+

Knockout for APEX

Dick Dral

Agenda

- Introduction
- Knockout
- Knockout and APEX
  - several demo’s
- Conclusion

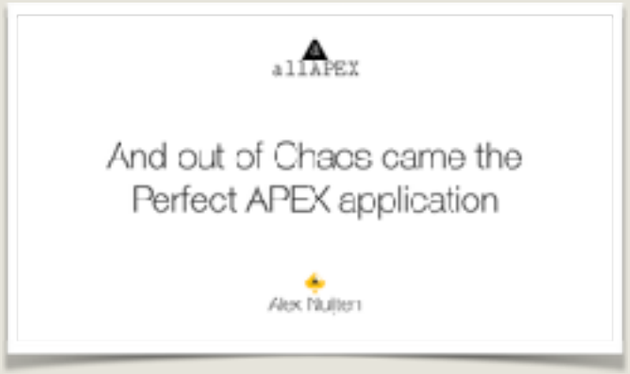
About me

- Dick Dral
- Almost 30 years of Oracle
- 10 years self employed
- Presentations
- Blogposts: dickdral.blogspot.com
- Demo application:
  www.speech2form.com/ords/f?p=OPFG
- Twitter: @dickdral

ORACLE ACE Associate

smart4 apex®

14:30

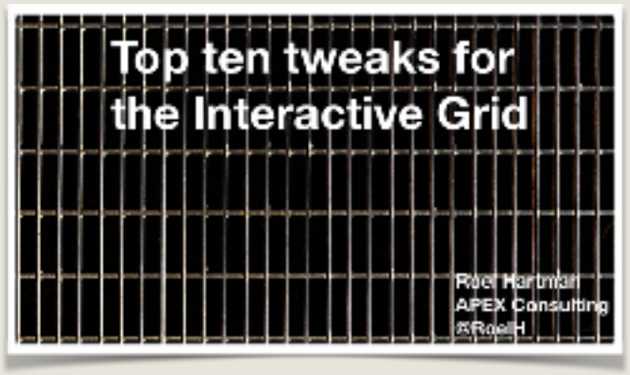


smart4

apex

14:30

13:30

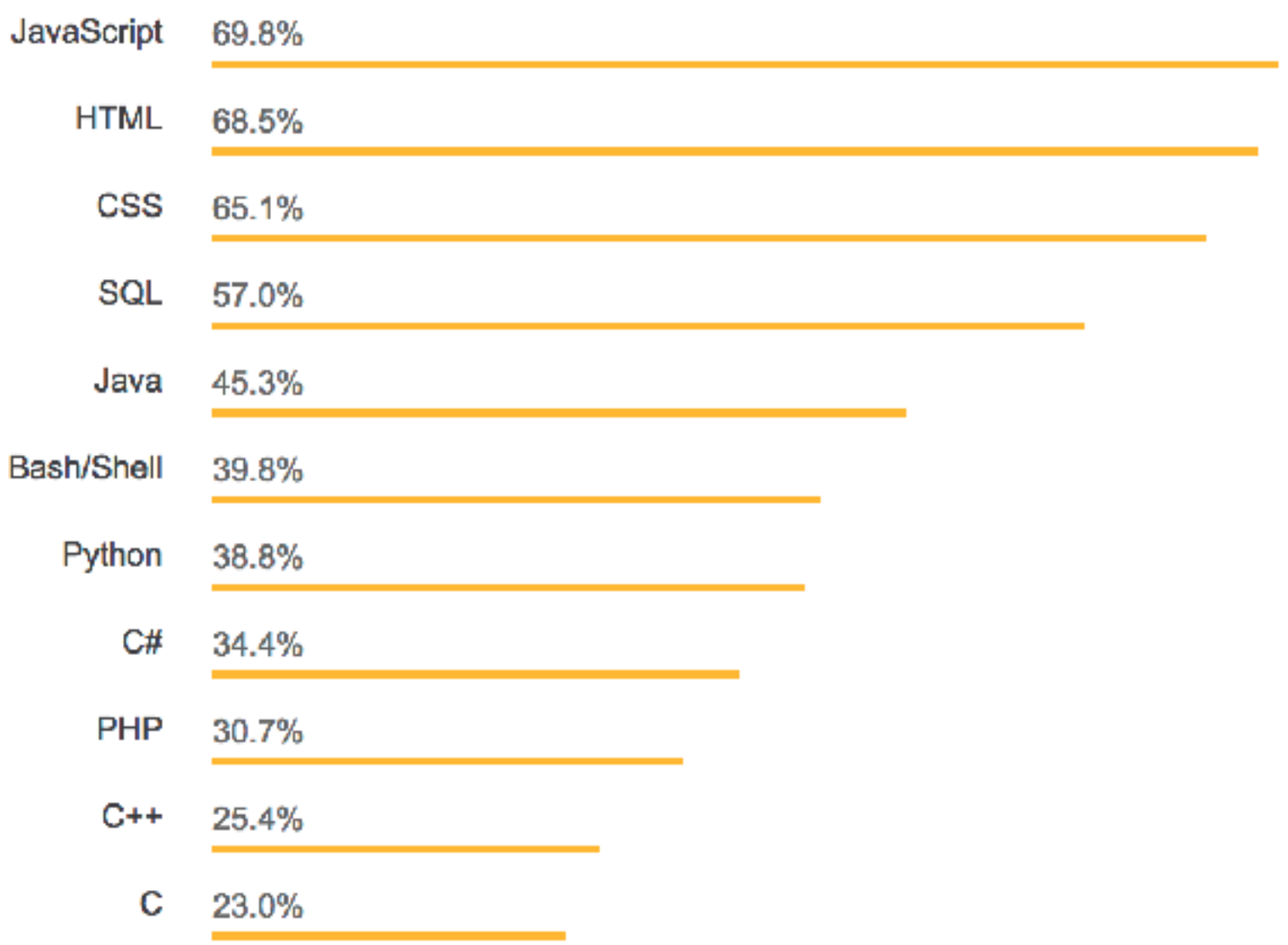


13:30

Warning!

