ZK Training

MODULE 1-1

INTRODUCTION

Table of Contents

Skim ZK

- What is ZK
  - What is not ZK
- Markup Language UI Programming
- Rich User Interface
- Strong MVC Design Pattern
- CSS UI Customization
- Eclipse IDE Support

What is ZK

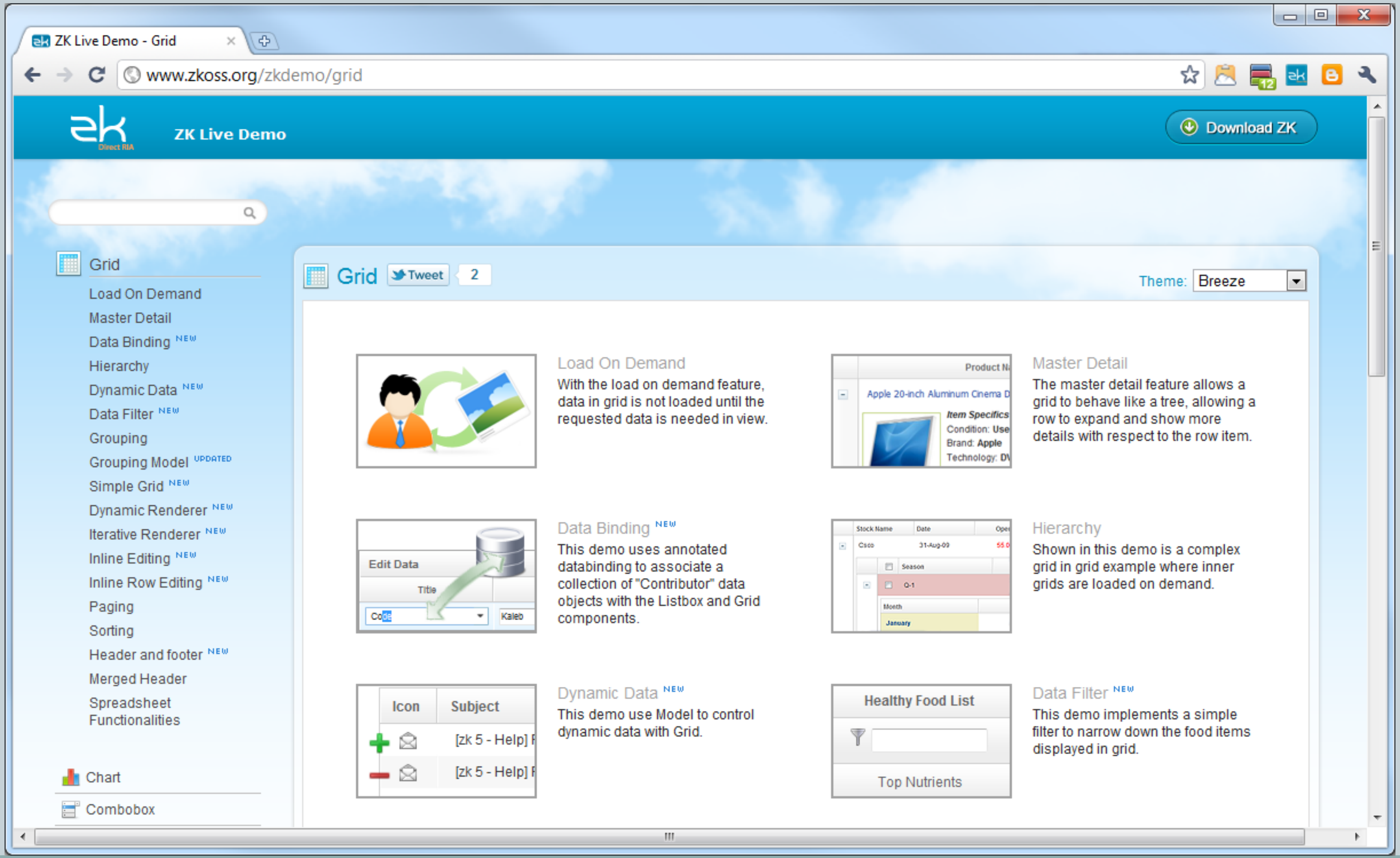
- An AJAX framework based on JAVA
- XML UI Programming
- Pure Java Programming support
- Allow Fast Prototyping
- Strong MVC Support
- Eclipse, Netbeans IDE Support
- XHTML Compatible & UI Design

What is not ZK

- No assumption to Persistence
- No assumption to middleware
- Does as thin as possible in controller layer
- Not limited in XUL
- Not limited in Browser

Markup Language UI Programming

http://www.zkoss.org/zkdemo/



Load On Demand
With the load on demand feature, data in grid is not loaded until the requested data is needed in view.

Master Detail
The master detail feature allows a grid to behave like a tree, allowing a row to expand and show more details with respect to the row item.

Data Binding NEW
This demo uses annotated databinding to associate a collection of "Contributor" data objects with the Listbox and Grid components.

