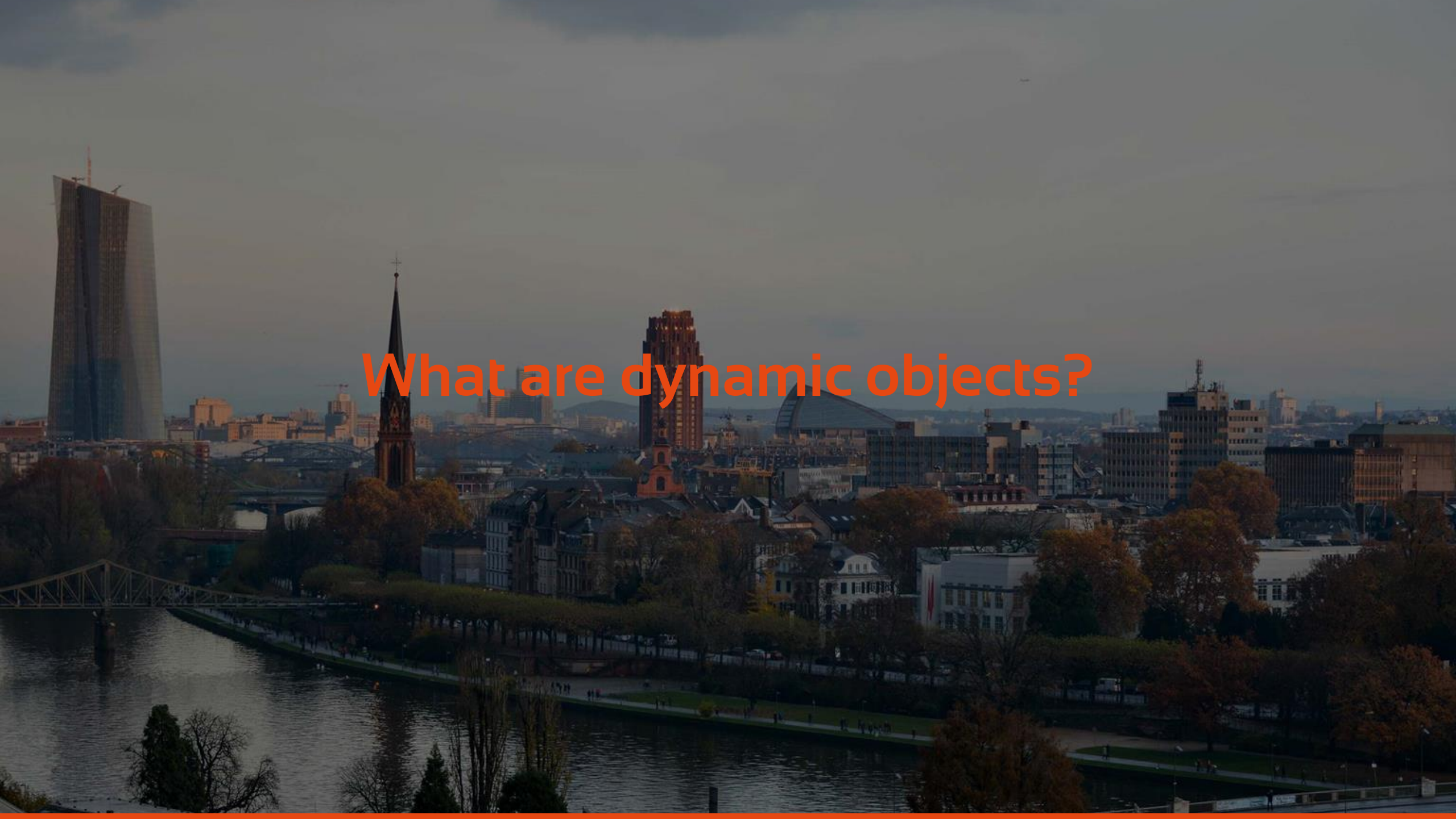

DataFlex Entwickler Tag 2019
Dynamic Objects
Dion Ikink

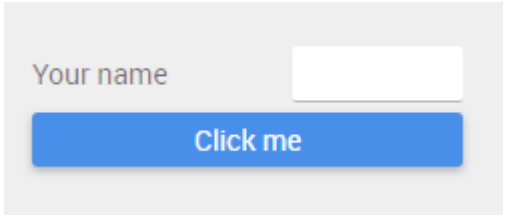
Contents

- What are static objects?
- What are dynamic objects?
- What makes it difficult?
- Dynamic Objects Library
- Demo
- Conclusion


What are dynamic objects?

