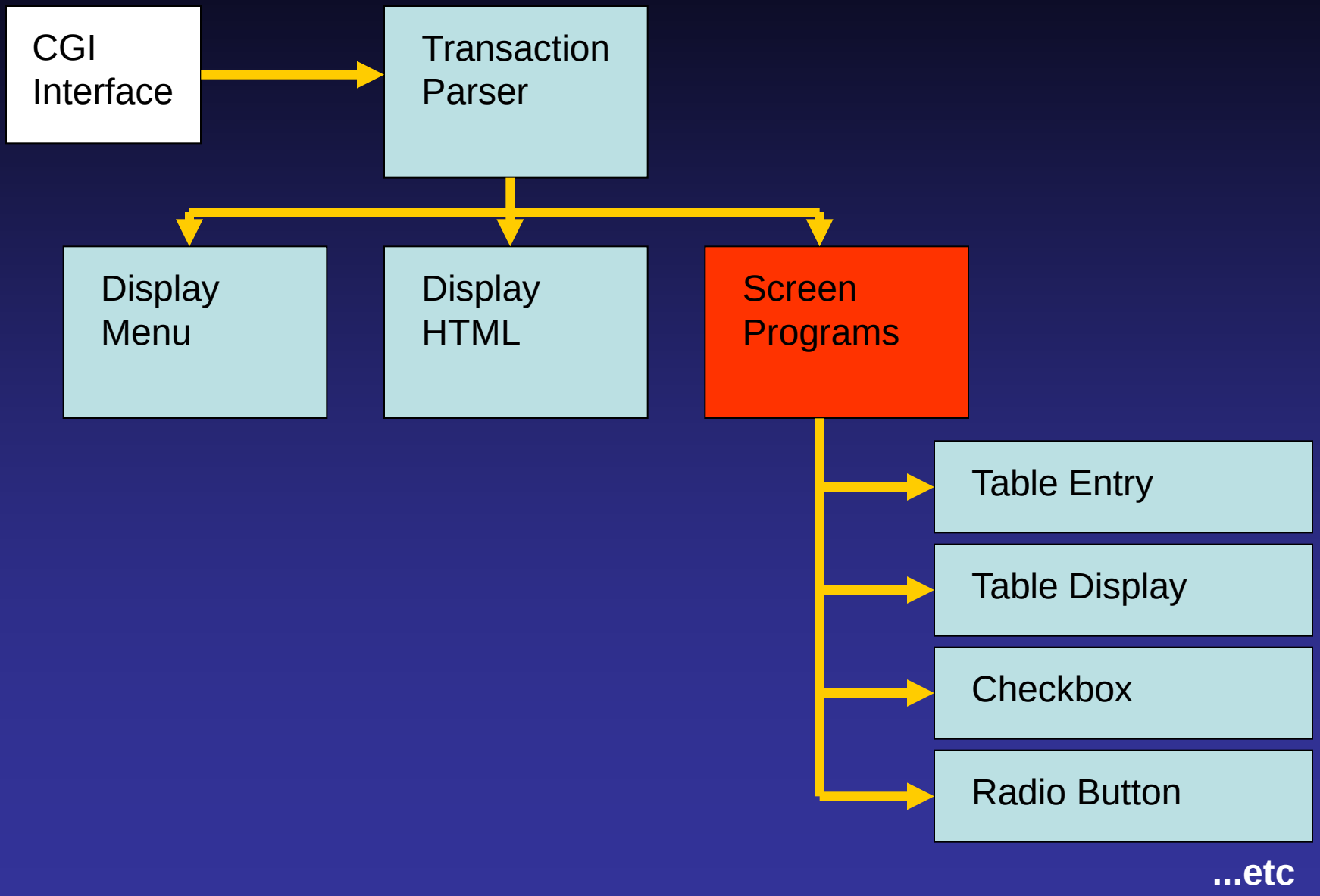

CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc

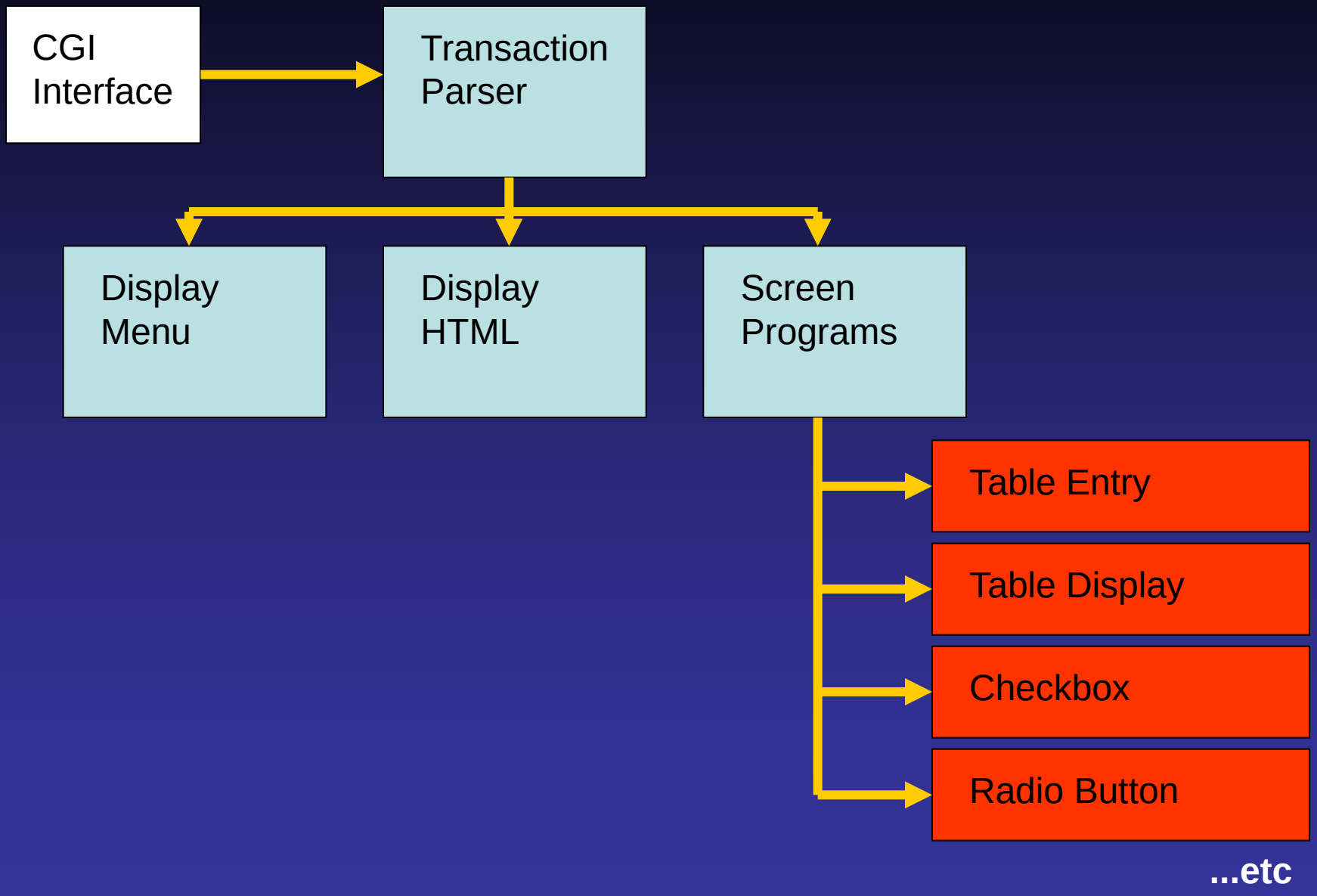

CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc


CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc


CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc

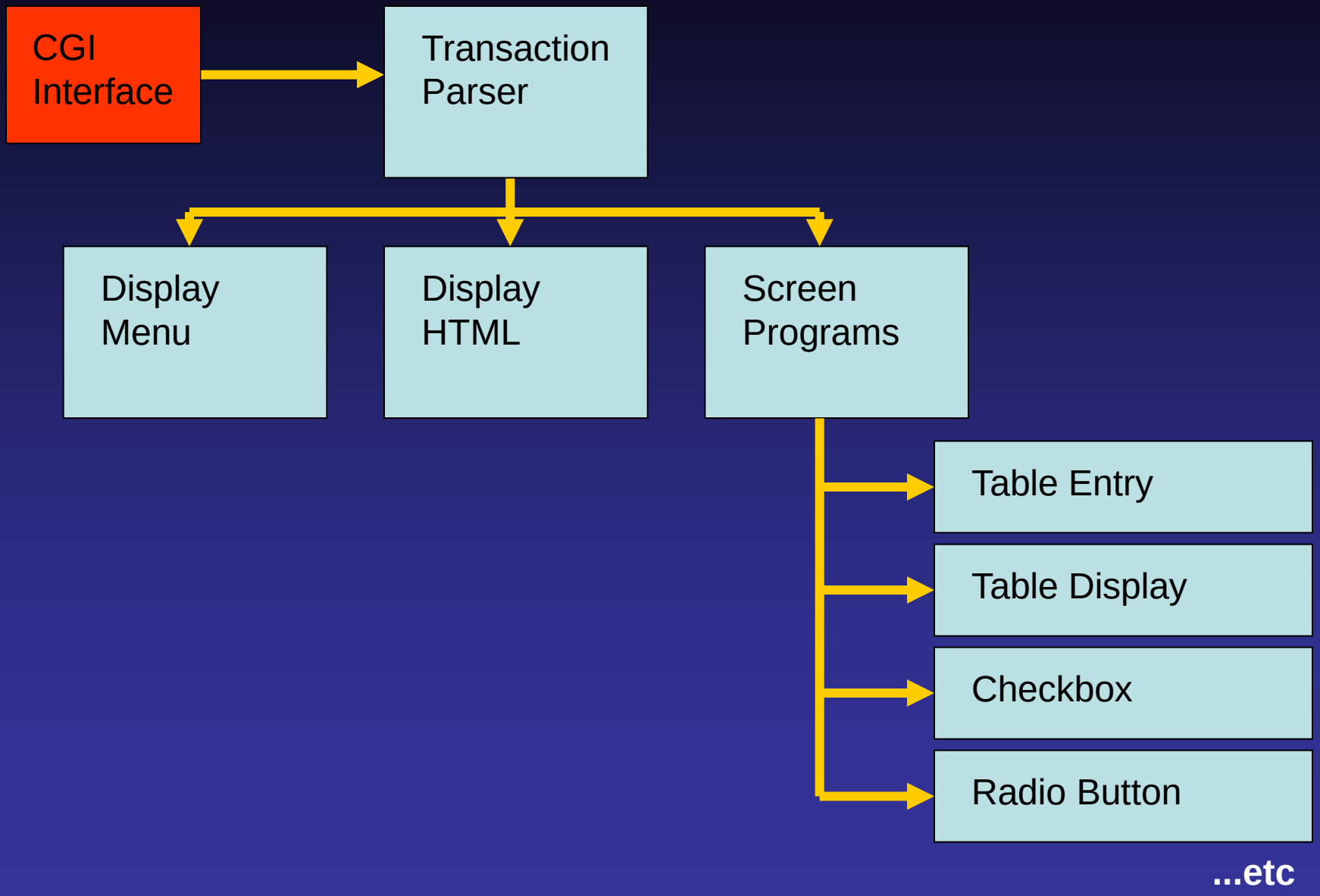

CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc


CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc


CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc

Web Page Generation Data Files

USERS	User authentication
SESSIONS	Persistent data management
HTML	Template HTML pages
MENUS	Dynamic menu content
LOG	Diagnostic transaction log


CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc

The CGI Interface Program

mysite.com/cgi/cgi.exe?t0=m&t1=links&x=jlfo9d9pqn

URL

All text before the ? is the web address of the CGI program.

Parameters

All items after the ? are parameters separated by ampersands.

The CGI Interface Program

The parameters and other data are passed to the C program via environment variables:

REQUEST_METHOD	GET or POST
REMOTE_ADDR	IP address of client
HTTP_HOST	Domain name from URL
QUERY_STRING	Parameters for GET
CONTENT_LENGTH	Data length for POST

The CGI Interface Program

The C program opens a database connection and calls the parser subroutine, passing in the parameters from the incoming message.

