Dynamic Elements

Johann Werner
NUREG GmbH

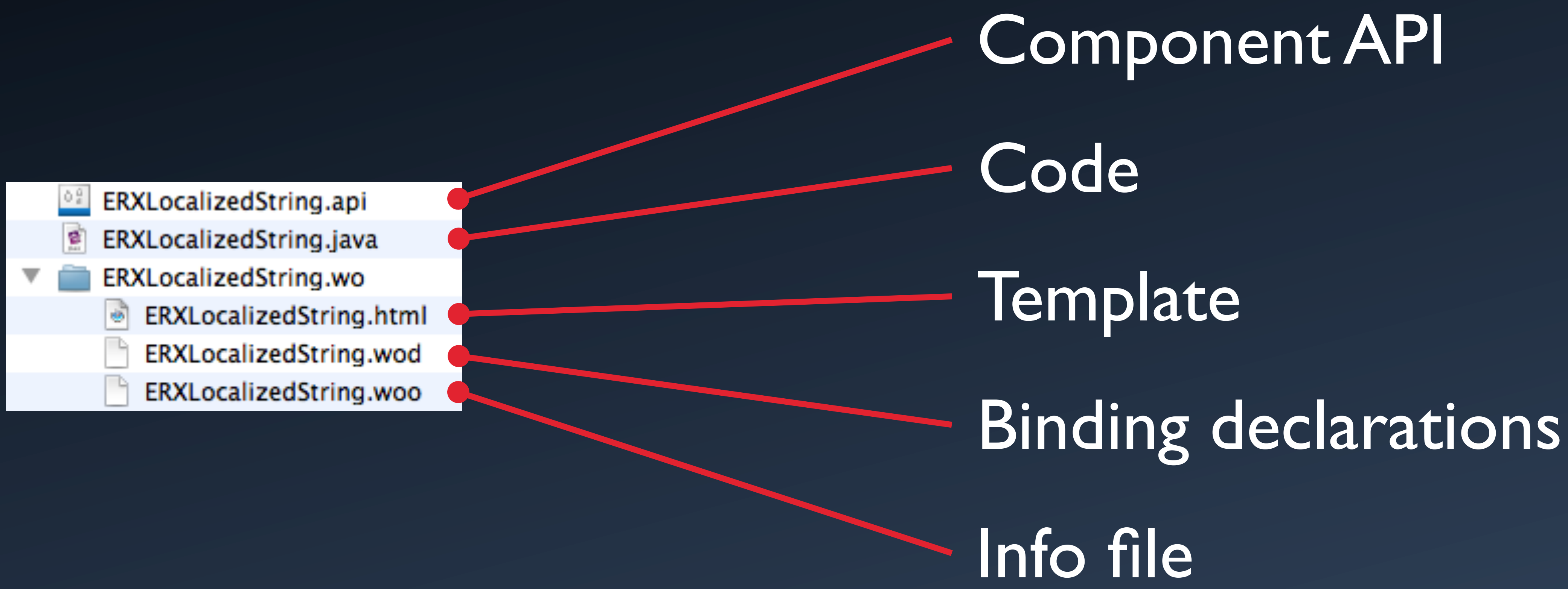
Outline

- What are components
- Different common types
- Dynamic components

Components

- reusable pieces of representation + functionality
- WebObjects comes with many
  (e.g. WOBody, WORepetition, WOString, …)
- Wonder adds a whole bunch
  (e.g. AjaxModalContainer, ERXLocalizedString, …)

Components



ERXLocalizedString.api
ERXLocalizedString.java
ERXLocalizedString.wo
ERXLocalizedString.html
ERXLocalizedString.wod
ERXLocalizedString.woo

Standard Component

- subclass of WOComponent
- automatic push/pull of binding values
- preserves state

Standard Component

Parent

```
[…]
<wo:MyString value="Hello World" />
[…]
```

Code

```
public class MyString extends WOComponent {
    public String value;

    public MyString(WOContext context) {
        super(context);
    }
}
```