What are static objects?

```
Object oForm is a cWebForm
        Set psLabel to "Your name"
        Set piColumnSpan to 4
End_Object

Object oButton is a cWebButton
        Set psCaption to "Click me"
        Set piColumnSpan to 4
End_Object
```



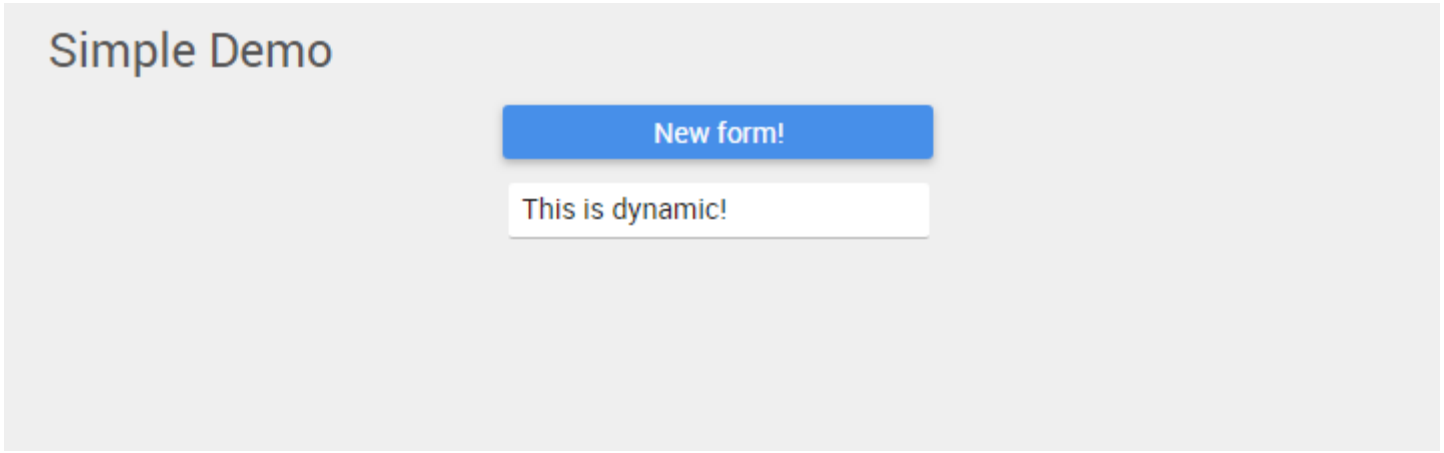
What are static objects?

- Regular DataFlex (web) objects
- Defined at compile time

- “Static” in their behavior
  - Functions
  - Procedures
  - Location

- Loaded quickly at runtime
- Little to no flexibility

What are dynamic objects?



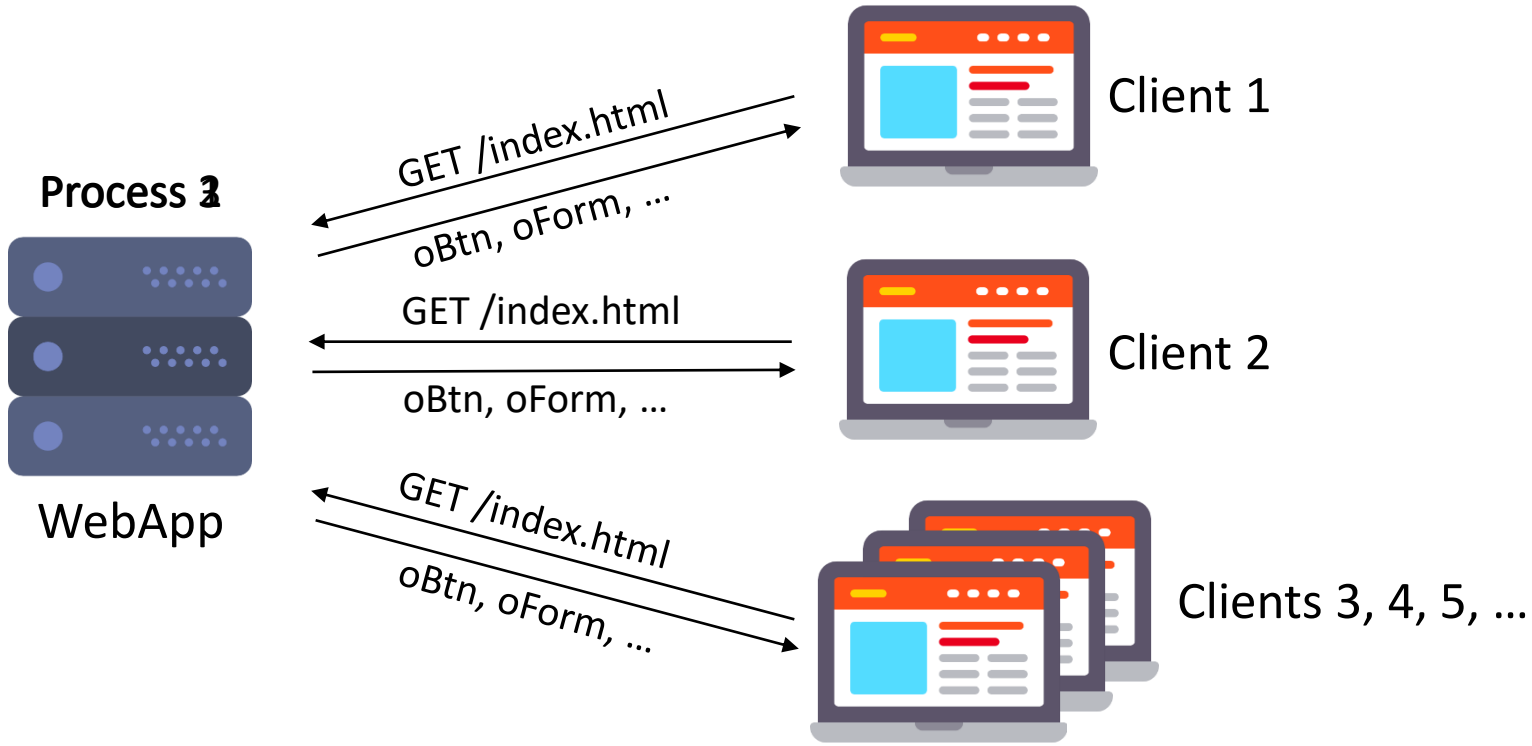
What are dynamic objects?