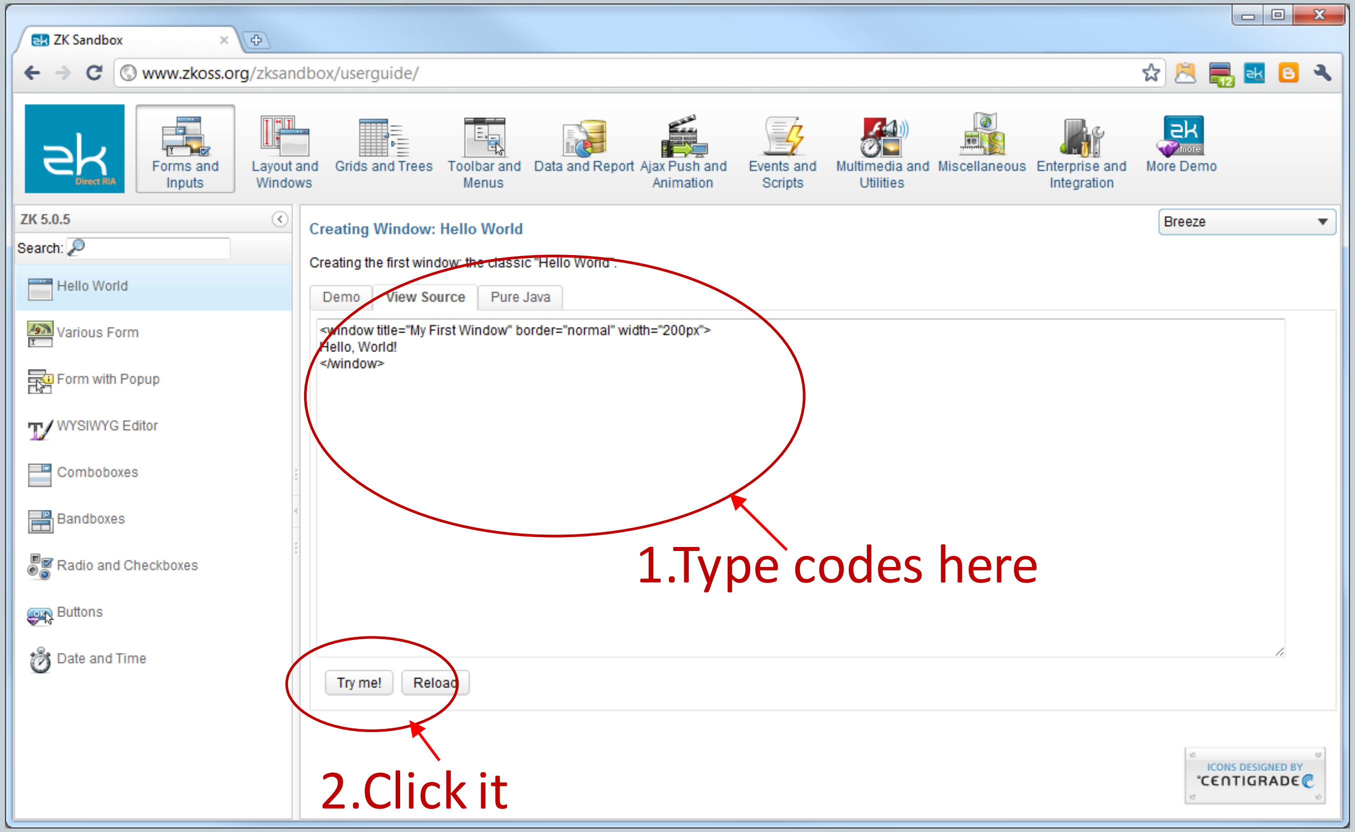
Hierarchy
Shown in this demo is a complex grid in grid example where inner grids are loaded on demand.

Dynamic Data NEW
This demo use Model to control dynamic data with Grid.

Data Filter NEW
This demo implements a simple filter to narrow down the food items displayed in grid.

Try ZUL Online

http://zkfiddle.org/
http://www.zkoss.org/zksandbox/



Creating Window: Hello World

Creating the first window: the classic "Hello World".

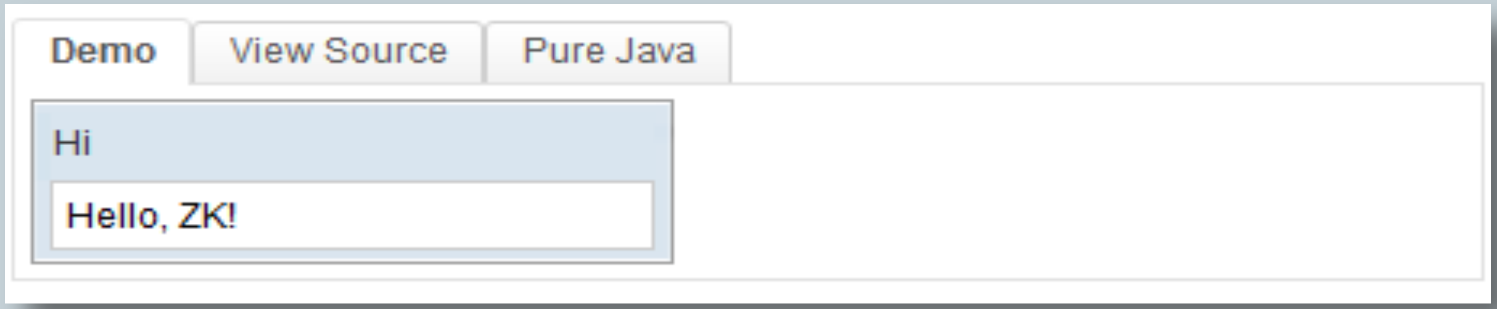
```
<window title="My First Window" border="normal" width="200px">
Hello, World!
</window>
```

Say Hello to ZK

- Type codes

```
<window title="Hi" border="normal" width="200px">
    Hello, ZK!
</window>
```

- Click "Try Me" Button



Hi

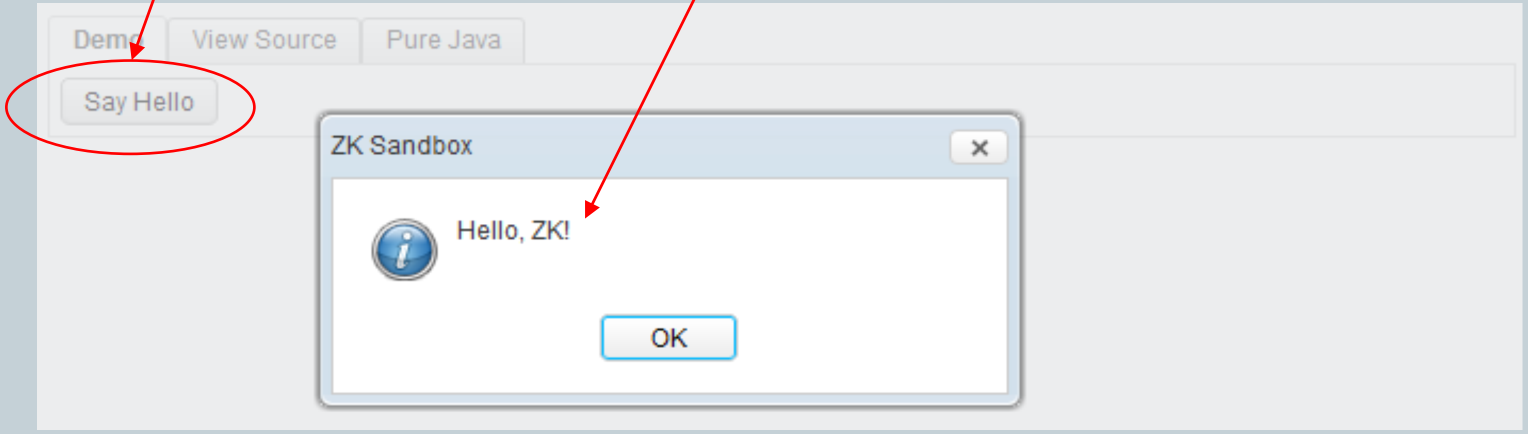
Hello, ZK!

Say Hello on demand

- Type codes

```
<button label="Say Hello"
    onClick='alert("Hello, ZK!")'/>
```

- Click "Try Me" Button



Hello, ZK!