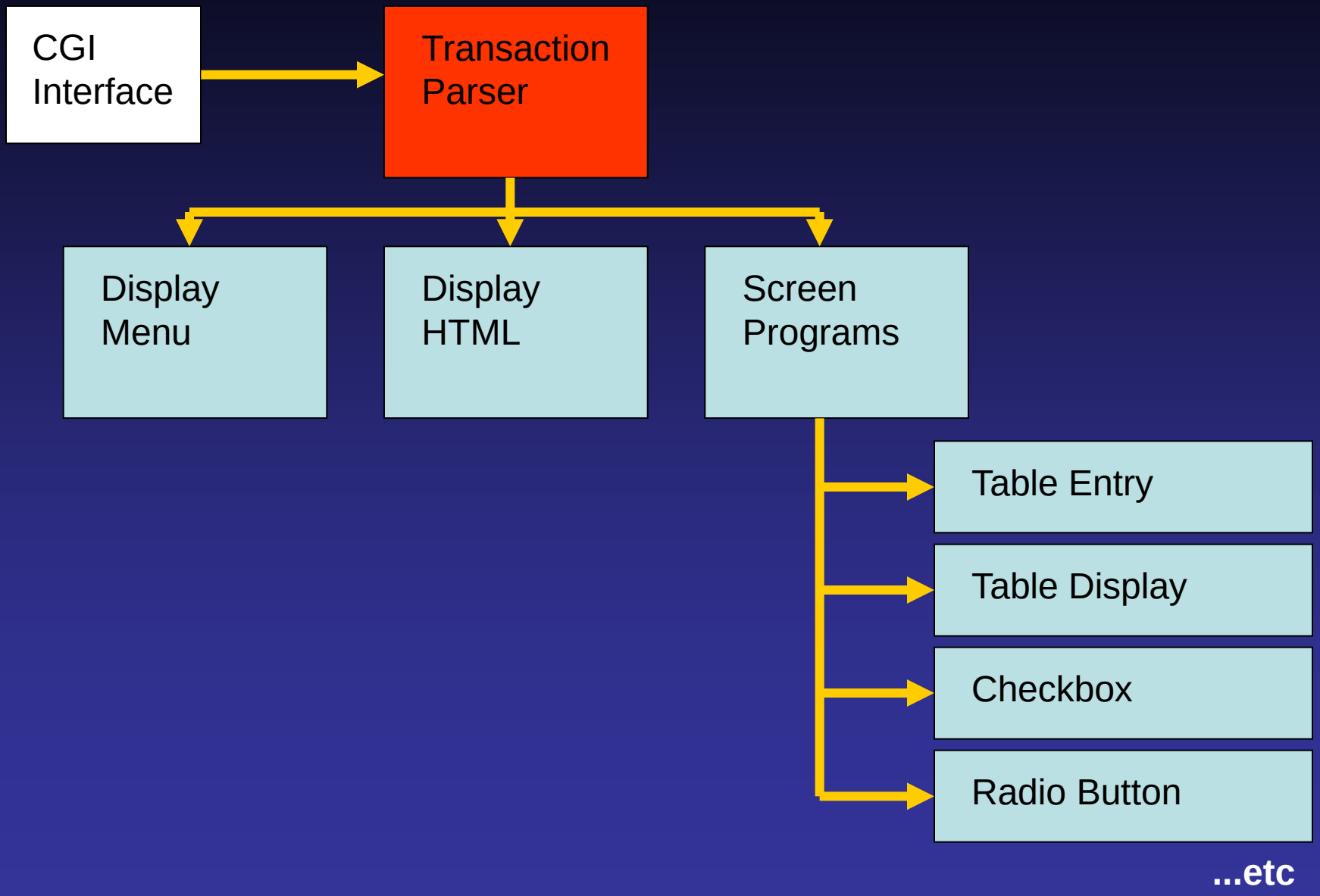
The response is passed back from the subroutine through an argument variable.

For long responses, the data is written to a temporary file and the pathname is returned by the subroutine.

The C program sends the response back to the client browser.

The CGI Interface Program

```
if (!QMConnect(SERVER_ADDRESS, SERVER_PORT,
                    SERVER_USER, SERVER_PASSWORD,
                    SERVER_ACCOUNT))
  {
   strcpy(Response, "The server may be offline.");
  }
 else
  {
   QMCall("CGI", 3, InputData, Params, Response);
   QMDisconnect();
  }
```


CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc

The Transaction Parser

The parameters in the incoming message are entirely under application control.

Our use is:

T*n*	Text item
B*n*	Button
C*n*	Checkbox
R*n*	Radio button
X	Session id

The Transaction Parser

The parser copies the parameter values to fields within dynamic arrays for each data type.

Special encoding of restricted characters is handled during this operation.

Our parser also supports multi-valued parameters but we will ignore this here.

We use T0 to identify the program to be executed to handle the incoming request.

The Transaction Parser

For example:

T0 = M	Menu action
T0 = H	Template HTML page display
T0 = *xxx*	Execute named program

The name can have a "phase" number appended to represent the stage in a multi-screen sequence.

OpenQM

Powered by QM on a Windows server

Generate Evaluation Licence

A thirty day, four user evaluation licence is available without charge. Please complete the details below and an authorisation code will be returned by email.

The software can be downloaded from this web site. A CD version can be supplied but there is a small charge for this.

Items marked * are mandatory.

Your name
Company
Address
City
State/County
Postcode/Zip
Country -Select-
Email
Telephone
Fax
System -Select-

Please tell us where you heard about OpenQM * -Select-

☐ Tick here to subscribe this user to the QM Newsletter.
☐ Tick here for this user to be notified by email of new releases
☐ Tick here if we may pass your details to the dealer/distributor for your region

Submit Reset

The Transaction Parser

The parser constructs the new page by merging:

- Fixed text (style definitions, banner, etc)
- The menu bar
- The page body

This is returned to the CGI interface program via an argument variable or a temporary file.

The Transaction Parser – Session Ids

Web transactions are separate events with no automatic persistence of data.

We need to track some persistent data:

- User authentication and access level
- Displayed menus

Each connection is given a random session id that is carried forwards from one transaction to the next.

The Transaction Parser – Session Ids

Persistent data is stored in the SESSIONS file.

