Studio Client Development with TypeScript and NPM

Modernizing and Standardizing CoreMedia's UI Development Technology Stack

COREMEDIA

AGENDA

Motivation

Converting to TypeScript/NPM

Ext JS vs. Modern JavaScript

Working with TypeScript/NPM

STUDIO DEVELOPMENT WITH JANGAROO 4

Jangaroo compiles Flex (AS/MXML) sources to Ext JS JavaScript

MODERNIZING STUDIO CLIENT DEVELOPMENT

Getting rid of ActionScript, MXML and Maven

> Flex (ActionScript, MXML)

  > Good choice in 2008

  > Bad choice in 2021 (declining since 2011)

> Maven

  > Good choice for Java

  > Bad choice for Web UI development

BETTER MIGRATE TO TYPESCRIPT!

WHY NOT GO WITH GENERATED JAVASCRIPT?

Jangaroo generates readable (Ext-)JavaScript code.
Why not continue using that as source code?

> Outdated JavaScript dialect (ECMAScript 3)

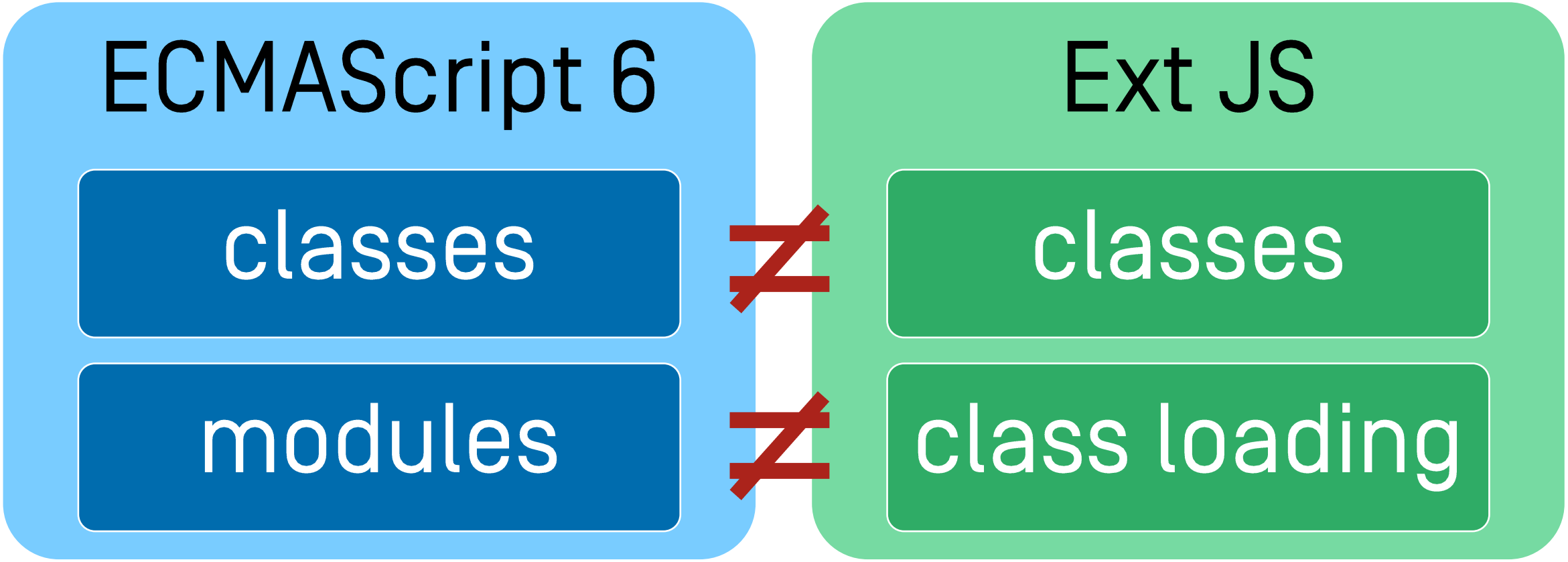
> We don't want to lose static typing

  > Comprehensive Ext API overwhelming and error-prone

  > Studio API is equally large & complex

  > Sencha IDE plugins are expensive and not as good

THE CHALLANGE(S)