- Defined & created at runtime
- “Dynamic” in their behavior
  - Changing location
  - Appearing when necessary
- More overhead
  - Longer load times
  - More memory usage
- Highly flexible

What makes it difficult?

- Process pooling

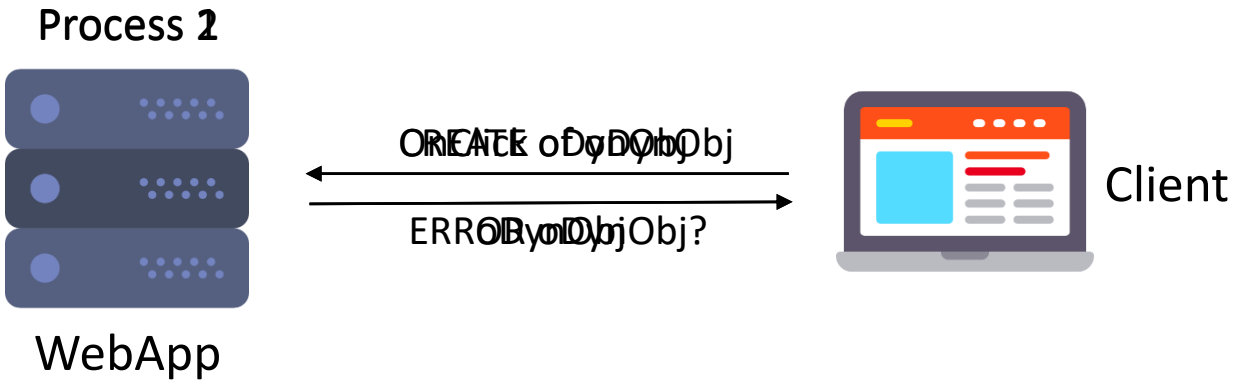
Process pooling



Process pooling

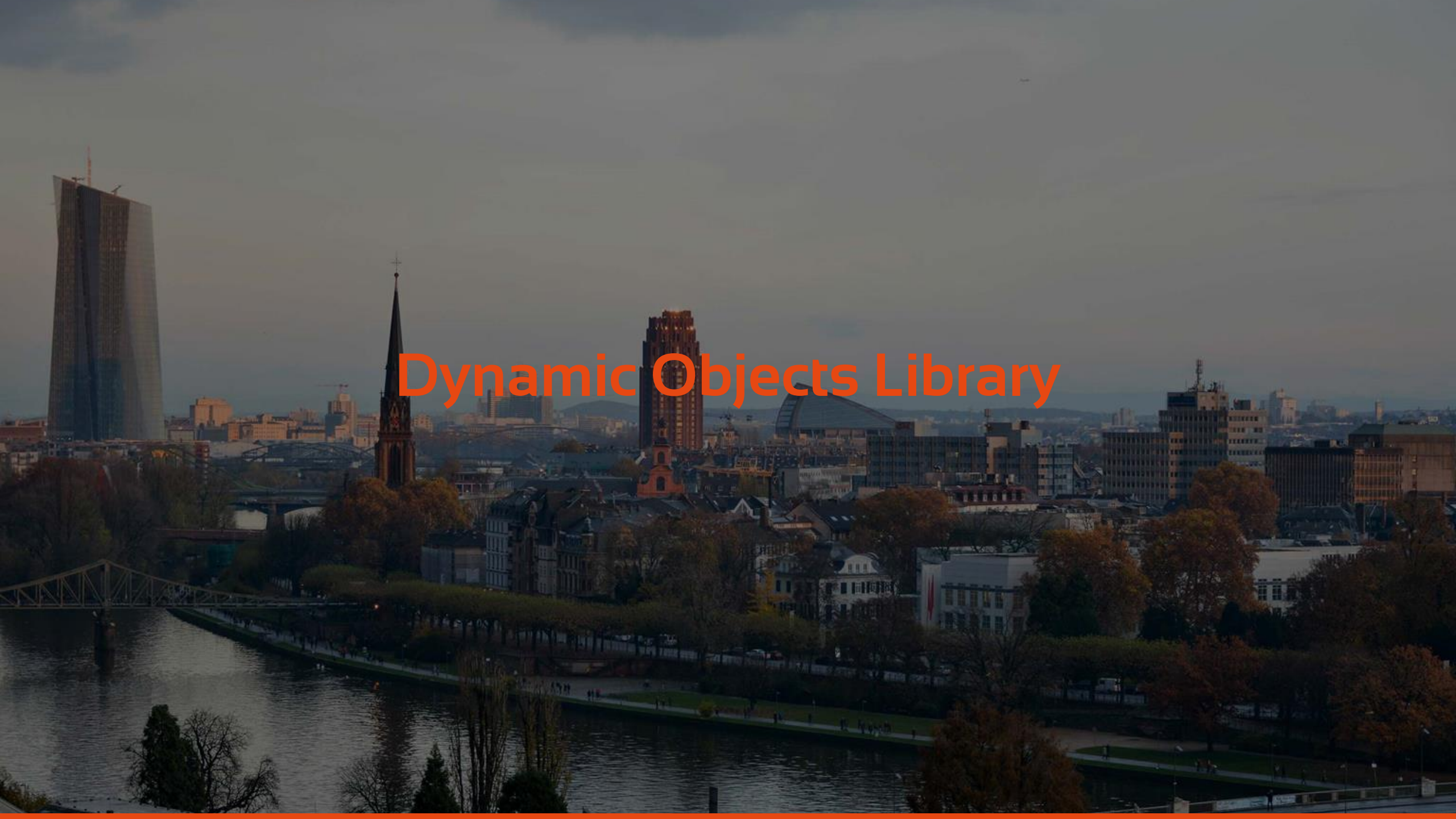
- Objects defined at compile time
  - Up-to-date overview for all processes
- Client loads dynamic objects
  - Process 1 creates them
- Client interacts with dynamic objects
  - Process 2 is unaware
    - Errors!

Process pooling



What makes it difficult?

- Process pooling
- Synchronization
  - Server ↔ clients
    - Properties
    - Location