Rich User Interface

- 200+ off-the-shelf state-of-art Ajax components
- Numerous third party widgets:
  - jQuery Plugin, JFreeChart, JasperReports, Google Maps, CKeditor …
- CSS-based skin, template-based look and customizable behavior
- Drag-and-drop, animation, context menu, bookmark management, …

Strong MVC Design Pattern

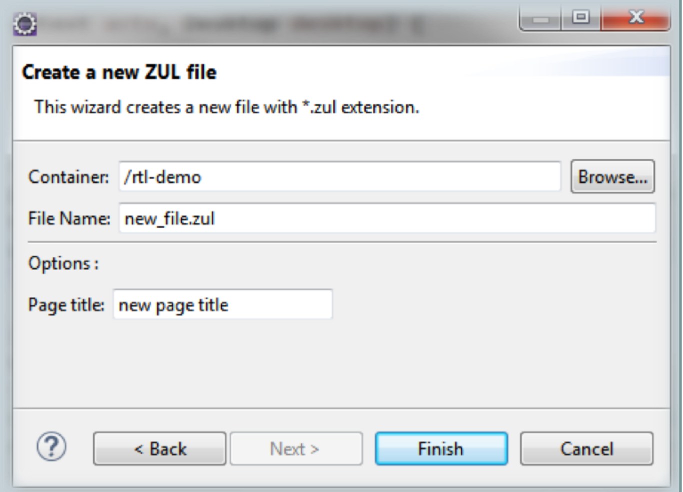
- Model (Your Java Code)
- View (zul file)
- Controller (SelectorComposer)

CSS UI Customization

- Standard CSS based
- Completely style customization guide

Eclipse IDE Support

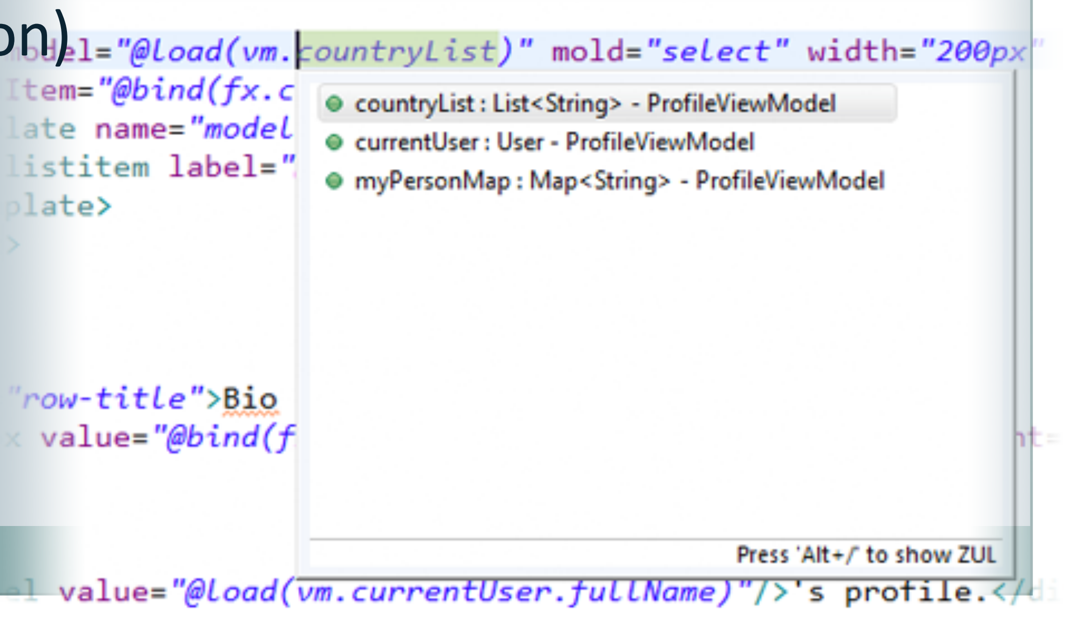
- Quick Installation
  - Eclipse Market Place
- ZK Project
  - Project and page wizards
  - ZK Library embedded
  - ZUL Editor (code completion)
- Maven Archetype
  - http://mavensync.zkoss.org/maven2/



Create a new ZUL file

This wizard creates a new file with *.zul extension.

Container: /rtl-demo
File Name: new_file.zul
Options :
Page title: new page title



```
odel="@load(vm.countryList)" mold="select" width="200px"
Item="@bind(fx.c
late name="model
listitem label="
plate>
>

"row-title">Bio
x value="@bind(f
el value="@load(vm.currentUser.fullName)"/>'s profile.</
```

```
countryList : List<String> - ProfileViewModel
currentUser : User - ProfileViewModel
myPersonMap : Map<String> - ProfileViewModel
```

Press 'Alt+/' to show ZUL

Programming in ZK

- How to program a ZUL file
- EL and Implicit Object
- Event Listener Registration

How to Program a ZUL file - 1

Supported XML instructions

- import
- link, meta & script
- component
- xel-method

How to Program a ZUL file - 2

Special Elements

- <attribute>
- <custom-attributes>
- <template>
- <zk>
- <zscript>

How to Program a ZUL file - 3

Special Attributes

- use="[component class]"
- forward="[target event]"
- fulfill="[target event]"
- apply="[composer class]”
- viewModel=“@id(‘vm’)@init(‘foo.MyViewModel’)”
- If, unless=${el-expression}

EL and Implicit Object

EL

- General Value assign
- Attributes
  - if, unless
  - forEach
  - forEachBegin
  - forEachEnd

Implicit Object

- self
- desktop
- page
- spaceOwner
- session
- execution

Event Listener Registration

- Register Event Listener
  - On page Event Listener
  - Use class Event Listener
  - Dynamically add Event Listener

Register Event Listener

- On Page Event Listener

```
<!--  short attribute -->
<window title="my window" border="normal">
    <button onClick='Messagebox.show("Hello, ZK1!")' label="hi"/>
</window>
```

```
<!--  long attribute -->
<window title="my window" border="normal">
    <button label="hi">
        <attribute name="onClick"><![CDATA[
        Messagebox.show("Hello, ZK1! ");
        ]]></attribute>
    </button>
</window>
```

Register Event Listener

- Use Class Event Listener

```
<!--  index.zul  -->
<window title="my window" border="normal">
    <button label="hi" use="pkg.MyButton"/>
</window>
```

```
//MyBytton.java
package pkg;
public class MyButton extends Button {
    public void onClick(Event event) {
        Messagebox.show("Hello, ZK2! ");
    }
}
```

Register Event Listener

- Use Class Event Listener

```
<zk>
    <zscript><![CDATA[
    public class MyButton extends Button {
        public void onClick(Event event) {
            Messagebox.show("Hello, ZK2!");
        }
    }
    ]]></zscript>
    <window title="my window" border="normal">
        <button label="hi" use="MyButton"/>
    </window>
</zk>
```