Template

```
<wo:str value="$value" />
```

Non-Synchronizing Component

- manual push/pull of binding values

➡ less overhead

- preserves state

Non-Synchronizing Component

Parent

```
[…]
<wo:MyString value="Hello World" />
[…]
```

Code

```
public class MyString extends WOComponent {
    public MyString(WOContext context) {
        super(context);
    }

    @Override
    public boolean synchronizesVariablesWithBindings() {
        return false;
    }

    public String value() {
        return (String) valueForBinding("value");
    }
}
```

Template

```
<wo:str value="$value" />
```

Non-Synchronizing Component

Parent

```
[…]
<wo:MyString value="Hello World" />
[…]
```

Code

```
public class MyString extends WOComponent {
    public MyString(WOContext context) {
        super(context);
    }

    @Override
    public boolean synchronizesVariablesWithBindings() {
        return false;
    }
}
```

Template

```
<wo:str value="$^value" />
```

Stateless Component

- manual push/pull of binding values

➡ less overhead

- no state
- single shared instance*

➡ less memory, fewer Java object-creations

Stateless Component

Parent

```
[…]
<wo:MyString value="Hello World" />
[…]
```

Code

```
public class MyString extends WOComponent {
    public MyString(WOContext context) {
        super(context);
    }

    @Override
    public boolean isStateless() {
        return true;
    }
}
```

Template

```
<wo:str value="$^value" />
```

Stateless Component

Parent

```
[…]
<wo:MyString value="Hello World" />
[…]
```

Code

```
public class MyString extends WOComponent {
    private Object myVar;

    public MyString(WOContext context) {
        super(context);
    }

    @Override
    public boolean isStateless() {
        return false;
    }

    @Override
    public void reset() {
        super.reset();
        myVar = null;
    }
}
```

Template

```
<wo:str value="$^value" />
```

Dynamic Component

- subclass of WODynamicElement
- no template

➡ less memory, faster

- direct access to bindings (WOAssociation)
- direct access to children elements
- must be thread-safe!

Dynamic Component

Parent

```
[…]
<wo:MyString value="Hello World" />
[…]
```

Code

```
public class MyString extends WODynamicElement {
    private WOAssociation value;

    public MyString(String name, NSDictionary<String, WOAssociation> associations, WOElement template) {
        super(name, associations, template);
        value = associations.get("value");
    }

    @Override
    public void appendToResponse(WOResponse response, WOContext context) {
        WOComponent component = context.component();
        response.appendContentString((String) value.valueInComponent(component));
    }
}
```

Demo

Demo Results

- The more specialized component class the faster it is
- Stateless and dynamic components create less Java objects

When should I use…

Standard components

- for pages, big blocks

Non Synchronizing components

- smaller components that are used often
- binding has
  - different name as the variable to hold it
  - no variable counterpart

