DataFlex

WEBAPP FRAMEWORK EVOLUTION

NIELS DE BOER

01

START OF THE WEBAPP FRAMEWORK

START OF THE WEBAPP FRAMEWORK
THE START

- DataFlex 17.1
- Rendering
- Client-side handeling
- Client server communication
- Ecmascript 5
- Prototype inheritance
- Var variables

START OF THE WEBAPP FRAMEWORK
DEFINECLASS

- defineClass
- Old class system
- Prototype chaining
- Eval (bad!)
- Eval(“2 + 2”)

START OF THE WEBAPP FRAMEWORK
WHAT DID CLASSES LOOK LIKE?

- Constructor function
- Df.defineclass

```
df.WebButton = function WebButton(sName, oParent){
    df.WebButton.base.constructor.call(this, sName, oParent);

    //  Web Properties
    this.prop(df.tString, "psCaption", "");
    this.prop(df.tString, "psTextColor", "");
    this.prop(df.tString, "psWaitMessage", "");
    this.prop(df.tBool, "pbShowWaitDialog", false);
    this.prop(df.tInt, "peAlign", -1);

    //  Events
    this.event("OnClick", df.cCallModeWait);



    //  Configure super classes
    this.pbShowLabel = false;

    // @privates
    this._sControlClass = "WebButton";
    this._bJSSizing = false;
};
```

```
df.defineClass("df.WebButton", "df.WebBaseControl",{

create : function(){
    df.WebButton.base.create.apply(this, arguments);

    this.set_pbShowWaitDialog(this.pbShowWaitDialog);
},
```

START OF THE WEBAPP FRAMEWORK
WHAT DID MIXINS LOOK LIKE?

- Constructor function
- getBase
- defineClass without super
- Df.mixin
- 4 steps to get it to work

```
df.WebColumn_mixin = function WebColumn_mixin(sName, oParent){
    this.getBase("df.WebColumn_mixin").constructor.call(this, sName, oParent);

    //  Assertions
    if(!(oParent && oParent instanceof df.WebList)){
        throw new df.Error(999, "WebColumn object '{{0}}' should be placed inside a Web
    }
```

```
df.defineClass("df.WebColumn_mixin", {
```

```
//  Generate new base class using df.WebColumn_mixin and df.WebForm
df.WebColumnBase = df.mixin("df.WebColumn_mixin", "df.WebForm");
```

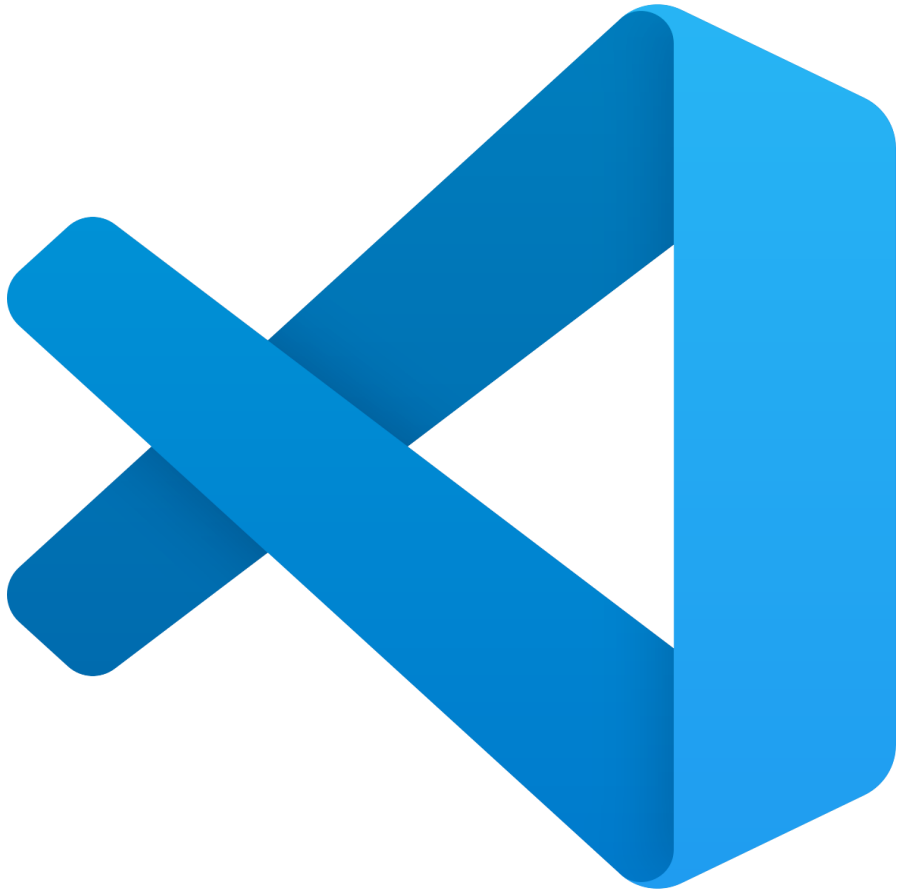
```
df.defineClass("df.WebColumn", "df.WebColumnBase",{

create : function(){
    this._sCellClass = "WebCol " + df.classWordBreak(this.peWordBreak);
},
```

02

WHY CONVERT?