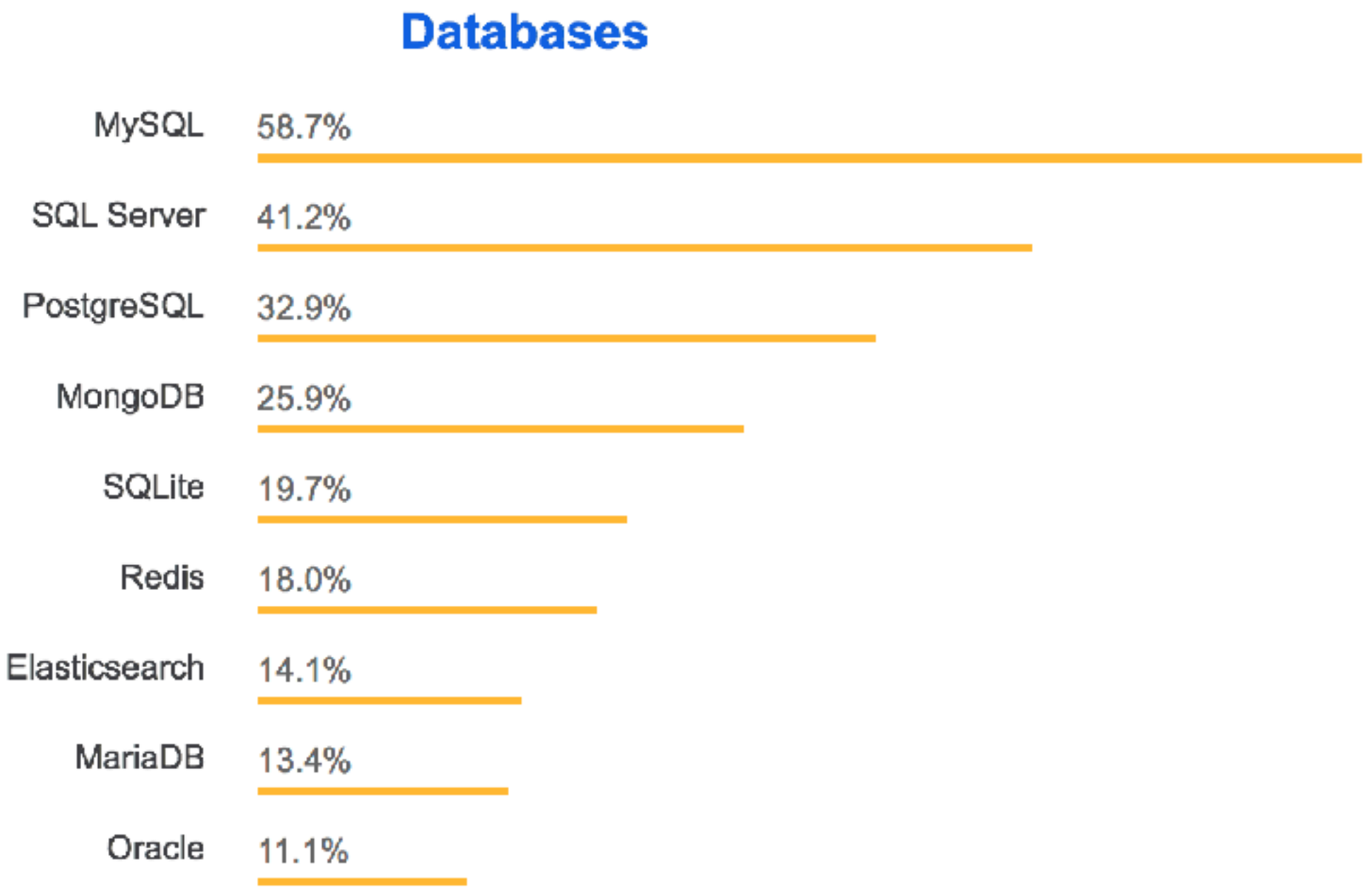
JavaScript in IT

Most Popular Technologies



Source: 2018 Stack Overflow Developer Survey

Databases in IT



Source: 2018 Stack Overflow Developer Survey

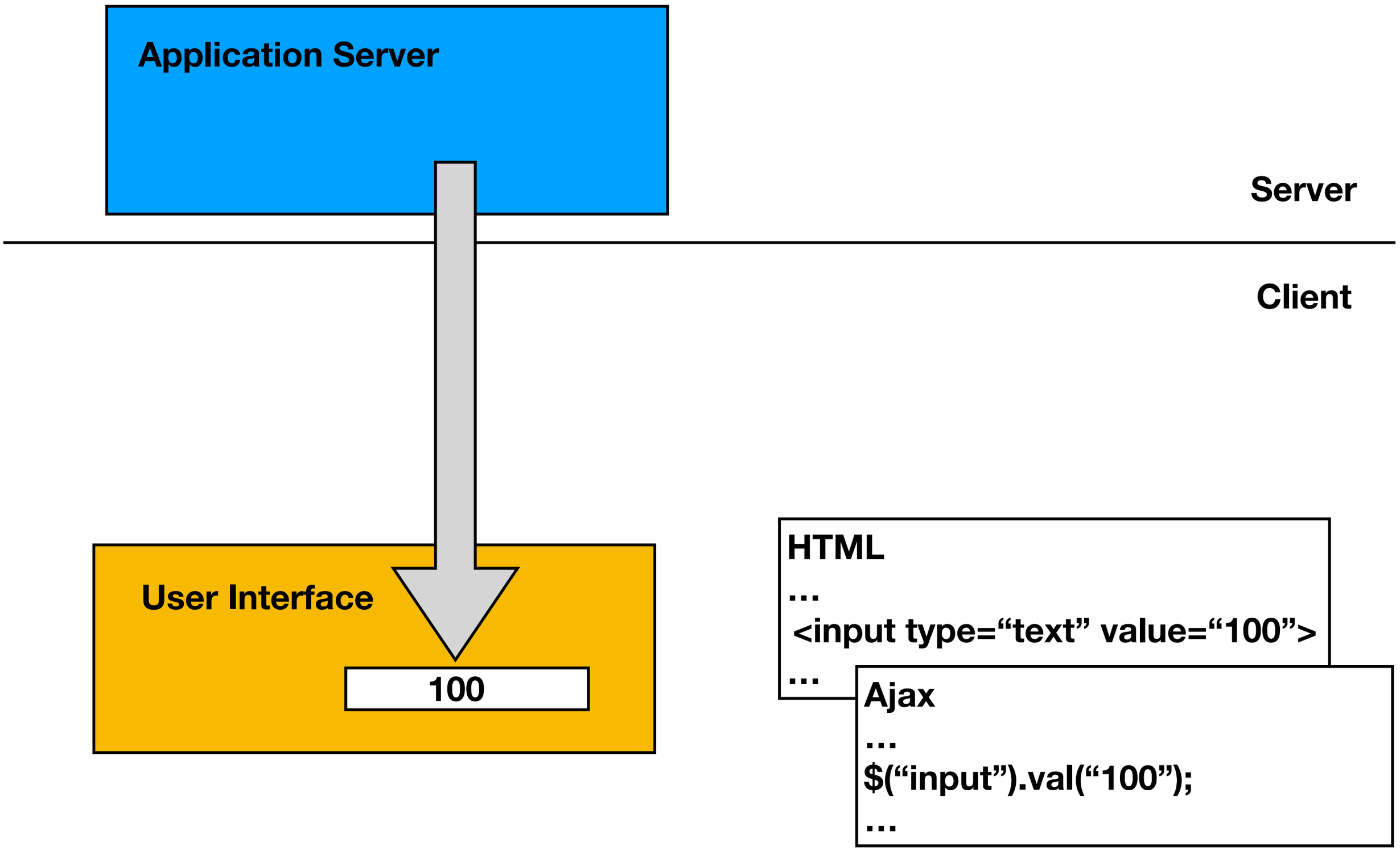
JavaScript in APEX

APEX V2.2	**260 Kb**