When should I use…

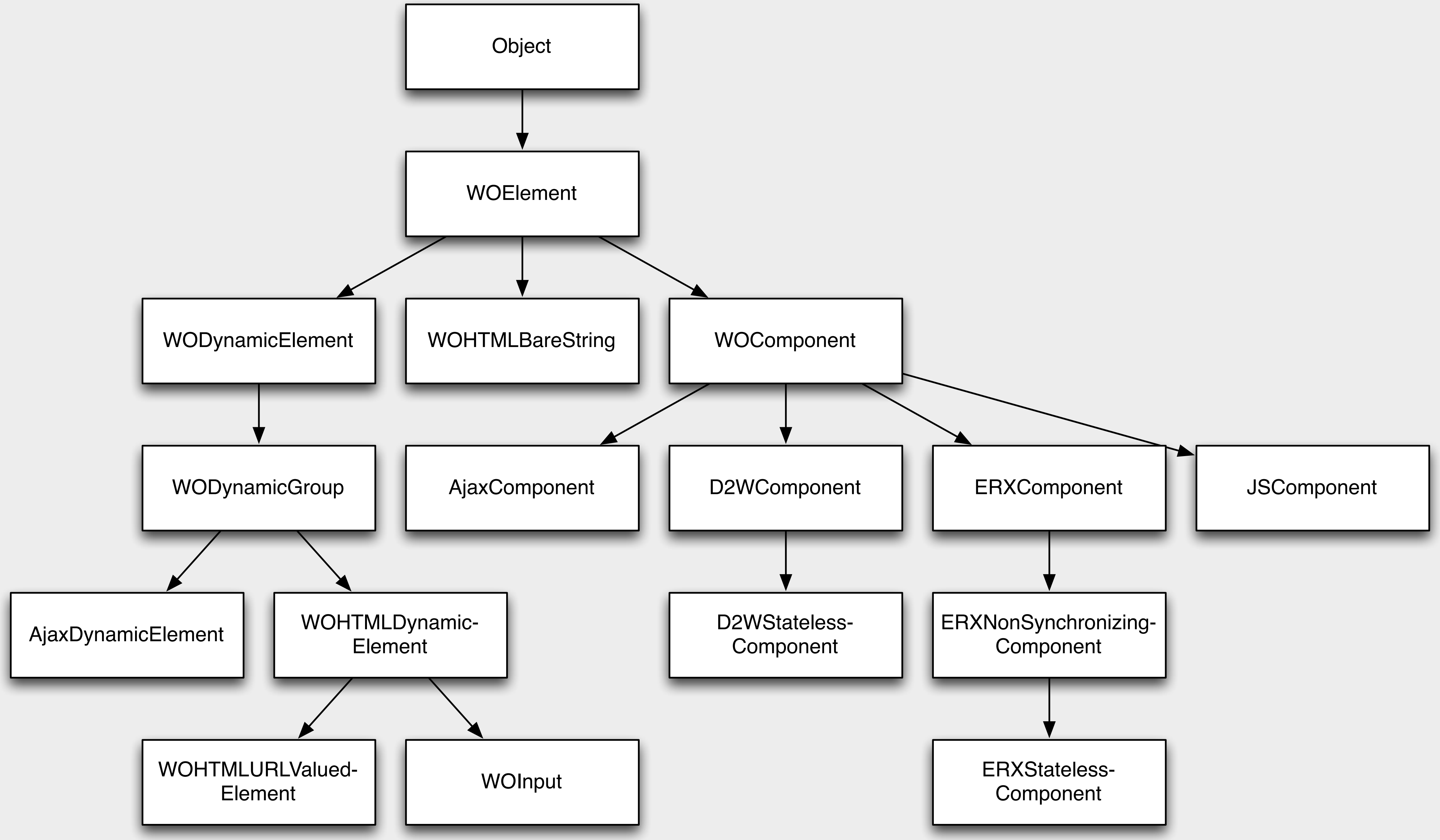
Stateless components

- no state necessary
- memory optimization

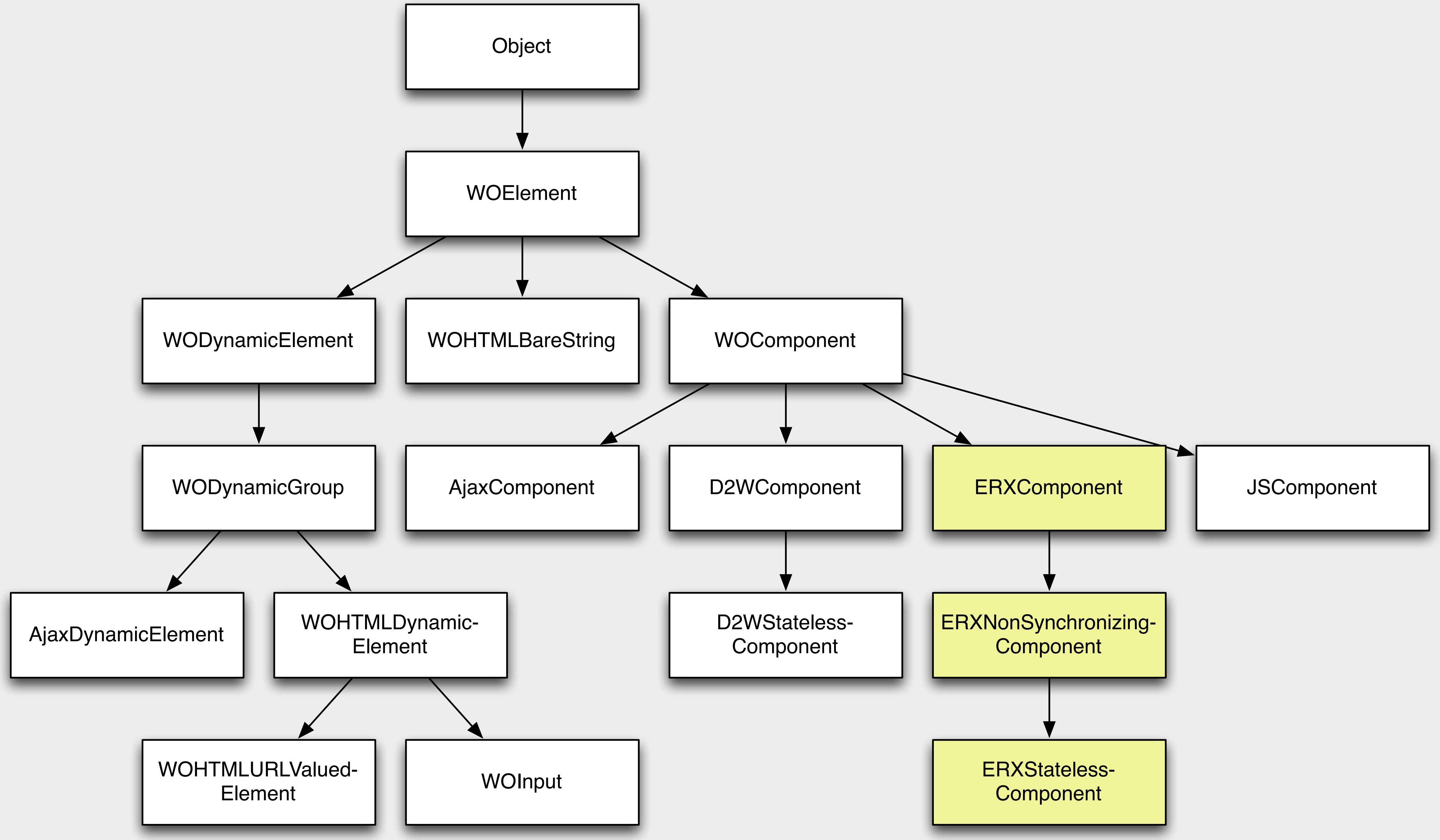
Dynamic components

- no state necessary
- complex output generation