Register Event Listener

- Dynamically add Event Listener

```
<window title="my window" border="normal">
    <button label="enable Hi" onClick="enable()"/>
    <button id="btn" label="hi" />
    <zscript><![CDATA[
    void enable() {
        btn.addEventListener("onClick", new EventListener() {
            public void onEvent(Event evt) {
                Messagebox.show("Hello, ZK Rocks!");
            }
        });
    }
    ]]></zscript>
</window>
```

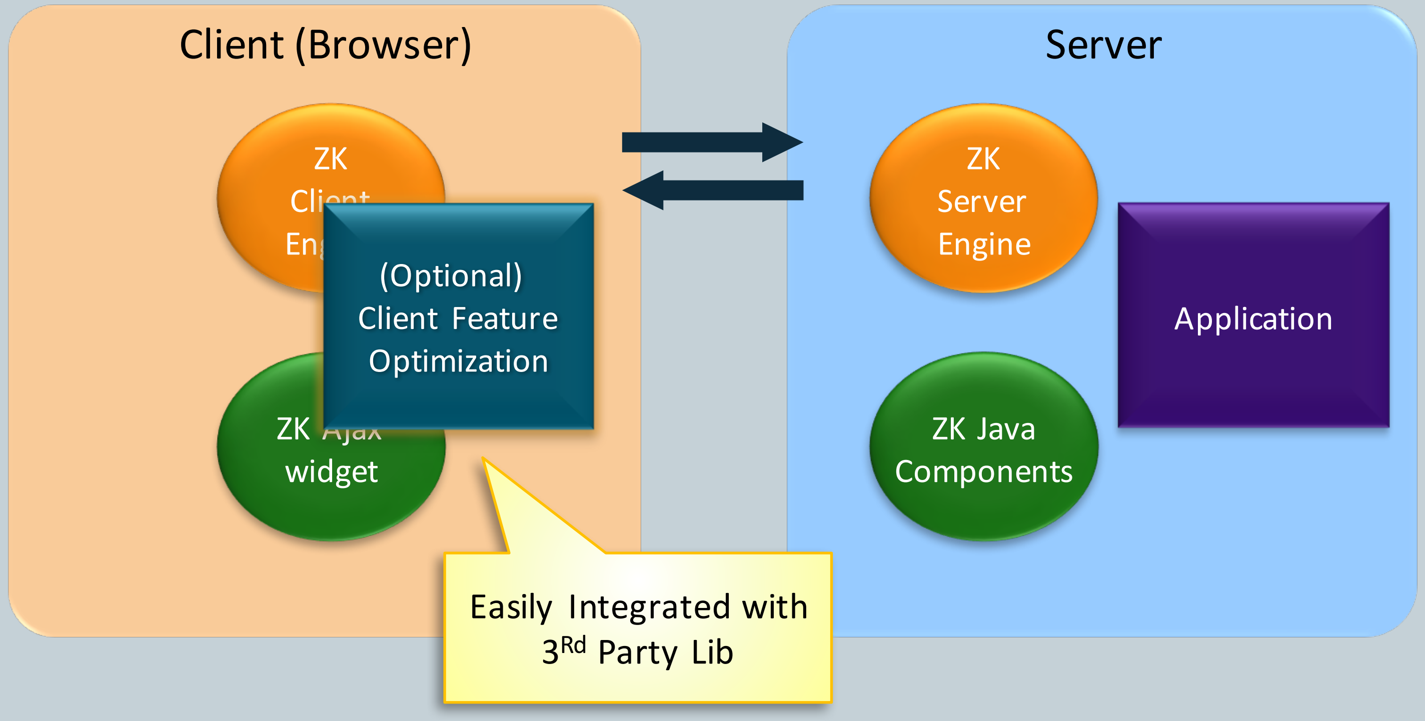
ZK MVC

- Must extends **`SelectorComposer`**
- Apply to a component in zul
- Scoped Components as member Field
- Event Listener as Controller Method
- Handle initialization in **`doAfterCompose`**()

Background Knowledge of ZK

- What is Server + Client Fusion ?
- How ZK Works ?
  - How ZK generates an HTML page
    - ZUL Document Parsing Flow
  - How ZK update the page
    - ZK Page Updating Flow
- Scope in ZK
  - Other Scope
- Component Id Space

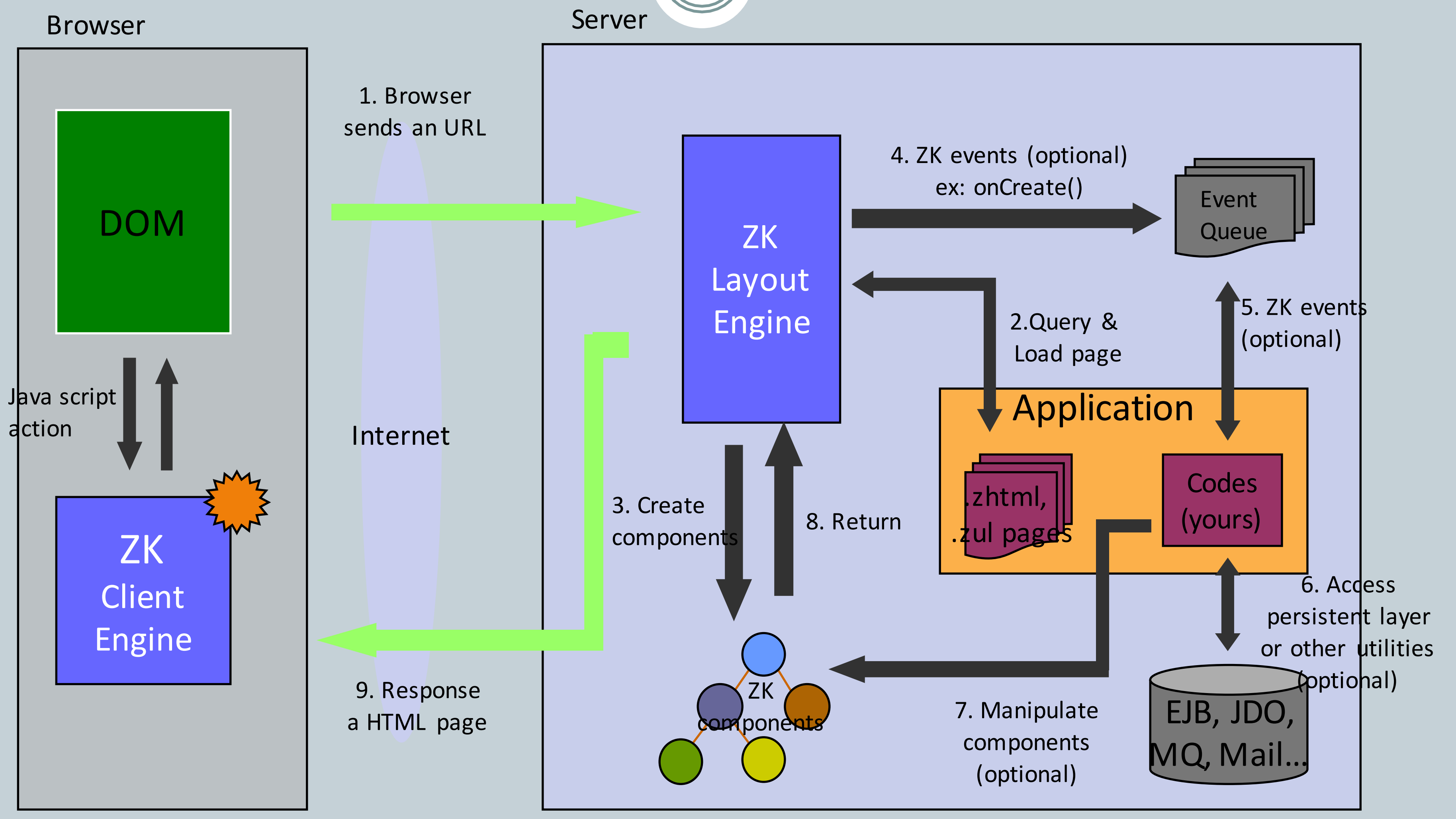
What is Server + Client Fusion ?