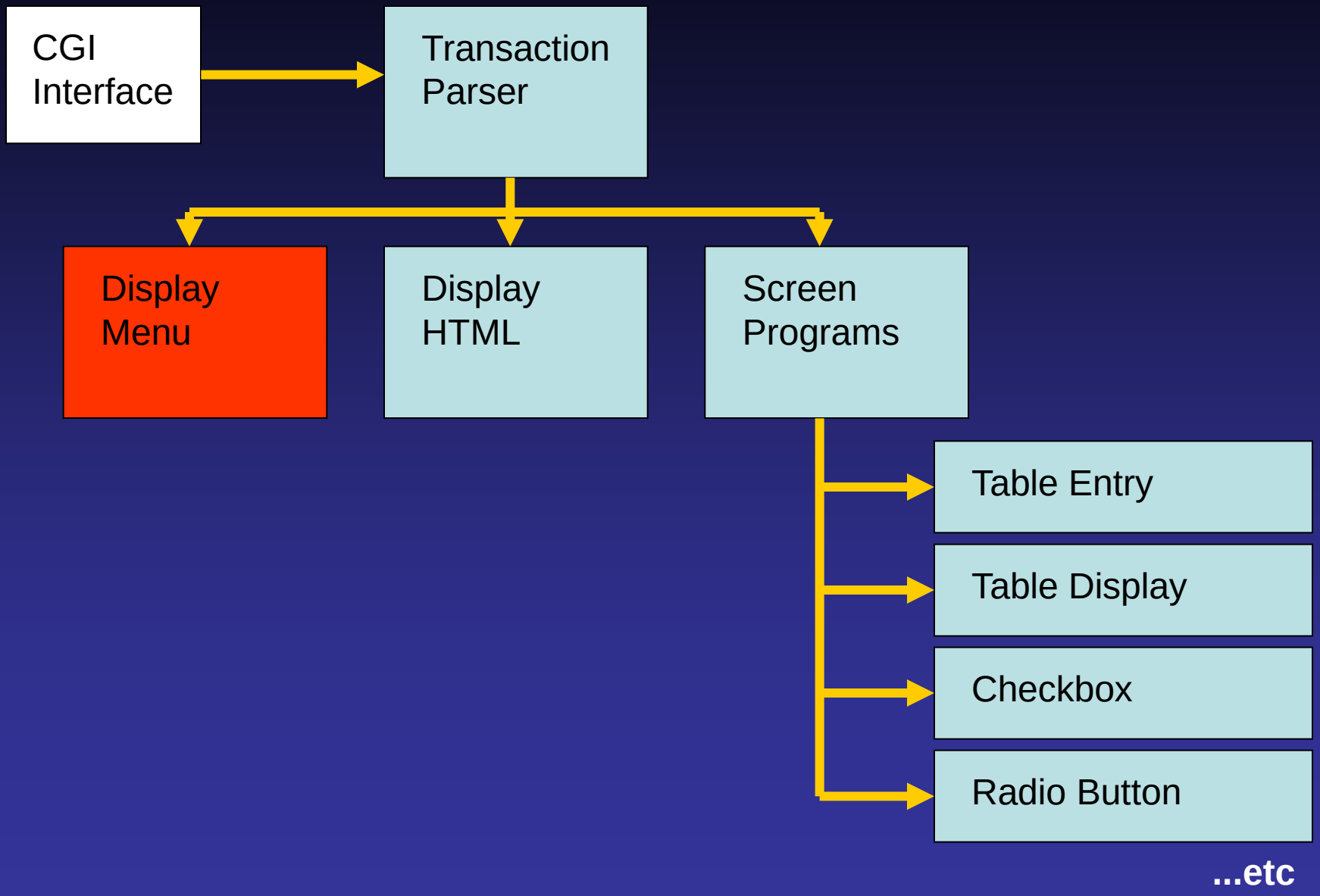
The X parameter links to the session record.

This is cross-checked against IP address, etc.

Every action checks the user's access level.

Session ids timeout after one hour of inactivity.

Old sessions are cleared out periodically.


CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc

Displaying Menus

We use a two level menu system where clicking on a top level item expands/collapses it.

The name of the top level menu to be displayed is stored in the SESSIONS file.

The underlying menu system can support multiple levels.

The expand/collapse action may also change the displayed page.

OpenQM

Powered by QM on a Windows server

About OpenQM

Sales and Downloads

Dealers Area

Help and Support

Privacy Policy

Links

Contact Us

Home

Taking Multivalue Where It Has Never Been Before...

OpenQM is the ***only*** multivalue database available both as a fully supported commercial product and in open source form under the General Public Licence.

The name OpenQM is a generic term used to describe both the open source and the commercial 'closed source' versions of the QM database.

The **commercial version** is available on Windows, Linux (Red Hat, Fedora, Debian, Ubuntu), FreeBSD, Mac OS X and Windows Mobile. A list price of less than one fifth of the price of some other multivalue products and no mandatory support contracts makes the total cost of ownership of an OpenQM database highly attractive. The licence includes 60 days of free support as well as 1 year of free upgrades or platform changes. Continued support is available from the distributor, dealer or directly from Ladybridge Systems at highly competitive prices based on actual use of the service or by pre-paid contract. Upgrade entitlement can be extended to 10 years for a one-off charge of 25% of the licence fee.

All new commercial QM licences include activation of the AccuTerm terminal emulator at no extra cost.

To download the commercial QM release 2.8-8 click here.

The **open source version** is currently released only for Linux though users are free to migrate it to other environments. Within the terms of the GPL, OpenQM can be freely used and distributed. For a brief summary of what you are allowed to do under the GPL click here. For guidance on which version is right for you click here.

To visit the open source download page click here. For open source resources click here.

See us at the 2009 Spectrum conference in Denver, 23-26 March.

New!
Read case studies of users who have migrated to QM.

About OpenQM

Sales and Downloads
Current Downloads
Archived Releases
Evaluation licence
What's New
Technical Notes
Resource Library
Newsletters
Pricing

Dealers Area

Help and Support

Privacy Policy

Links

Contact Us

Home

Get QM Today

Obtaining a fully supported copy of QM couldn't be easier.

You can buy the commercial product from a distributor or a dealer. For both routes, the software can be downloaded from this web site or supplied on CD. First line support will be provided by the company from which you purchase the software.

- Distributors sell mostly to other resellers but may be interested in direct sales to end users of the software. We are keen to establish distributors in all major languages and geographic regions.
- Dealers usually sell QM with an application software package of their own. They may also offer QM without any accompanying application. The dealers shown here are willing to sell QM without a companion application. Other dealers may be listed on distributors' websites or can be found by contacting your regional distributor.
- A 30 day evaluation licence is available without charge.
- There is also a free Personal Version for private non-commercial use.