APEX V3.2	**2,8 Mb**
APEX V4.2	**27 Mb**
APEX V5.0	**47 Mb**
APEX V5.1	**80 Mb**

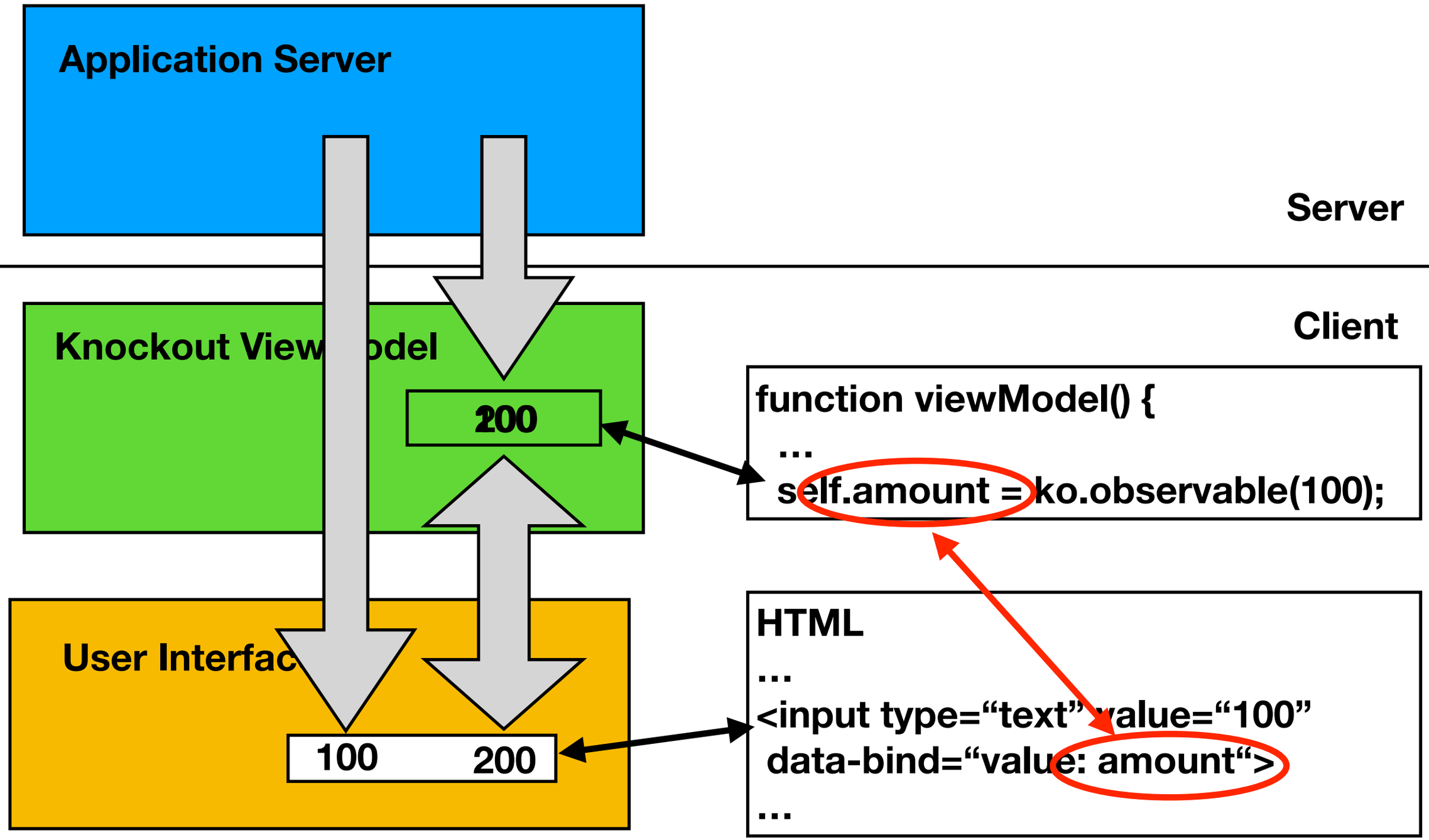
KnockoutJS

- Written by Steve Sanderson, Microsoft employee
- Open source
- Client side behavior
- Two way binding
- Dependency tracking
- Templating