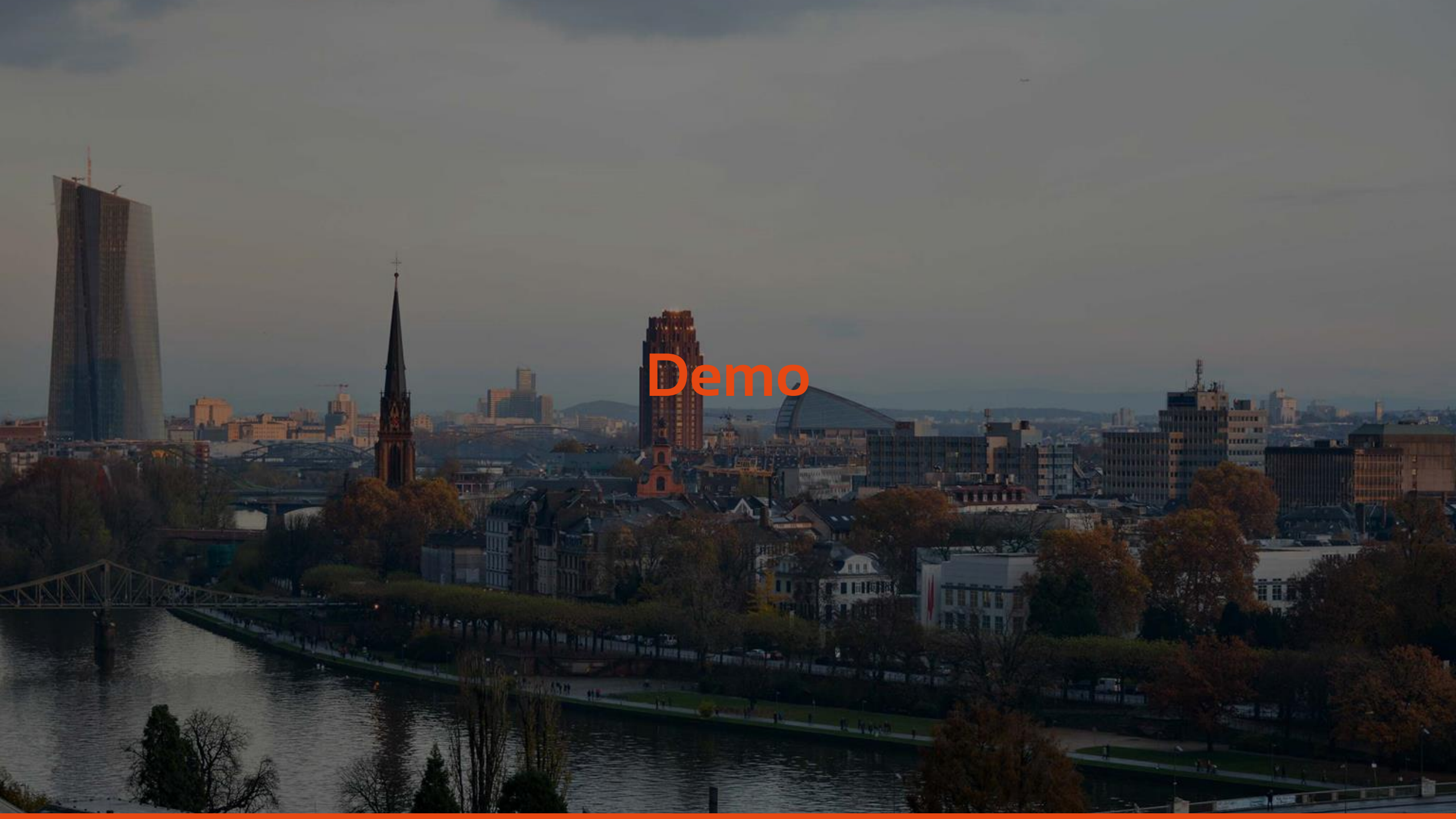

Dynamic Objects Library

Dynamic Objects Library

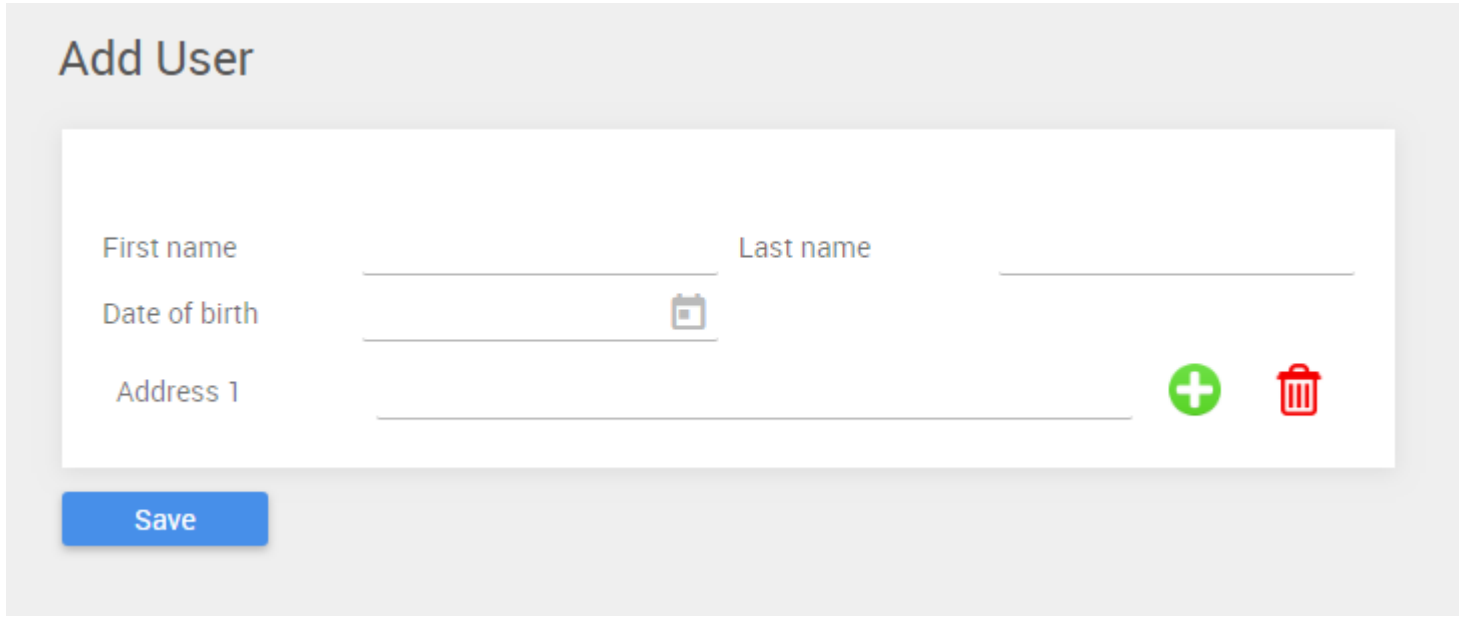
- Handles process pooling
  - Creates server objects when necessary
  - Saves objects outside process memory
- Handles synchronization
- Provides API
  - Insert
  - Move (in development)
  - Delete