SENCHA'S ES6 PLANS IN 2016

- Allow use of ES6 classes to develop Ext JS applications
- Use decorators and Babel to compile Ext JS class concepts that ES6 does not support, mainly
  - Configs
  - Mixins
- Special handling of constructor

STATE OF THE ART

- Nothing of this happened

- You can only use *other* ES6+ language extensions like arrow functions (supported since Sencha Cmd 6.x)
- However, these are *not* supported in ActionScript ☹

BABEL: COMPILING ECMASCRIPT

Babel is used to compile “modern” ECMAScript / JavaScript to JavaScript that also works in older environments (browser, Node.js)

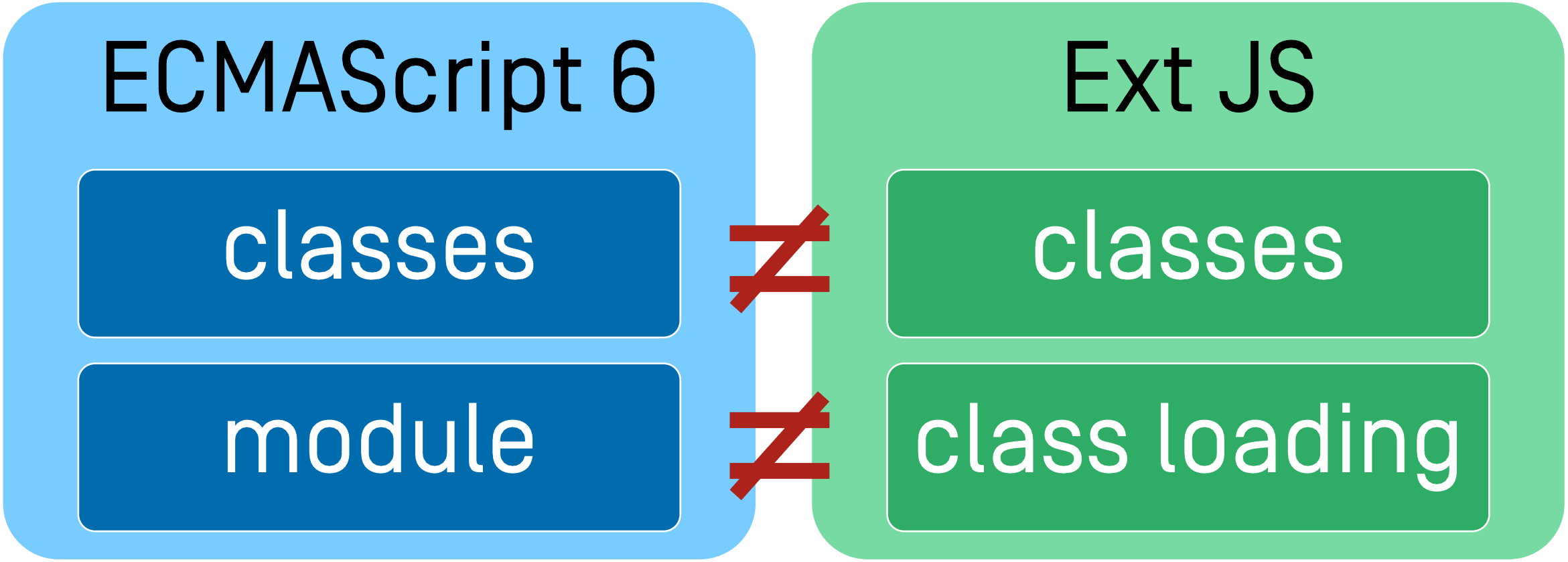
Babel plugins: plug into the code transformation chain

KEEP CALM

AND

BUILD YOUR OWN BABEL PLUGIN

THE CHALLANGE(S)