QM is available licensed for simultaneous use by 1 to 500 users as standard. Please contact us if you have an application with a higher user count. All versions also allow a number of additional 'free' phantom processes. This number is eight for a minimal licence and grows by one for every two users beyond the first four.

Because new versions of QM are published frequently, a user licence entitles you to free upgrades for a period of one year from the date of purchase of your licence. This can be extended to ten years for a one-off charge of 25% of the licence fee.

Select the appropriate supplier from the options above. If you would like to try QM before you buy, select evaluation licences.

To subscribe to the QM Newsletter or to receive notification of new releases by email, visit the Email Subscription page.

Displaying Menus

Menu templates are stored in the MENUS file.

Each entry contains:

- Displayed text
- Target item type (menu, HTML, program, URL)
- Target identity
- Access filter
- Action on expand
- Action on collapse

Displaying Menus

```
function display.menu
$include common.h

   menu = '<br><div align="right"><font size="2">'

   gosub show(1, ses.rec<S.AREA>)
   menu := '</font></div>'

   return menu
```

Displaying Menus

```
local subroutine show(depth, mnu.id)
   private mnu.rec, num.items, mnu.idx, text, type, action, filter

   read mnu.rec from mnu.f, upcase(mnu.id) then
      num.items = dcount(mnu.rec<M.TEXT>, @vm)
      for mnu.idx = 1 to num.items
         text = change(mnu.rec<M.TEXT, mnu.idx>, ' ', ' ')
         type = mnu.rec<M.TYPE, mnu.idx>
         action = mnu.rec<M.LINK, mnu.idx>
         filter = mnu.rec<M.FILTER, mnu.idx>

         if filter = '' or index(filter, ses.rec<S.LEVEL>, 1) then
            * ----- Menu items construction goes here -----
         end
      next mnu.idx
   end
   return
end
```

Displaying Menus

* Menu items

```
begin case
   case type = 'H'   ;* HTML document
      menu := '<a href="':link('h','t1=':action):'">'
      menu := if depth = 1 then '<b>':text:'</b>' else text
      menu := '</a><br>'

   case type = 'M'   ;* Menu
      menu := '<a href="':link('m','t1=':action):'">'
      menu := if depth = 1 then '<b>':text:'</b>' else text
      menu := '</a><br>'
      locate upcase(action) in ses.rec<S.MENUS,1> setting pos then
         gosub show(depth + 1, action)
      end
```

Displaying Menus

```
  case type = 'P'   ;* Program
     menu := '<a href="':link(action):'">'
     menu := if depth = 1 then '<b>':text:'</b>' else text
     menu := '</a><br>'

   case type = 'U'   ;* URL
     menu := '<a href="http://':action:'">'
     menu := if depth = 1 then '<b>':text:'</b>' else text
     menu := '</a><br>'

   case 1
     menu := text : '<br>'
end case
```

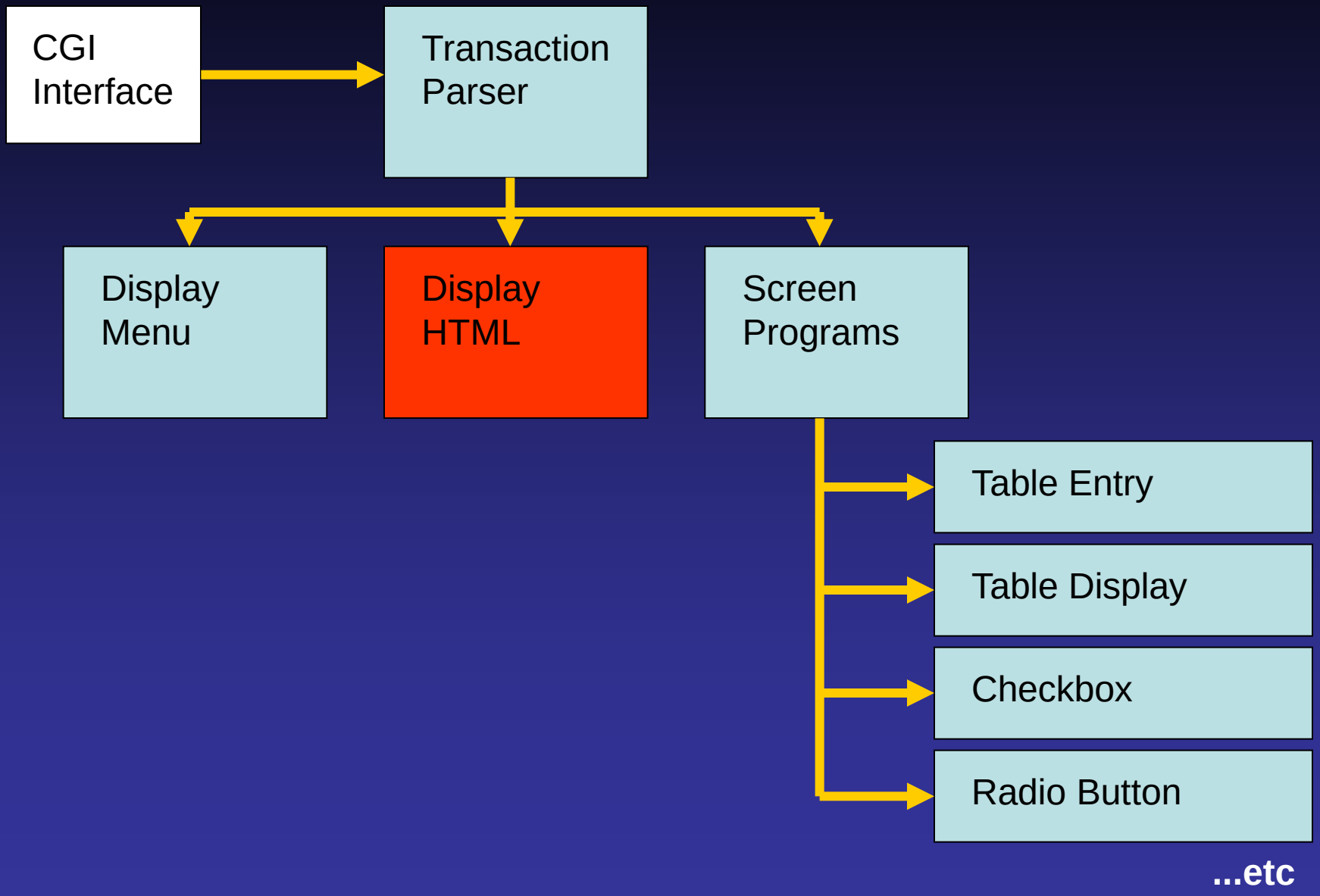
Displaying Menus – Link Generation

```
function link(screen.name, arg1, arg2, arg3, arg4, arg5, arg6) var.args
$include common.h

   s = '?T0=':screen.name:'&X=':session.id

   if assigned(arg1) then s := '&' : arg1
   if assigned(arg2) then s := '&' : arg2
   if assigned(arg3) then s := '&' : arg3
   if assigned(arg4) then s := '&' : arg4
   if assigned(arg5) then s := '&' : arg5
   if assigned(arg6) then s := '&' : arg6

   return (s)
end
```


CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc

Displaying HTML Pages

Some pages are pre-stored HTML.

May be a whole page or just some part of a page.

Stored "pages" may be nested to any depth.

This can contain special tokens to insert variable data into the page.

Also supports conditional inclusion of parts of the page.

SHOW.HTML(page, args)

HTML Page Insertion Tokens

Enclosed in <<...>> brackets.

<<CGI.LINK>>	The CGI program URL
<<HTML.xxx>>	Insert HTML item xxx
<<SESSION.ID>>	Insert session id
<<TKN.xxx>>	Insert text from record xxx
<<name>>	Insert named variable
<<n>>	Insert argument n

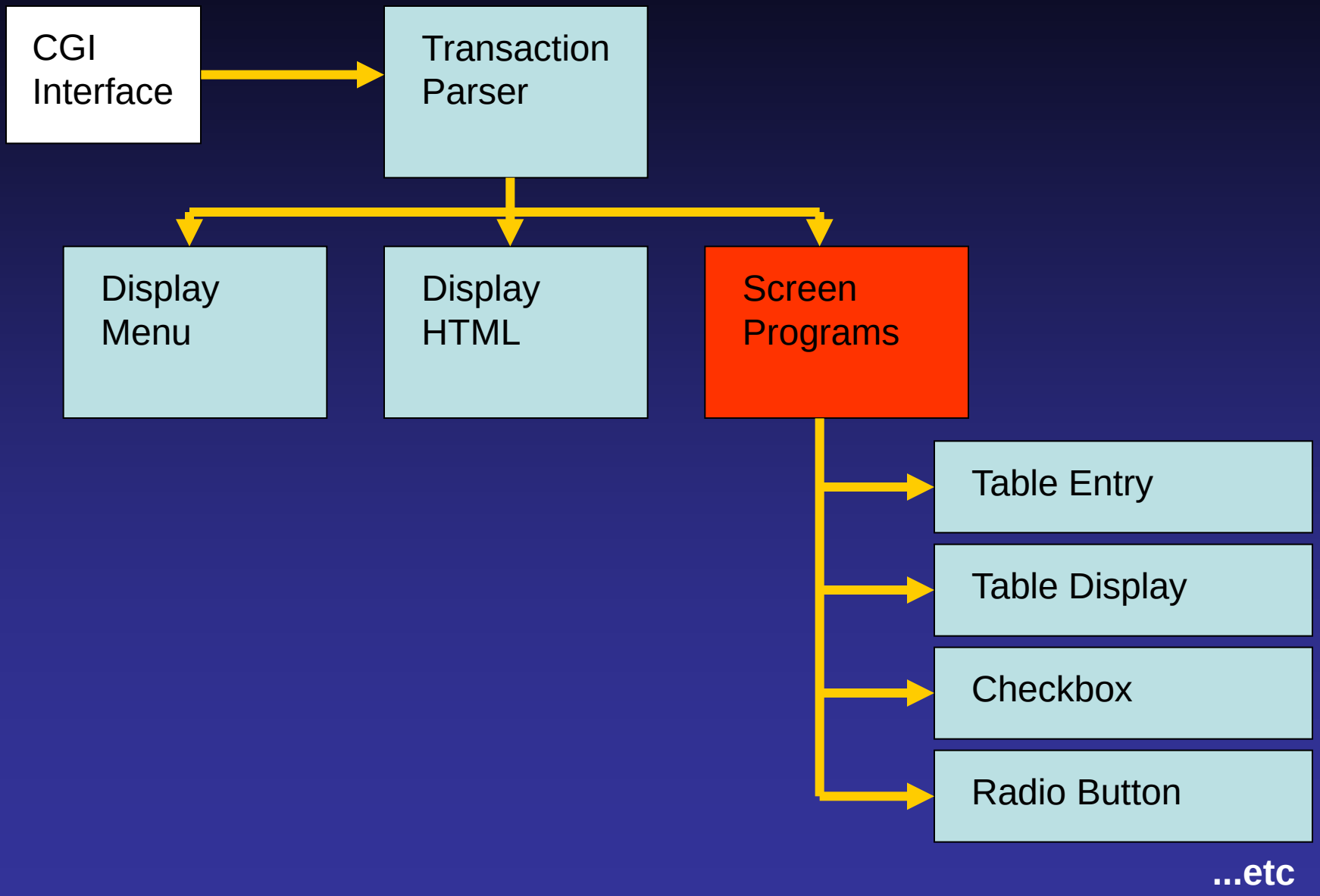
HTML Page Named Variables

!name value

Allows one stored page element to set data to be used in another nested element.