Dynamic Objects Library

- Subclassing
  - Define object-specific behavior
    - Object interaction
    - Functions, procedures & events
- Dynamic Properties
  - Initial properties of dynamic objects
  - Overwrite properties set in subclass


Demo

Demo



Dynamic Objects Library

`cWebDynamicObjectContainer`

- Core of the library

- Functions like a container
  - Holds only dynamic objects
  - Multiple containers can exist

Dynamic Objects Library

Function CreateDynamicObject

- Integer iClassId
  - TIP: Use RefClass
- String sDynamicObjectId
  - Must be unique
- String sParentId
  - ID of dynamic parent
  - Empty string for root object
- Returns Handle

Dynamic Objects Library

Procedure InitDynamicProp

- `String sPropName`
- `Variant vPropValue`

Dynamic Objects Library

Procedure InsertAfter

- `String sDynamicObjectId`
- `String sInsertAfter`

Dynamic Objects Library

Procedure DestroyDynamicObject

- `String sDynamicObjectId`

Workflow

1. Create container
2. Define object-specific behavior
   - Procedures & functions
3. Creation Phase
   - Fill container with initial objects
4. Activation of container
   - Display initial dynamic objects
5. Modification Phase
   - Insert new/modify existing objects

Usually done together

Workflow

1. Create container

```
Object oAddressContainer is a cWebDynamicObjectContainer
    Set piColumnCount to 12

    { WebProperty=Client }
    Property Integer piRows 1
End_Object
```

Workflow

1. Create container

```
Procedure OnLoad
    Handle hoObj
    Integer iRows

    WebGet piRows to iRows

    Get CreateDynamicObject (RefClass(cWebForm)) ("oAddressForm" + String(iRows)) "" to hoObj
    Send InitDynamicProp of hoObj "psLabel" ("Address " + String(iRows))
    Send InitDynamicProp of hoObj "piColumnSpan" 10

    Get CreateDynamicObject (RefClass(cAddRow)) "oAddRow" "" to hoObj
    Send InitDynamicProp of hoObj "piColumnIndex" 10

    Get CreateDynamicObject (RefClass(cRemoveRow)) "oRemoveRow" "" to hoObj
    Send InitDynamicProp of hoObj "piColumnIndex" 11

    Send Activate
End_Procedure
```

Part of step 3

Part of step 4

Workflow

1. Create container

```
Get CreateDynamicObject (RefClass(cWebForm)) ("oAddressForm" + String(iRows)) "" to
hoObj
Send InitDynamicProp of hoObj "psLabel" ("Address " + String(iRows))
Send InitDynamicProp of hoObj "piColumnSpan" 10

Get CreateDynamicObject (RefClass(cAddRow)) "oAddRow" "" to hoObj
Send InitDynamicProp of hoObj "piColumnIndex" 10

Get CreateDynamicObject (RefClass(cRemoveRow)) "oRemoveRow" "" to hoObj
Send InitDynamicProp of hoObj "piColumnIndex" 11

Send Activate
```

Workflow

2. Define object-specific behavior

- `cAddRow (cWebImage)`

- `cRemoveRow (cWebImage)`

Workflow

2. Define object-specific behavior (add row)

```
Class cAddRow is a cWebImage
    Procedure Construct_Object
        Forward Send Construct_Object

        Set psUrl to "Images/add.png"
        Set piColumnSpan to 1
        Set piHeight to 30
        Set pePosition to wiFit
    End_Procedure
End_Class
```

Workflow

2. Define object-specific behavior (remove row)

```
Class cRemoveRow is a cWebImage
    Procedure Construct_Object
        Forward Send Construct_Object

        Set psUrl to "Images/delete.png"
        Set piColumnSpan to 1
        Set piHeight to 30
        Set pePosition to wiFit
    End_Procedure
End_Class
```