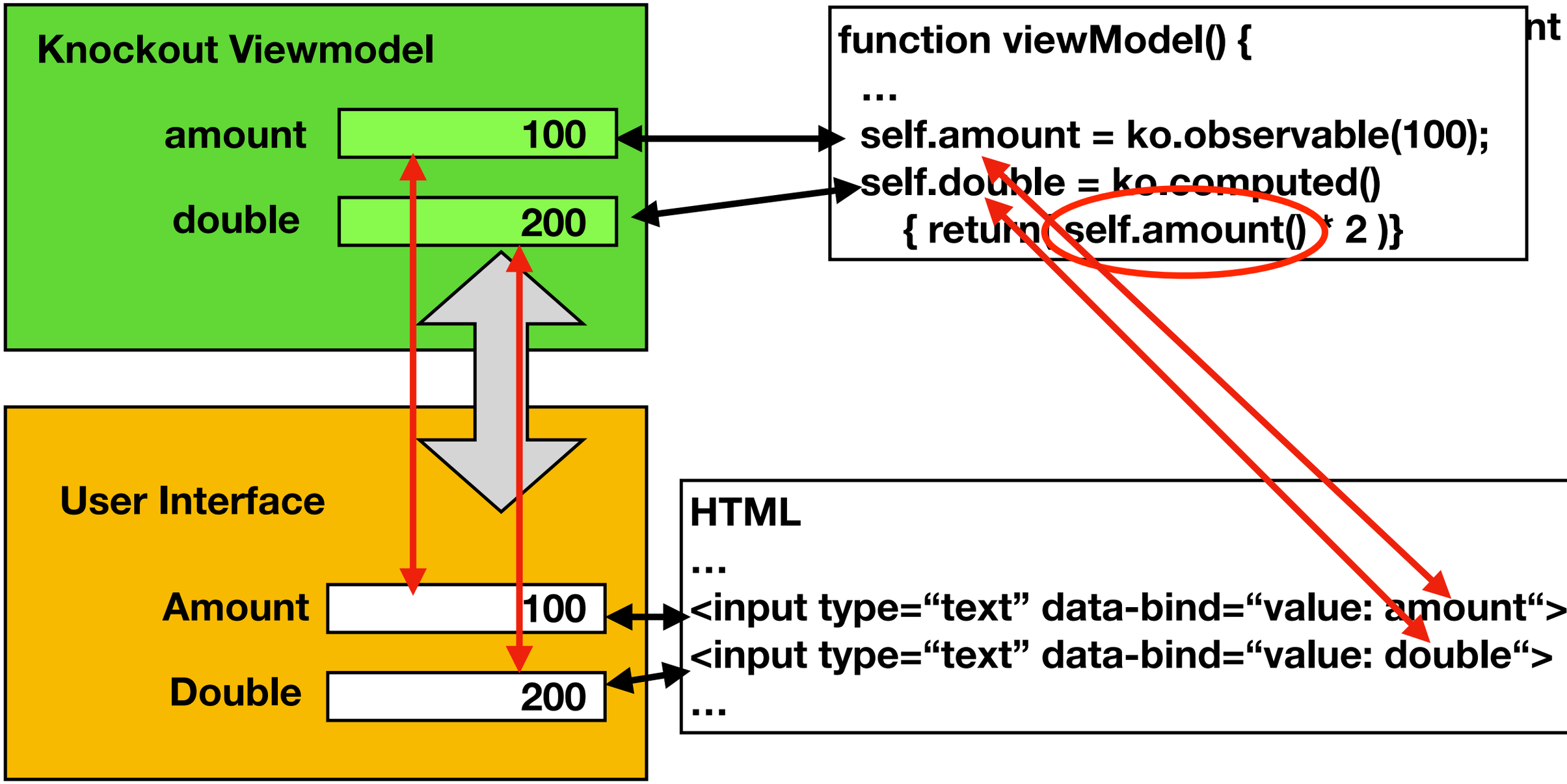
2-way binding



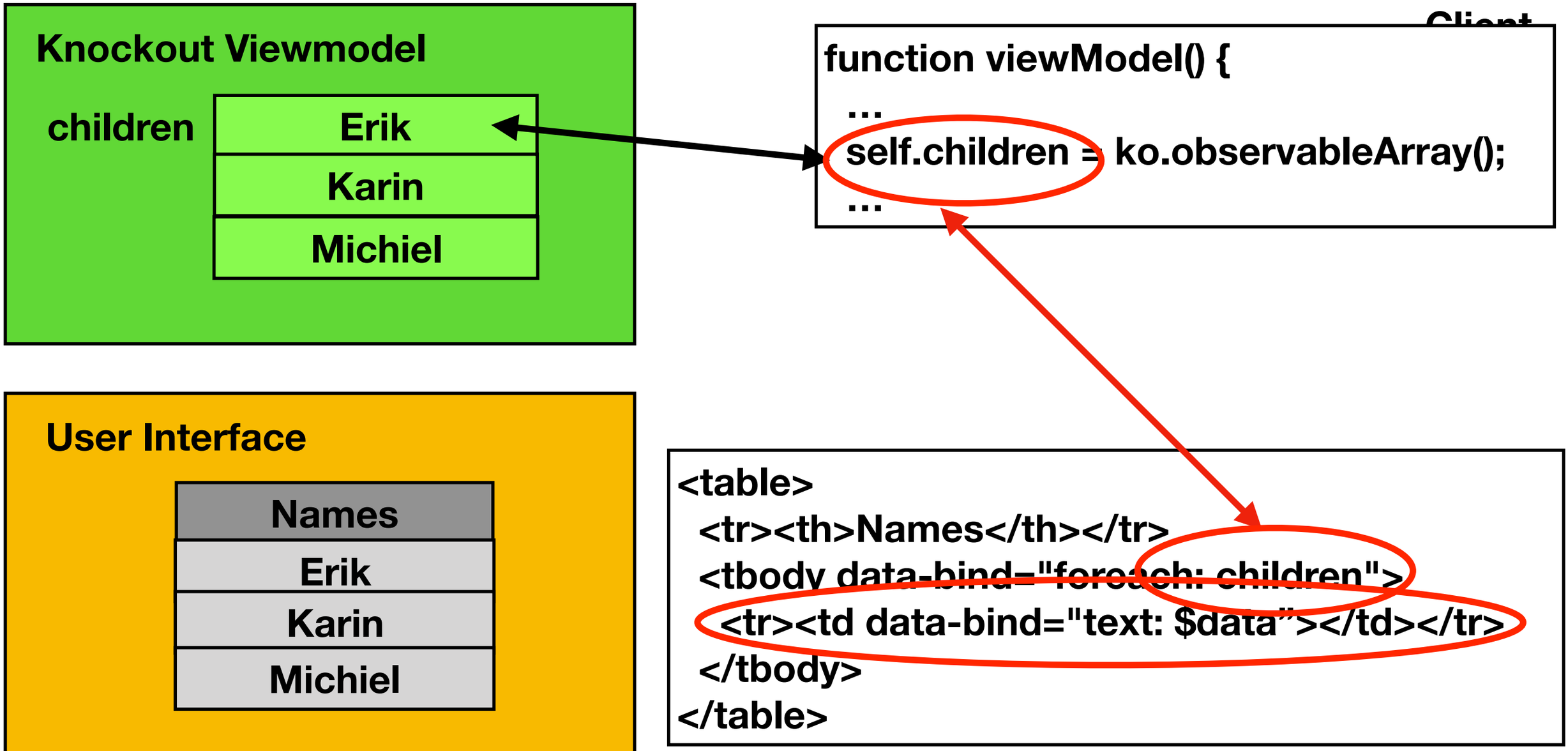
2-way binding



Dependency tracking



Templating


Knockout Viewmodel
children
Erik
Karin
Michiel
Client
function viewModel() {
...
self.children = ko.observableArray();
...
User Interface
Names
Erik
Karin
Michiel
<table>
<tr><th>Names</th></tr>
<tbody data-bind="foreach: children">
<tr><td data-bind="text: $data"></td></tr>
</tbody>
</table>