HTML Page Conditional Inclusion

?IS level	Only if user has this access
?IS.NOT level	User does not have this access
?	Unconditional

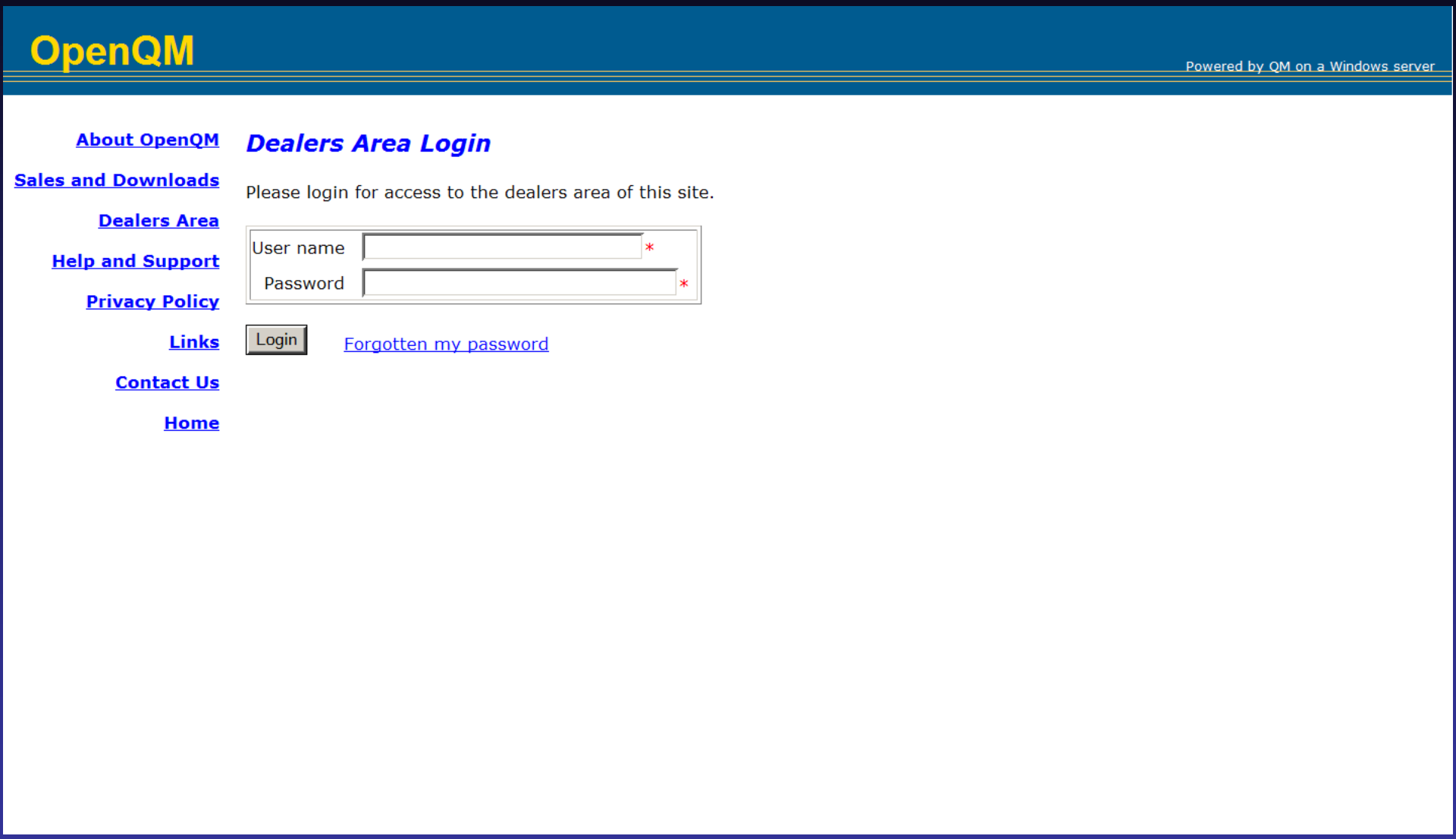

CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc

Screen Programs

These handle the main interactive pages.

The program name is formed from a fixed prefix followed by the alphabetic part of the T0 parameter.

Any numeric part forms the "phase", defaulting to 1.


OpenQM
Powered by QM on a Windows server
About OpenQM
Sales and Downloads
Dealers Area
Help and Support
Privacy Policy
Links
Contact Us
Home
Dealers Area Login
Please login for access to the dealers area of this site.
User name
Password
Login
Forgotten my password

Screen Programs

```
program s.login
$include common.h
   begin case
      case phase = 1
         gosub display.form

      case phase = 2
         begin case
            case b<1>   ; gosub logon.user
            case b<2>   ; gosub forgotten.password
         end case

      case phase = 3
         gosub forgotten.password

      case phase = 4
         gosub request.password
   end case
   return
```