Total UI Control In Presentation Layer

Safer Business Layer & Data Layer

How ZK generates an HTML page



ZUL Document Parsing flow

- Create new Desktop instance
- Page Initial phase
- Component creation phase
- Event processing phase
- Rendering phase

Document initial phase

- **Document initial phase**
- Component creation phase
- Event processing phase
- Rendering phase

```
<?page id="userGuide" title="ZK Live Demo"?>
<?init class="pkg.UserRightInit"?>
<window id="win" border="normal" width="200px"
    onCreate="self.setTitle(btn.getLabel())">
    <textbox id="tbx" onChange="win.setTitle(tbx.getValue())"/>
    <button id="btn" label="Hello, ZK!" />
</window>
```

Component creation phase

- Document initial phase
- **Component creation phase**
- Event processing phase
- Rendering phase

```
<?page id="userGuide" title="ZK Live Demo"?>
<?init class="pkg.UserRightInit"?>
<window id="win" border="normal" width="200px"
    onCreate="self.setTitle(btn.getLabel())">
    <textbox id="tbx" onChange="win.setTitle(tbx.getValue())"/>
    <button id="btn" label="Hello, ZK!" />
</window>
```

Event processing phase

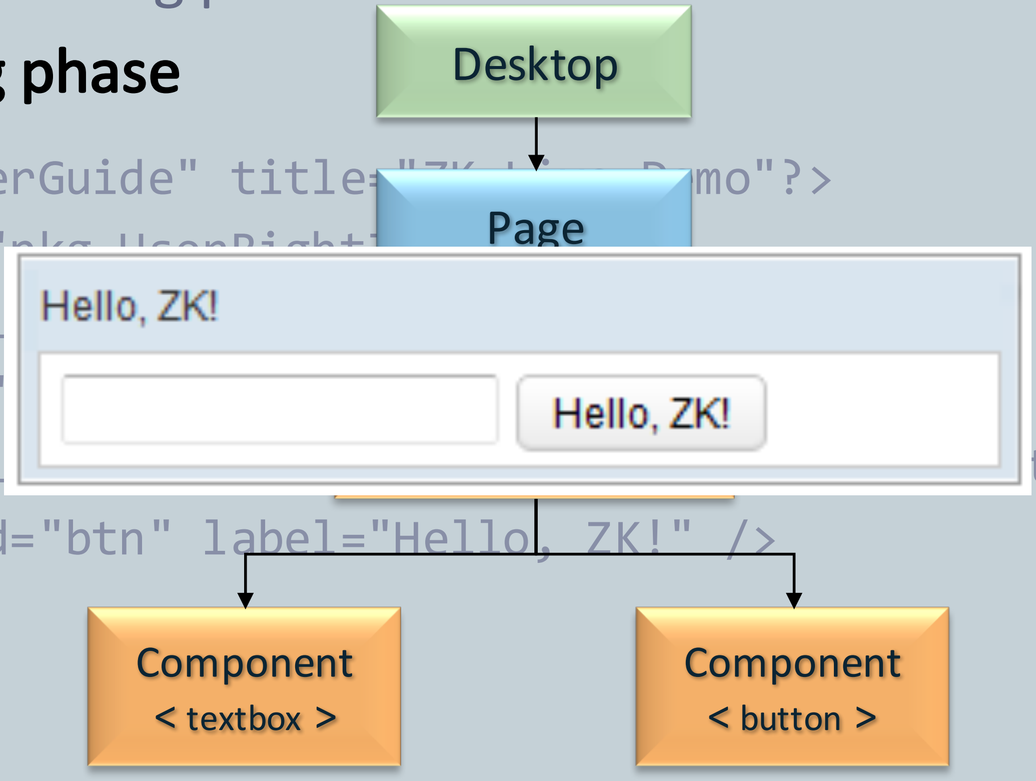
- Document initial phase
- Component creation phase
- **Event processing phase**
- Rendering phase

```
<?page id="userGuide" title="ZK Live Demo"?>
<?init class="pkg.UserRightInit"?>
<window id="win" border="normal" width="200px"
    onCreate="self.setTitle(btn.getLabel())">
    <textbox id="tbx" onChange="win.setTitle(tbx.getValue())"/>
    <button id="btn" label="Hello, ZK!" />
</window>
```