Knockout - other features

- data-bind: “click” => perform viewModel function
- data-bind=“visible:” => show/hide element based viewModel variable
- data-bind=“enable:” => enable/disable element based viewModel variable
- data-bind=“valueUpdate: “afterkeydown” => update viewModel variable after each key pressed

APEX & Knockout

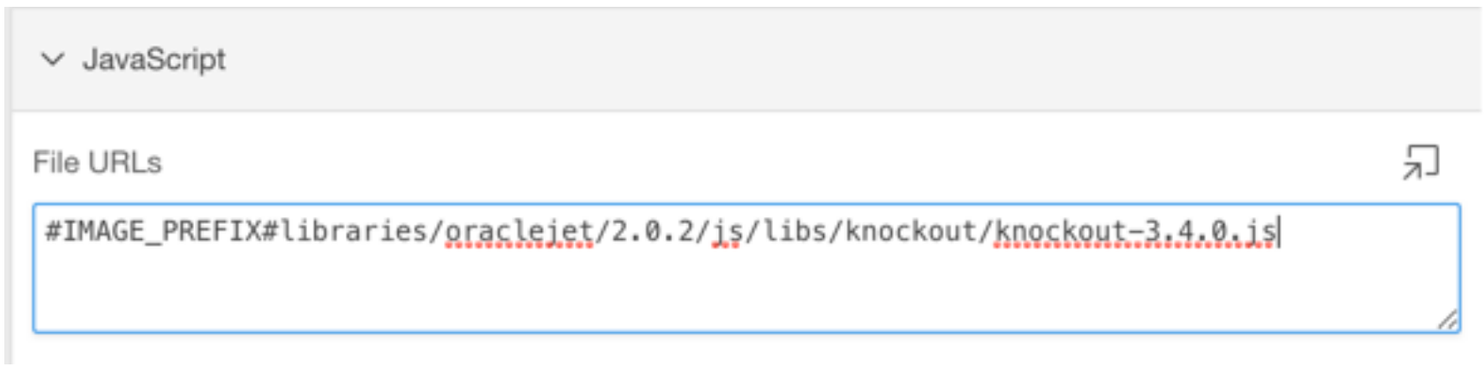
- Load JS library
- Define viewModel
- Define data-binds in HTML
- Define template in HTML

APEX & KO : JS Library

- Use library from APEX files:
  #IMAGE_PREFIX#libraries/oraclejet/2.0.2/js/libs/knockout/knockout-3.4.0.js

- Or download latest version

- Register on page



- Or for whole application

  - shared components > user interface attributes > desktop > JavaScript

Simple addition

Simple addition: view model

View Model

```
function addition() {
  var self = this;
  self.number1 = ko.observable(0);
  self.number2 = ko.observable(0);

  self.total = ko.computed(function() {
   return (get_number(self.number1()) + get_number(self.number2()));
  }, self);
}
```

Apply View Model

```
var viewModel = new addition();
ko.applyBindings(viewModel);
```

Simple addition: bindings

Edit item properties:

inputs:

output:

Client side error checking

Client side error checking

View Model

```
function addition() {
  var self = this;
  self.number1 = ko.observable(0);
  …
  self.number1_error = ko.computed( function() {
      var number = parseInt(nvl(self.number1(),0));
      var message = ( number < 0) ? " Number 1 may not be negative":"";
      return(message);
  }, self);
  …
}
```

Post Item Text

```
<span data-bind="text: number1_error" class="error"></span>
```

HTML Editor

Generate log statements

Generate log statements: view model

```
...
   self.output = ko.computed(function() {
                  var code = generate2( self.variables(),self.language(), self.log_type(),
                                                  self.indent(), self.separate() );
                  return ( code );
               });
  self.inputClicked = function() {
      var identifier = get_identifier();
      if ( identifier != undefined && identifier.length > 0 )
           { self.variables.push({ identifier: identifier}); }
   }
   self.removeVariable = function(element) {
      self.variables.remove(element);
   }
  self.removeAllVariables = function(variable) {
       self.variables.removeAll();
   }
...
```

Generate log statements: view model

```
...
   self.output = ko.computed(function() {
                  var code = generate2( self.variables(),self.language(), self.log_type(),
                                                  self.indent(), self.separate() );
                  return ( code );
               });
  self.inputClicked = function() {
      var identifier = get_identifier();
      if ( identifier != undefined && identifier.length > 0 )
           { self.variables.push({ identifier: identifier}); }
   }
   self.removeVariable = function(element) {
      self.variables.remove(element);
   }
  self.removeAllVariables = function(variable) {
       self.variables.removeAll();
   }
...
```