Screen Programs

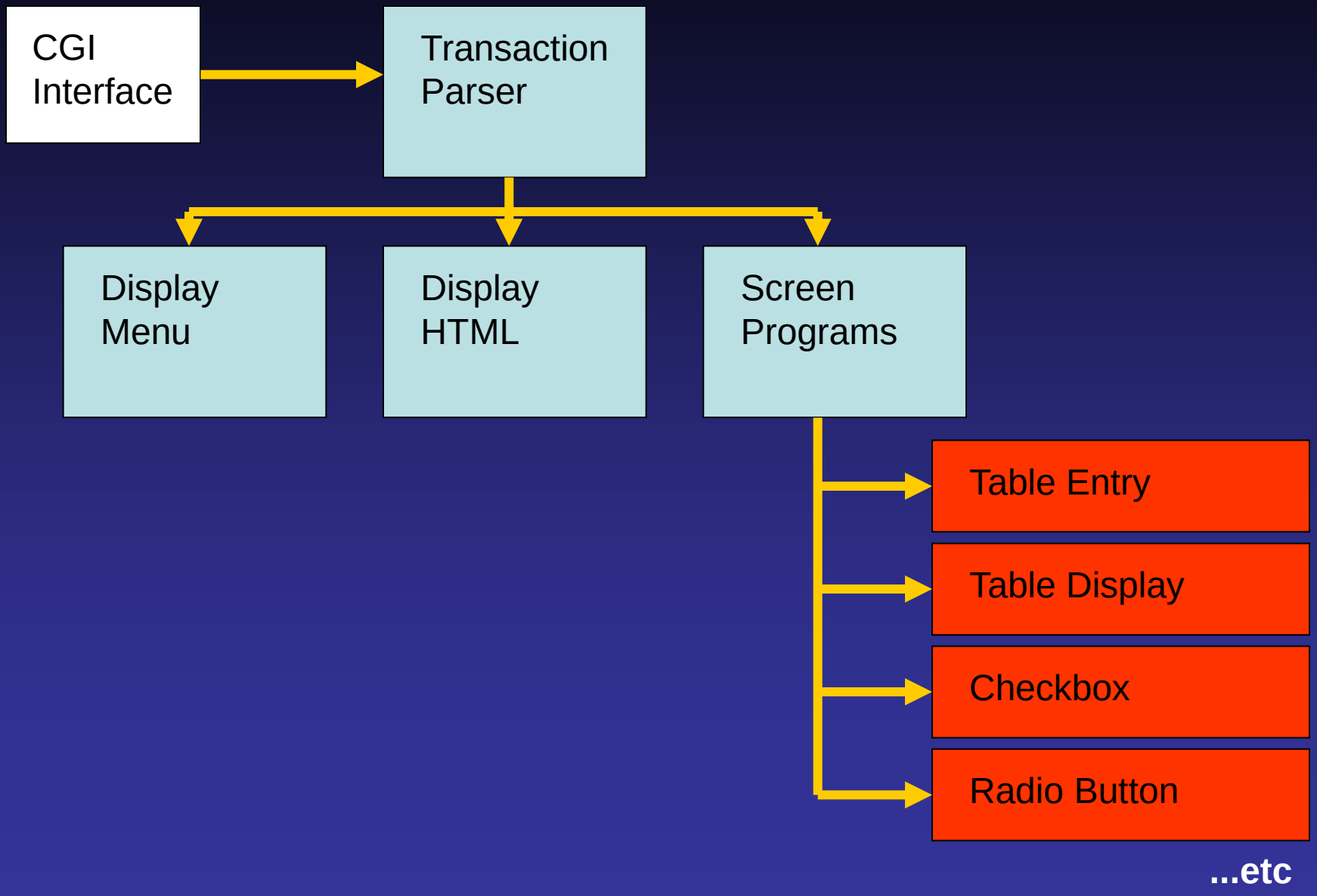
```
display.form:
   body = title('Dealers Area Login')
   body := form('login2')

   body := '<p>Please login for access to the dealers area of this site.</p>'
   body := '<table rules="none">'
   body := table.entry(1, 'User name', @true)
   body := table.entry(2, 'Password||password', @true)
   body := '</table><br>'

   body := button(1, 'Login')
   body := '      '
   body := '<a href="':link('LOGIN3'):'">Forgotten my password</a>'

   body := '</form>'
   body := set.focus('T1')

   return
```


CGI Interface
Transaction Parser
Display Menu
Display HTML
Screen Programs
Table Entry
Table Display
Checkbox
Radio Button
...etc

Screen Programs

```
function title(text)
$include common.h

   return '<h1>' : text : '</h1>'
end
```

Screen Programs

```
function form(link)
$include common.h

   s = '<form method="POST" name="form" action="':cgi.link:'">'
   s := '<input type="hidden" name="X" value="':session.id:'"/>'
   s := '<input type="hidden" name="T0" value="':link:'"/>'

   return s
end
```

Screen Programs

```
function table.entry(idx, text, mandatory)
$include common.h
   left.text = field(text, '|', 1)
   width = field(text, '|', 2) ; if width = '' then width = 35

   s = '<tr><td align="right">':left.text:' </td>'
   s := '<td align="left"> '
   s := '<input type="text" name="T':idx:'"'
   if t.err<idx> then s := ' style="background: lightsalmon"'
   s := ' size="':width:'" value="':t<idx>:'"/>'
   if mandatory then s := '<font color="#FF0000">*</font>'
   s := '</td></tr>'

   return (s)
end
```

Screen Programs

```
function button(idx, text, disable) var.args
   s = '<input type="submit" value="':text:'" name="B':idx:""

   if assigned(disable) then
      if disable then s := ' onclick="this.disabled=true;form.submit();"'
   end

   s := '/>'

   return s
end
```

Screen Programs

```
function checkbox(idx, checked)
$include common.h

   s = '<input type="checkbox" name="C':idx:'" value="C':idx:""
   if checked then s := ' checked="checked"'
   if c.err<idx> then s := ' style="background: lightsalmon"'
   s := '/>'

   return s
end
```

Screen Programs

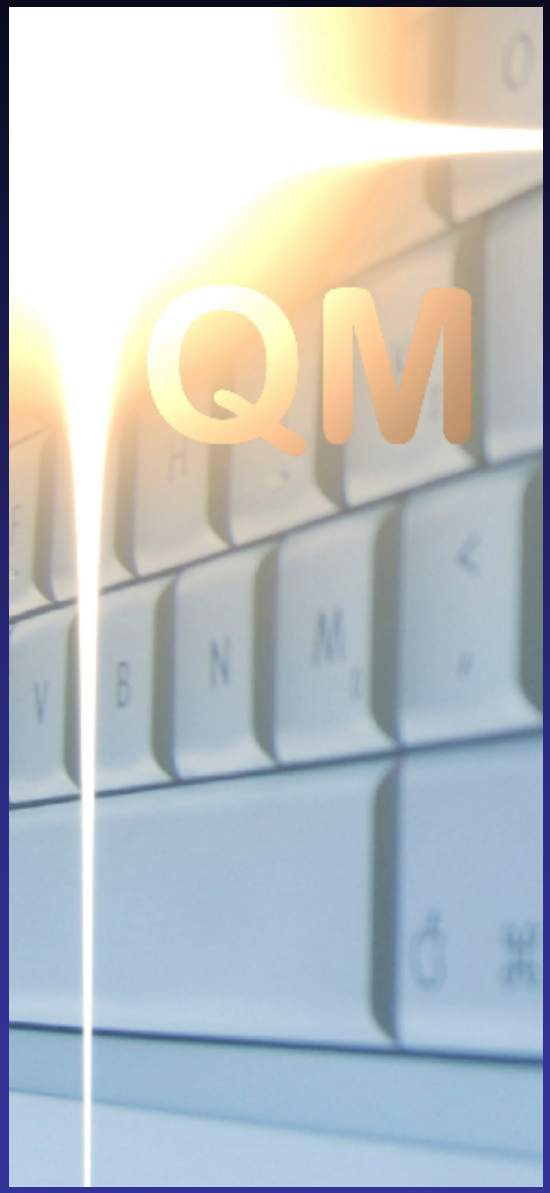
```
function set.focus(name)
$include common.h

   s = '<script type="text/javascript">'
   s := 'document.form.':name:'.focus();'
   s := '</script>'

   return s
end
```

OpenQM

QUESTIONS?



OpenQM