ECMASCRIPT 6 VERSUS EXT JS CLASSES

```
class MyComponent
  extends Container {

  static xtype = "mycomponent";
  static _ = class extends Container._ {
    title = "default title";
  }

  constructor(config) {
    config.items = computeItems(config);
    super(config);
  }

  static computeItems(config) { … }
}
```

```
Ext.define("MyComponent", {
  extend: "Ext.container.Container",

  xtype: "mycomponent",
  config: {
    title: "default title"
  },

  constructor: function (config) {
    config.items = computeItems(config);
    this.callSuper(config);
  },

  statics: {
    computeItems: function (config) { … }
  }
});
```

ES6 MODULES VS. EXT JS CLASS LOADING

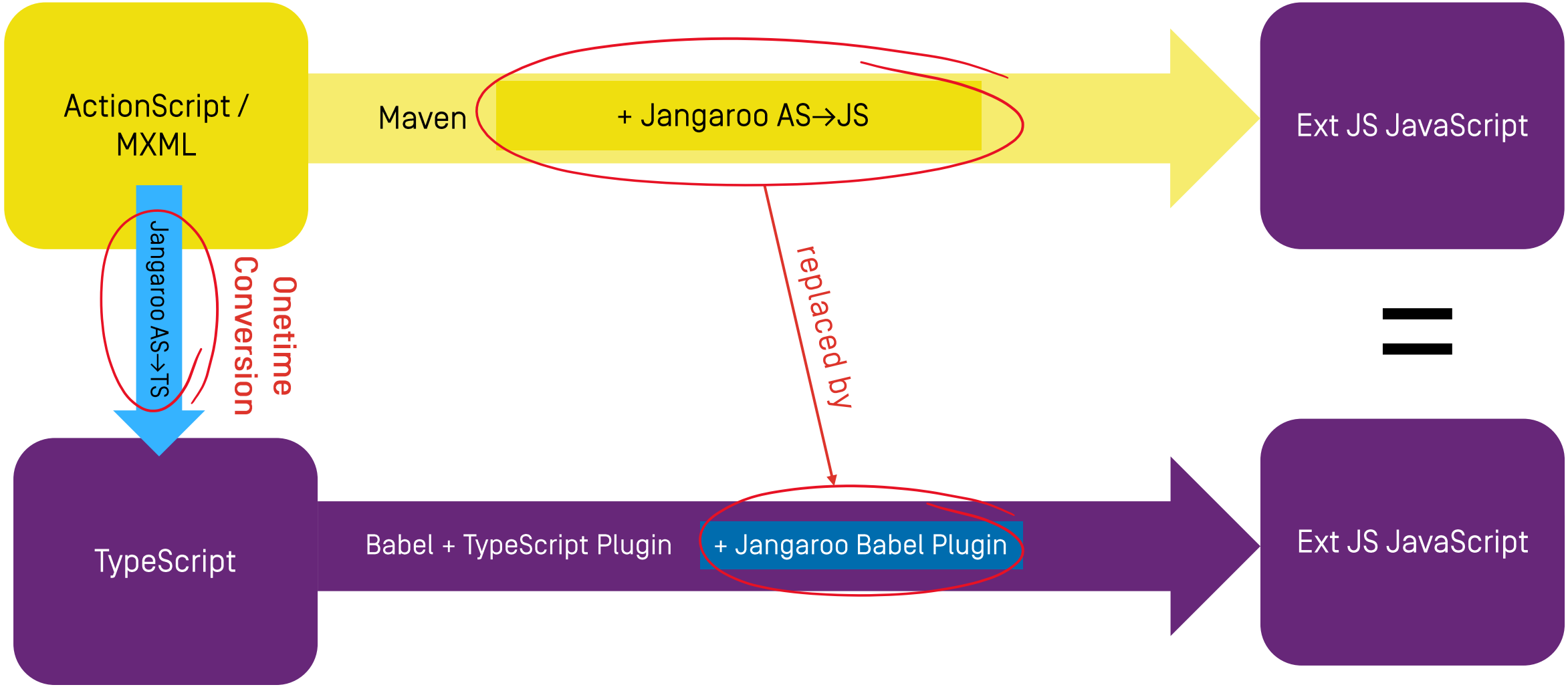
ES6 import/export versus Ext JS requires/define

```
import Toolbar from "ext-js/toolbar/Toolbar";
import Button from "ext-js/button/Button";

export default class MyToolbar
  extends Toolbar {

  items = [
    new Button._({
      text: "click me"
    })
  ];
}
```