WHY CONVERT?
READABILITY

- More readable code
- Editor features
  - Go to definition
  - Better code completion
- Familiar syntax



03

THE CONVERSION

THE CONVERSION
WHAT DID WE CONVERT

- WebApp Framework classes (Over 100 files!)
- Designer
- Navigation designer
- Df.defineclass
- Modules

THE CONVERSION
NEW CLASS SYNTAX

- Export
- Constructor keyword
- Functions defined in class
- Super keyword
- Df.defineclass no longer used

```
export class WebButton extends WebBaseControl {
    constructor(sName, oParent) {
        super(sName, oParent);

        //  Web Properties
        this.prop(df.tString, "psCaption", "");
        this.prop(df.tString, "psTextColor", "");
        this.prop(df.tString, "psWaitMessage", "");
        this.prop(df.tBool, "pbShowWaitDialog", false);
        this.prop(df.tInt, "peAlign", -1);

        //  Events
        this.event("OnClick", df.cCallModeWait);



        //  Configure super classes
        this.pbShowLabel = false;

        // @privates
        this._sControlClass = "WebButton";
        this._bJSSizing = false;
    }

    create() {
        super.create();

        this.set_pbShowWaitDialog(this.pbShowWaitDialog);
    }
```

THE CONVERSION
NEW MIXIN SYNTAX

- Mostly the same as a class
- Variable superclass
- Less work
- More readable

```
export const WebColumn_mixin = superclass => class extends superclass {
    constructor(sName, oParent) {
        super(sName, oParent);

        //  Assertions
        if (!(oParent && oParent instanceof WebList)) {
            throw new df.Error(999, "WebColumn object '{{0}}' should be placed ins
        }
```

```
//  Generate new base class using df.WebColumn_mixin and df.WebForm
export class WebColumnBase extends WebColumn_mixin(WebForm) {}
```

THE CONVERSION
VAR EXAMPLE

- Javascript has 3 types of variables
  - Var (Old!)
  - Let
  - Const
- Move away from var as much as possible
- Var is function scoped
- Hoisted
- Possible to assign before declaration

THE CONVERSION
VAR EXAMPLE

```
varExample() {
    myVariable = "Test";

    var myVariable;

    console.log(myVariable);
}
```

```
letExample() {
    myVariable = "Test"

    let myVariable;

    console.log(myVariable);
}
```

```
varExample() {

    for (var index = 0; index < 5; index++) {
        //Do something here

    }

    console.log(index);
}
```

```
letExample() {

    for (let index = 0; index < 5; index++) {
        //Do something here

    }

    console.log(index);
}
```

THE CONVERSION
LET OR CONST

- Use let for reassigning
- Use const for constant values
- Block scoped
- Not hoisted

	BLOCK SCOPED	TDZ	CREATES GLOBAL PROPERTY	REASSIGNABLE	REDECLARABLE
var	✗	✗	✓	✓	✓
let	✓	✓	✗	✓	✗
const	✓	✓	✗	✗	✗

THE CONVERSION
DOCUMENTATION

- Documentation updated
- Syntax updated to ES6
- Mixins
- Communication page updated
- Removed usage of tWebValuetrees

Mixins

This chapter describes how to use Mixins within the Web Framework.

Mixins are JavaScript classes that take another class as a parameter to inherit from.

```
/*
Mixin
*/
const Council_Mixin = superclass => class extends superclass {
    constructor(sName) {
        //  Call constructor of super class
        super(sName);

        this.bIsCouncil = True;
    }
    sayCouncil(){
        console.log("I am a council member!");
    }
}
```

04

WHAT DOES THIS MEAN FOR YOU?

WHAT DOES THIS MEAN FOR YOU?
CUSTOM CONTROLS

- Not too much!
- Backwards compatible
- Convert
- Do not use imports right now
- Modules are loaded async

- But how?

05

CSS VARIABLES

CSS VARIABLES
WHY?

- Modifying themes was tedious
- Multiple occurences
- Time consuming

```
/* Color Pallet

Main Color = Blue

    Dark            #005899
    Regular         #0072C6
    Light           #4DA8E8

Error Color = Red

    Dark            #971519
    Regular         #B72025

Warning Color = Yellow

    Dark            #E2961C
    Regular         #F4A11E

Highlight Color = Yellow

    Regular         #F4A11E

Grayscale

    Darkest (Black) #181818
                    #5C5C5C
                    #8E8A87
                    #BABABA
                    #D9D9D9
                    #E6E6E6
                    #F3F3F3
    Lightest (White)#FFFFFF

*/
```

CSS VARIABLES
NEW THEMES

- Root holds all variables
- Only one variable to change
- Variables are accessible in the browser
- Easily configurable

```css
/* Color Palette */
:root {
    /* Main Color = Blue */
    --df-MainDark: #005899;
    --df-MainRegular: #0072C6;
    --df-MainLight: #4DA8E8;

    /* Error Color = Red */
    --df-ErrorDark: #971519;
    --df-ErrorRegular: #B72025;

    /* Warning Color = Yellow */
    --df-WarningDark: #E2961C;
    --df-WarningRegular: #F4A11E;

    /* Highlight Color = Yellow */
    --df-Highlight: #F4A11E;

    /* Grayscale from darkest to lightest */
    --df-Grayscale1: #181818;
    --df-Grayscale2: #5C5C5C;
    --df-Grayscale3: #8E8A87;
    --df-Grayscale4: #BABABA;
    --df-Grayscale5: #D9D9D9;
    --df-Grayscale6: #E6E6E6;
    --df-Grayscale7: #F3F3F3;
    --df-Grayscale8: #FFFFFF;
}
```

06

FUTURE POSIBILITIES

FUTURE POSIBILITIES
JAVASCRIPT

- User configurable themes
- Extend themes with more variables
- Updated styler
- Implementing Webpack in build process
- Documenting important JS classes

QUESTIONS?