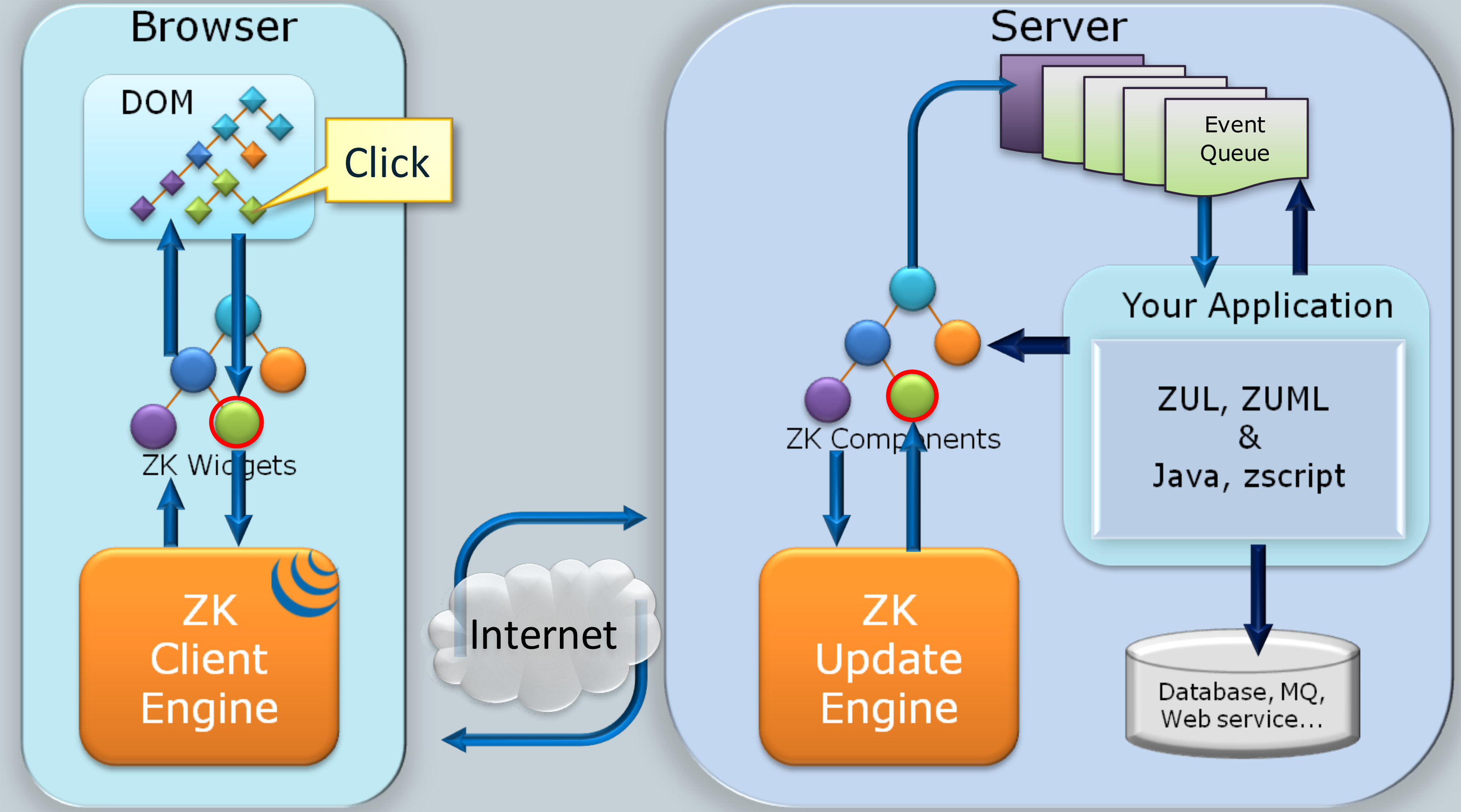
Rendering phase

- Document initial phase
- Component creation phase
- Event processing phase
- **Rendering phase**

```
<?page id="userGuide" title=          mo"?>
<?init class="
<window id="wi
    onCreate="
    <textbox i                                        tValue())"/>
    <button id="btn" label="Hello, ZK!" />
</window>
```



How ZK update the page

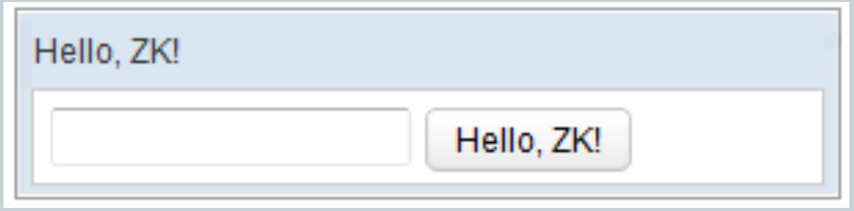


ZK Page Updating Flow

- Request processing phase
- Event processing phase
- Rendering phase

Request processing phase

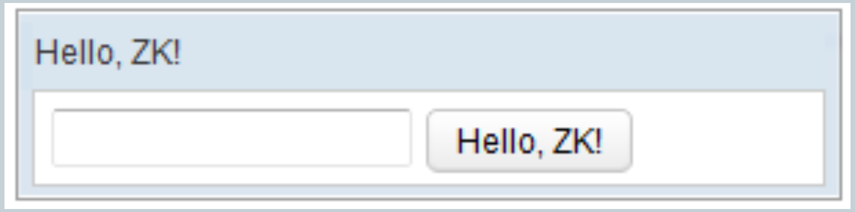
- **Request processing phase**
- Event processing phase
- Rendering phase



1. End user type "New Title" in textbox then tab away
2. Browser fire JavaScript onChange event
3. ZK Client Engine catch that and forward the onChange command to Server side ZK Update Engine via Ajax
4. ZK Update Engine receives the onChange Ajax request from browser with new value for the textbox java component
5. ZK Update Engine then update the textbox component with the new value "New Title" and post onChange event to the Textbox component

Event processing phase

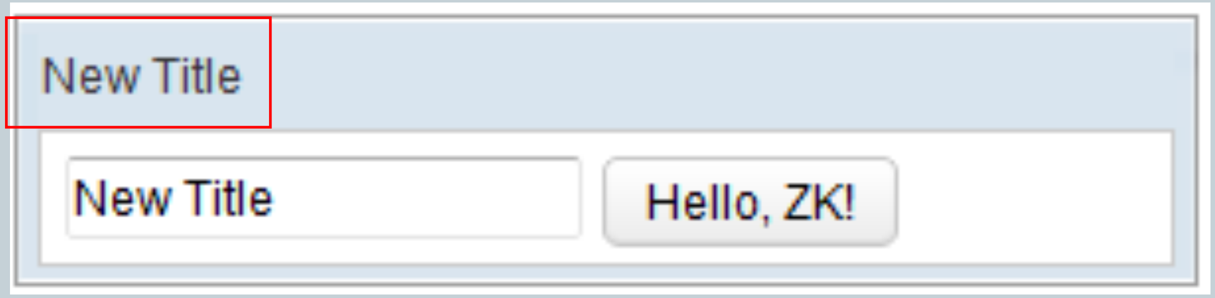
- Request processing phase
- **Event processing phase**
- Rendering phase



```
<?page id="userGuide" title="ZK Live Demo"?>
<?init class="pkg.UserRightInit"?>
<window id="win" border="normal" width="200px"
    onCreate="self.setTitle(btn.getLabel())">
    <textbox id="tbx" onChange="win.setTitle(tbx.getValue())"/>
    <button id="btn" label="Hello, ZK!" />
</window>
```