Generate log statements: view model

```
...
   self.output = ko.computed(function() {
                  var code = generate2( self.variables(),self.language(), self.log_type(),
                                                  self.indent(), self.separate() );
                  return ( code );
               });
  self.inputClicked = function() {
     var identifier = get_identifier();
     if ( identifier != undefined && identifier.length > 0 )
          { self.variables.push({ identifier: identifier}); }
   }
    self.removeVariable = function(elem
     self.variables.remove(element);
   }
  self.removeAllVariables = function(va
     self.variables.removeAll();
   }
...
```

P30_INPUT data binding:

data-bind="value:input, click:inputClicked"

Generate log statements: view model

```
...
    self.output = ko.computed(function() {
                    var code = generate2( self.variables(),self.language(), self.log_type(),

                    retur
                });
    self.inputClicked
        var identifier =
        if ( identifier !=
            { self.variables.push({ identifier: identifier}); }
    }
    self.removeVariable = function(element) {
        self.variables.remove(element);
    }
    self.removeAllVariables = function(variable) {
        self.variables.removeAll();
    }
...
```

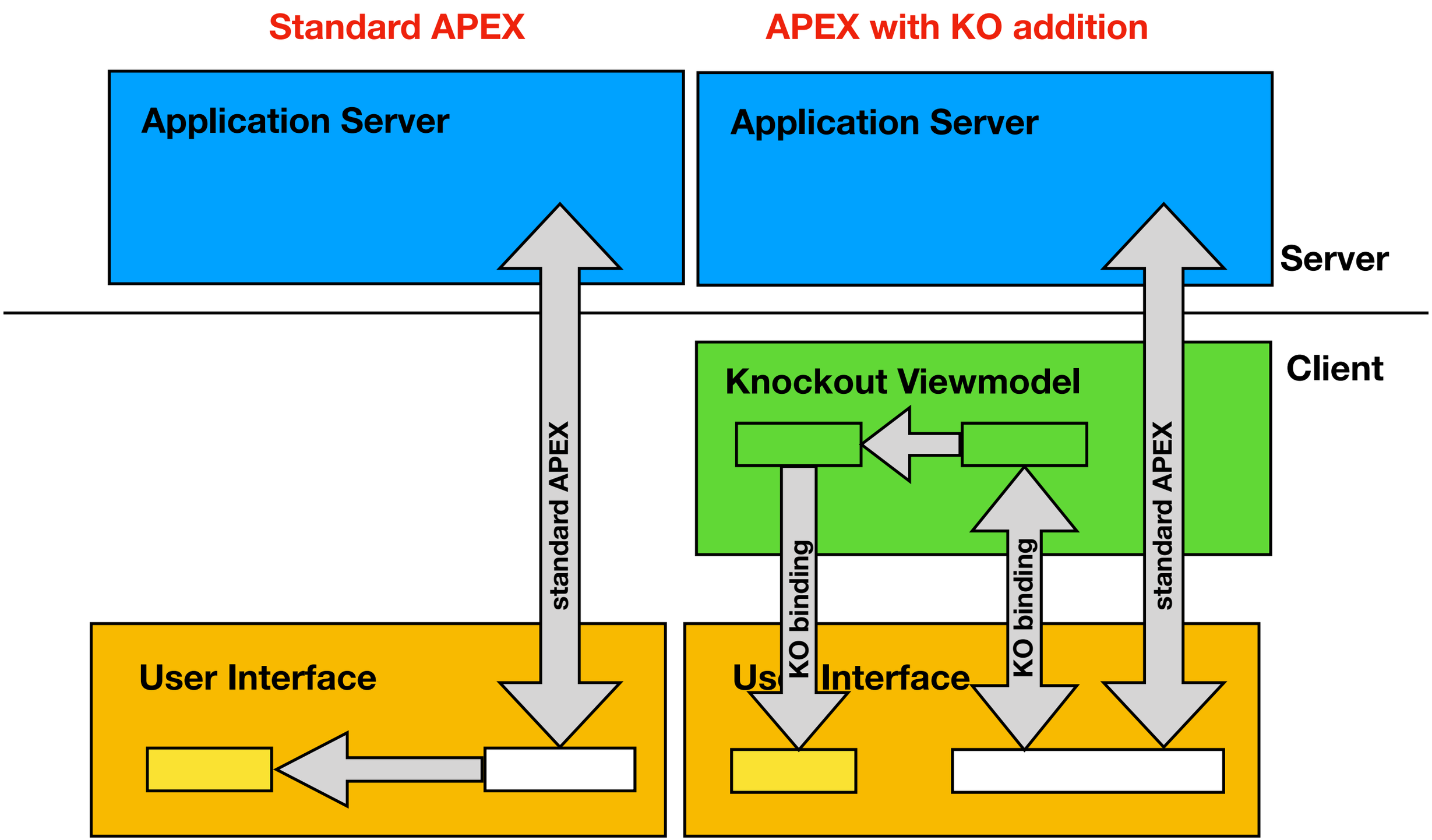
```
<tbody data-bind="foreach: variables”>
  <tr><td data-bind="text: identifier"></td>
    <td data-bind="click: $parent.removeVariable”>
      <i class="fa fa-trash report-icon" ></i>
    </td>
  </tr>
```