Workflow

2. Define object-specific behavior (add row)

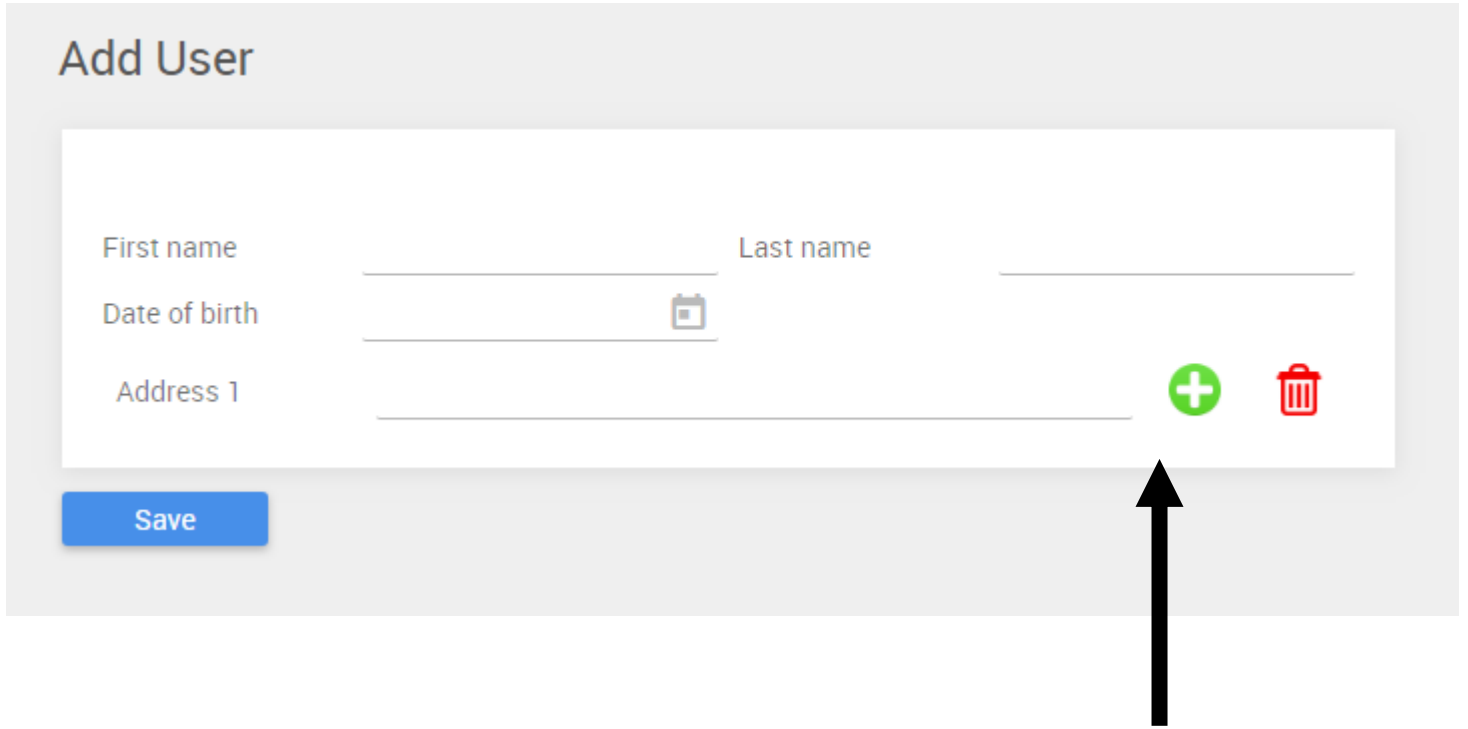
```
Procedure OnClick
    Integer iLastRow
    Handle hoOwner hoObj

    Get Owner to hoOwner
    WebGet piRows of hoOwner to iLastRow

    Get CreateDynamicObject (RefClass(cWebForm)) ("oAddressForm" + String(iLastRow + 1)) "" to hoObj
    Send InitDynamicProp of hoObj "psLabel" ("Address " + String(iLastRow + 1))
    Send InitDynamicProp of hoObj "piColumnSpan" 10
```

Part of step 5

```
    Send InsertAfter of hoOwner ("oAddressForm" + String(iLastRow + 1)) ("oAddressForm" + String(iLastRow))
```

```
    WebSet piRows of hoOwner to (iLastRow + 1)
End_Procedure
```

Demo



Workflow

2. Define object-specific behavior (remove row)

```
Procedure OnClick
    Integer iLastRow
    Handle hoOwner

    Get Owner to hoOwner
    WebGet piRows of hoOwner to iLastRow

    If (iLastRow > 0) Begin
        Send DestroyDynamicObject of hoOwner ("oAddressForm" + String(iLastRow))

        WebSet piRows of hoOwner to (iLastRow - 1)
    End
End_Procedure
```

Part of step 5

Workflow

3. Creation Phase

- (Initially) fill container with objects
- Objects appear in order
- Shown in step 1

```
Get CreateDynamicObject (RefClass(cWebForm)) ("oAddressForm" + String(iRows)) "" to hoObj
```