Rendering phase

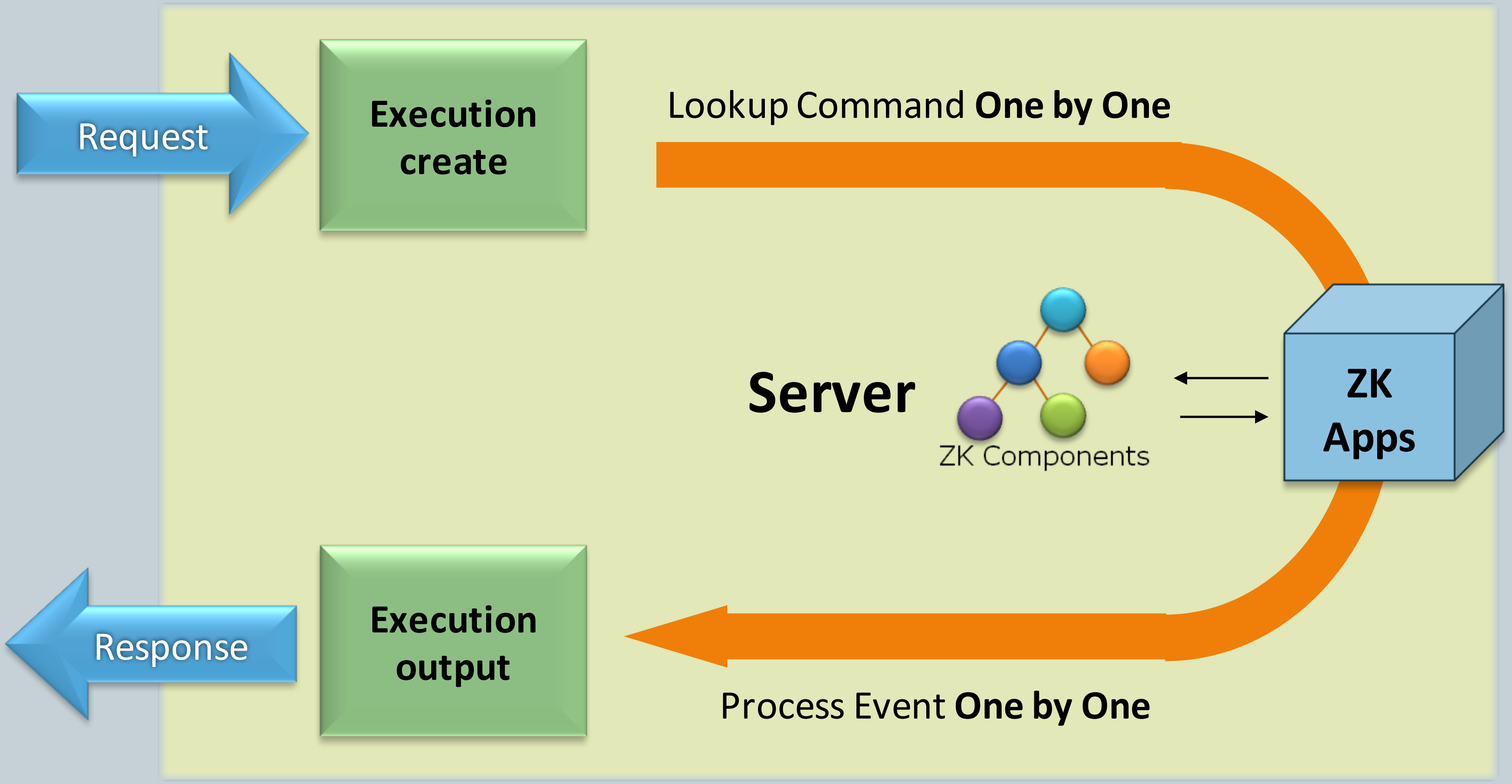
- Request processing phase
- Event processing phase
- **Rendering phase**



Scope of ZK

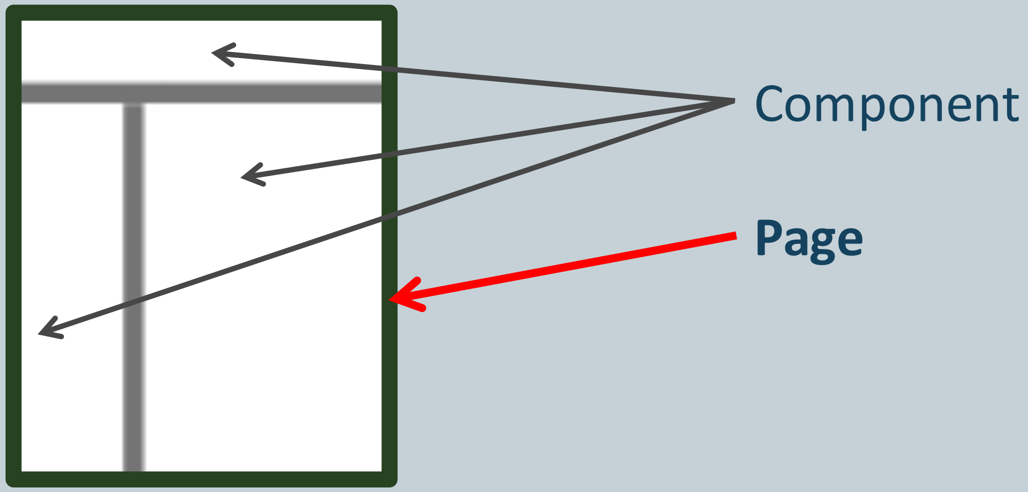
- Execution
- Page
- Desktop

Execution


Request
Execution create
Lookup Command **One by One**
Server
ZK Components
ZK Apps
Process Event **One by One**
Execution output
Response

Page

- A collection of components
- Confines the components belonging to the page so it display in a certain portion of the browser.
- <?page title="**My Page Title**"?> (optional)



Desktop

- An **URL** representing the browser screen
- A collection of pages serving the same URL request
- Implicitly created by ZK when end users visit an URL.