- access to child elements

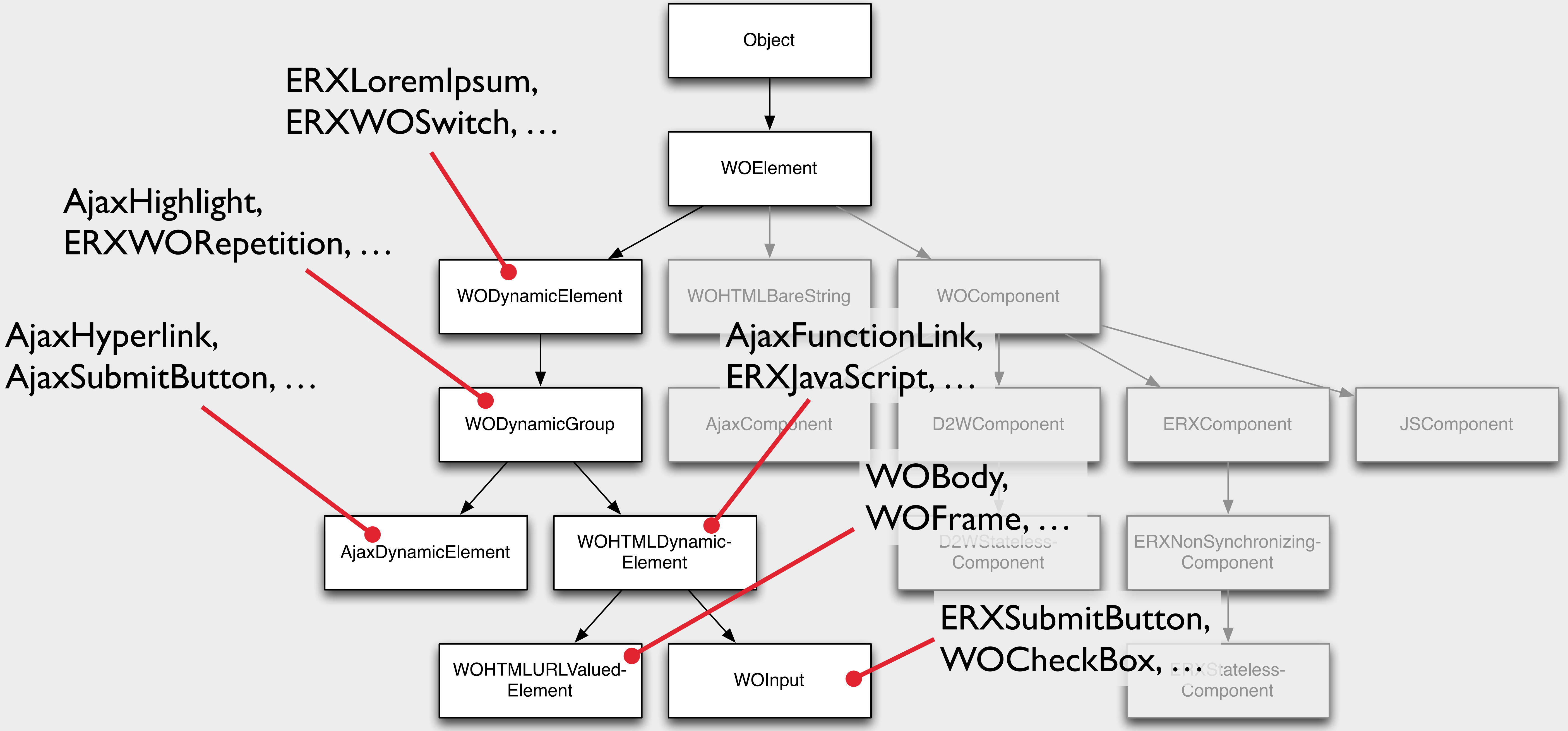
Component Class Tree



Component Class Tree

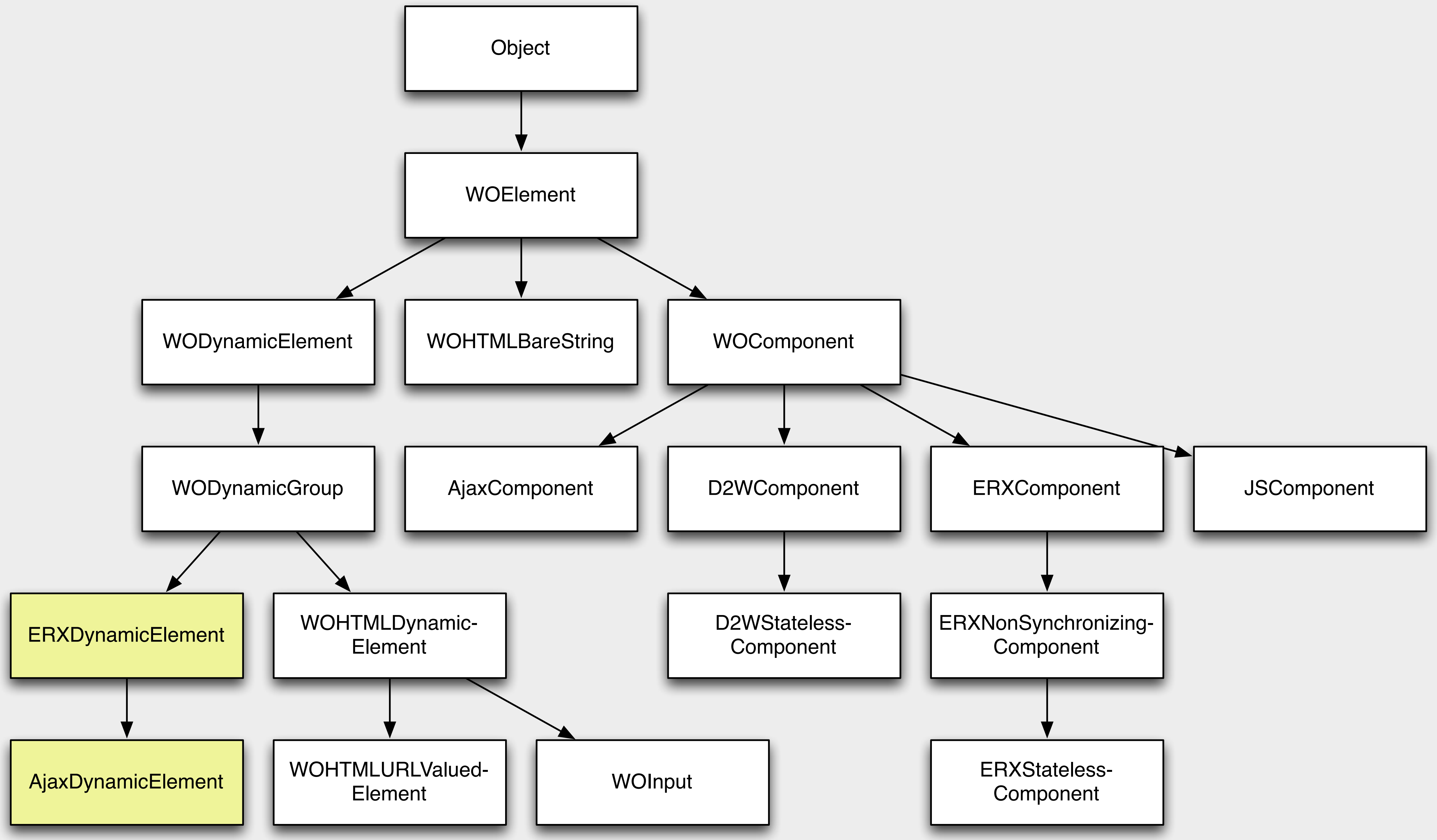


Component Class Tree



WOWODC 2012

Component Class Tree



Dynamic Base Classes

- ERXDynamicElement in ERXExtensions
- AjaxDynamicElement in Ajax

DEMO

Creating Dynamic Element

- extend ERXDynamicElement / AjaxDynamicElement
- appendToResponse
  - response.appendContentString(…)
  - appendTagAttributeToResponse(response, name, value)
  - appendChildrenToResponse(response, context)

Creating Dynamic Element

- xxxValueForBinding(name, [default,] component)
- ContextData
  - beforeProcessing(context)
  - afterProcessing(context)

WOWODC '012

MONTREAL JUNE 30, JULY 1ST AND 2ND 2012

Q&A