Analyse / Debug

- Inspect view model as JS variable
  - Assign view model to global variable **vm**
- Set values with JS functions
  - **vm.number1(10)**
- Use Knockout Context Debugger
  - Chrome add-in to inspect view model

Multi Row Data

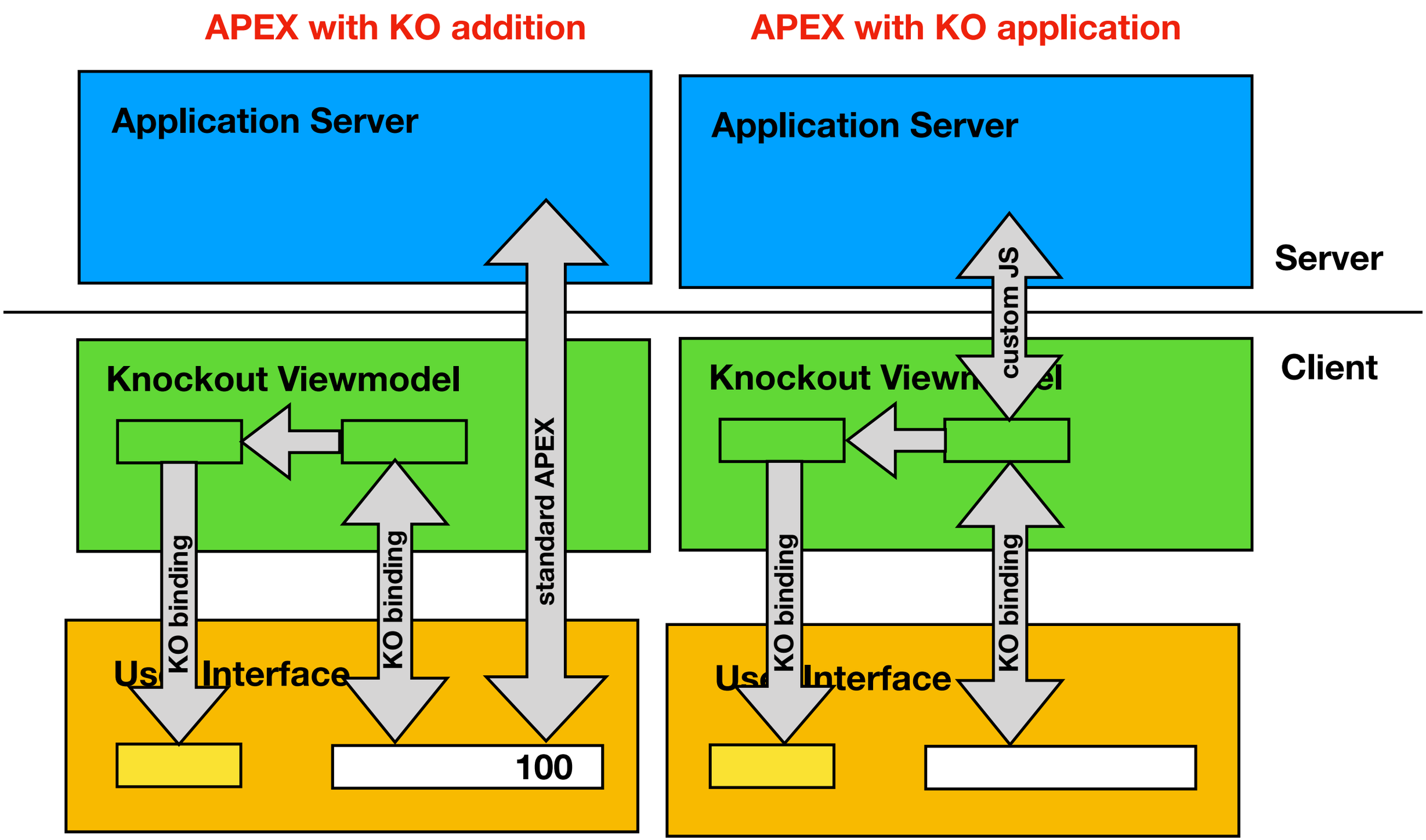
Multi Row Page



Multi Row Data

- Fetch data with APEX Classic report / Tabular Form
- Bind the fetched data to Knockout
- Use Knockout pre-rendered to bind existing data
- Nested view models
- Jorge Rimblas example application
- Fetch product information using Ajax
- Add row functionality

Multi Row Page



Conclusion

- Knockout can be used for client side behavior
- Easy coding using depencies
- Easy debugging with JS view model and Chrome add-in
- APEX Multi line pages with Knockout are not trivial
- Best solution for multi line pages should be decided for each use case

Info Sources

- www.knockoutjs.com
- https://github.com/ErikSchierboom/knockout-pre-rendered
- https://github.com/rimblas/apex-ko-presentation
- https://github.com/dickdral/apex-knockout-demo
- Google / Stack Overflow