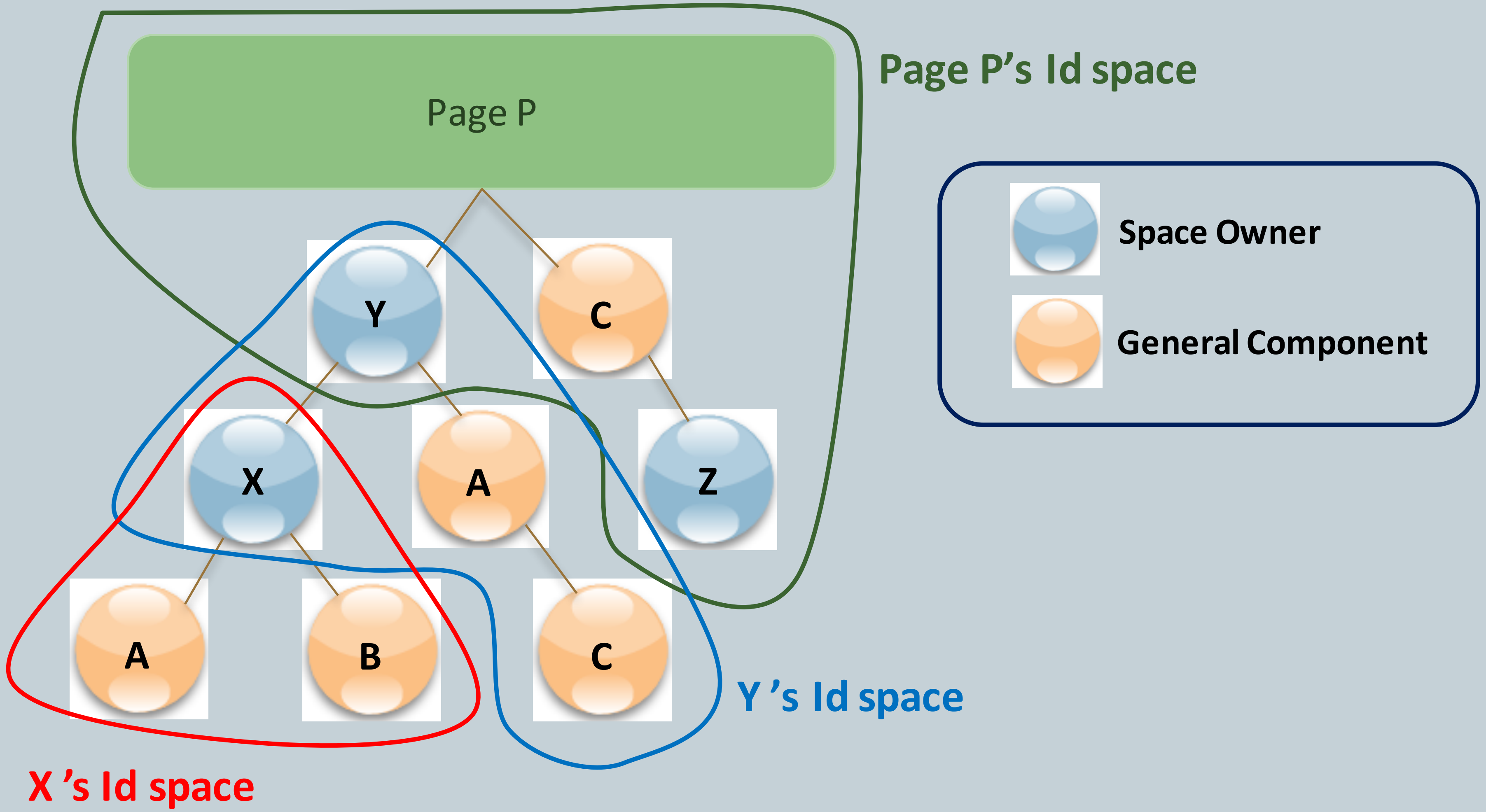
Other Scope

- Request
- Response
- Session
- Application
- Component
- Id Space

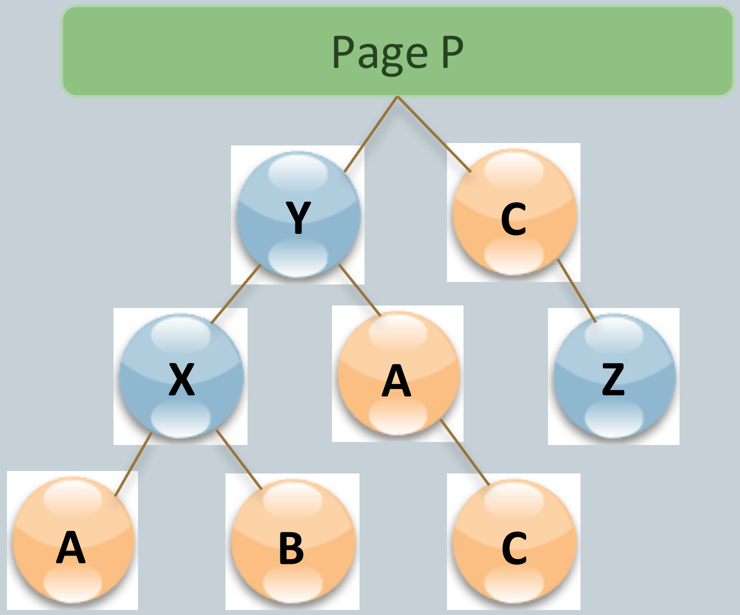
Id Space Scope

- You can Identify Component by Id
- Group Components in an Id Space
  - You can use the same id within different Id space.
- Page and window are the two default IdSpaces

Id Space Scope - View



Id Space Scope - API



● Space Owner

● General Component

```
Component A = Y.getFellow("A");
Page P = C.getSpaceOwner();
Component Y = Y.getSpaceOwner();
```

Q & A

Invocation Sequence

```
<zscript>
public class MyButton extends Button {
    public void onClick(Event event) {
        Messagebox.show("Hello, ZK2! ");
    }
}
</zscript>
<button id="btn" label="hi"
    onClick='Messagebox.show("Hello, ZK1! ")' use="MyButton"/>
<zscript>
btn.addEventListener("onClick", new EventListener() {
    public void onEvent(Event evt) {
        Messagebox.show("Hello, ZK3!");
    }
});
</zscript>
```

Invocation Sequence (cont.)

- **Hello, ZK1!** -- Define as attribute
- **Hello, ZK3!** -- Dynamically add via addEventListener()
- **Hello, ZK2!** -- Declare as member method

ZUL way or Swing way

- ZK support both ZUL way and Swing way to design a page.
- Quick prototyping (no compiling, link, deployment cycle)
- Intuitive data model perception (the document is the screen).
- Easy migration path to Ajax for legacy HTML & JSP pages (mixing tags in the same page)
- Screen designer does not have to know Java programming (separation of labors).

ZK Resources

- Documents
  http://books.zkoss.org
- ZK Forum
  http://www.zkoss.org/forum/
- ZK Javadoc
  http://www.zkoss.org/javadoc/

Summary

- Rich user experience
- Simple programming model
- Extensible server centric architecture
- Markup and Script Languages