Workflow

4. Activation

- Signals client to render dynamic objects
- End of Creation Phase
- Shown in step 1

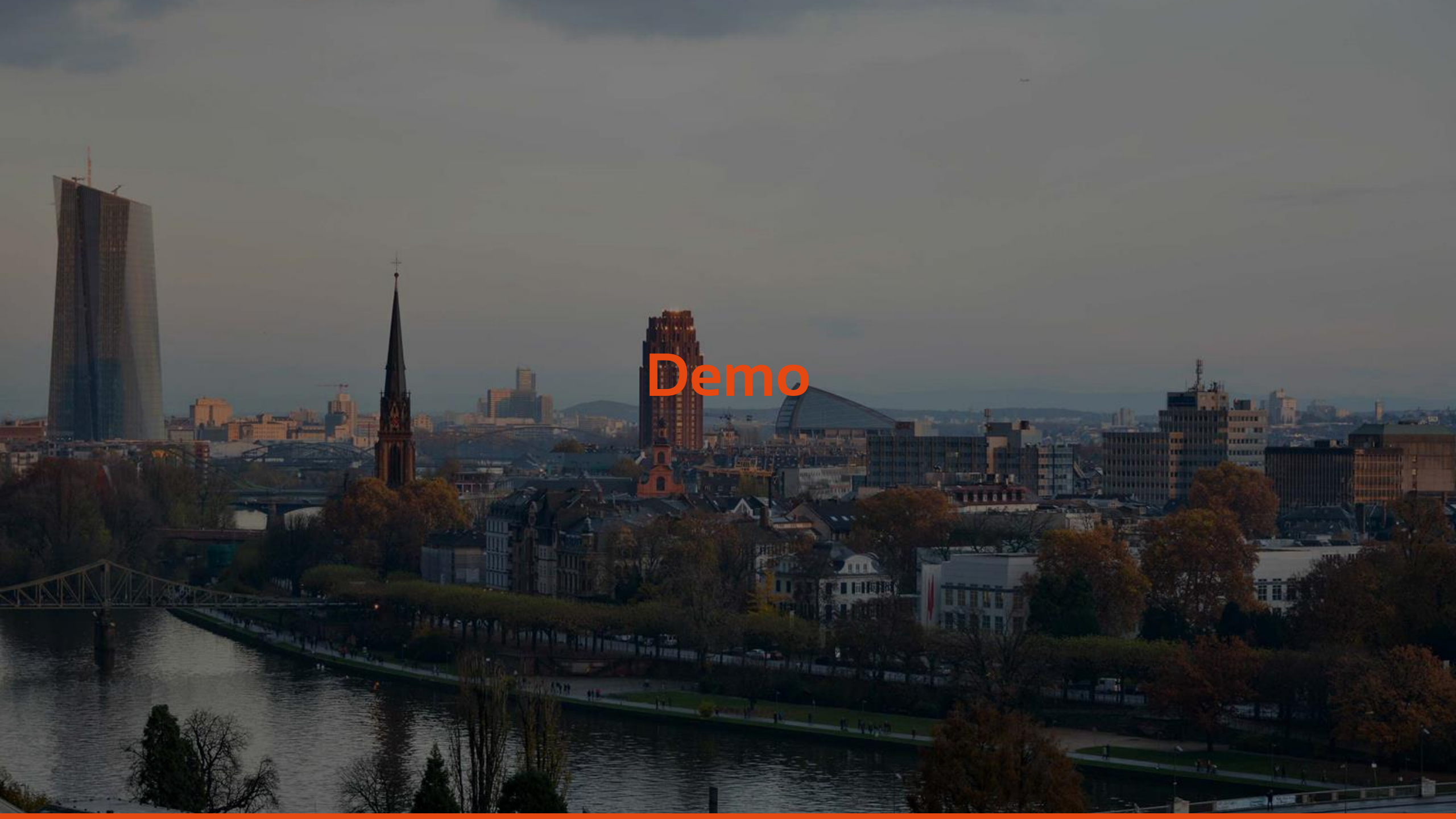
Send Activate

Workflow

5. Modification Phase

- Insert, Move or Destroy dynamic objects
- Shown in step 2

```
Send InsertAfter of hoOwner ("oAddressForm" + String(iLastRow + 1)) ("oAddressForm" + String(iLastRow))
Send DestroyDynamicObject of hoOwner ("oAddressForm" + String(iLastRow))
```


Demo

Conclusion

- Dynamic Objects Library
  - Create & modify objects at runtime
- Two important concepts
  - Subclassing
  - Dynamic Properties
- Five step workflow
  - Detailed description in manual


DataFlex Entwickler Tag 2019
Vielen Dank für Ihre Aufmerksamkeit!
Haben Sie Fragen?