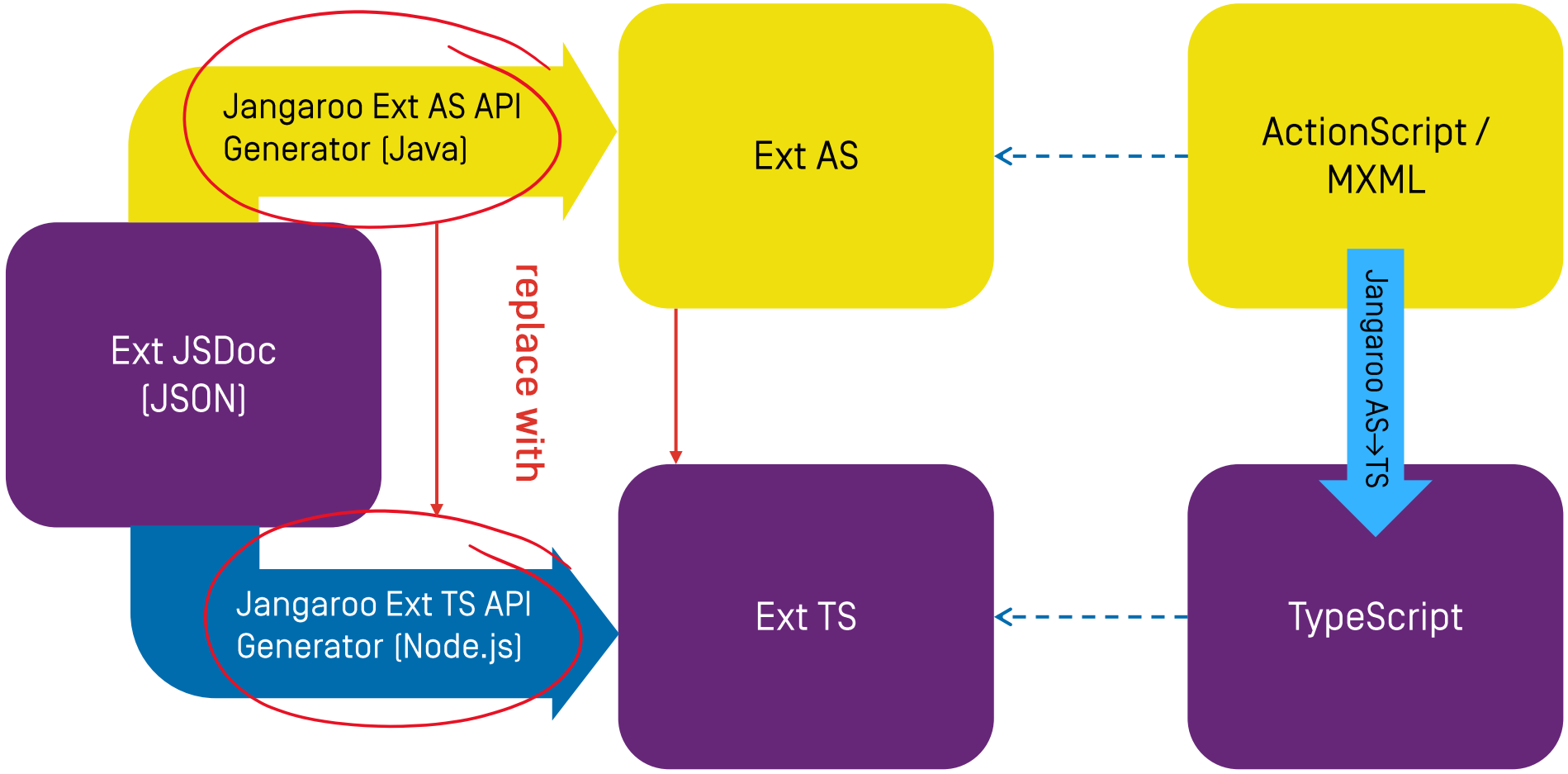
```
Ext.define("MyToolbar", {
  requires: [
    "Ext.toolbar.Toolbar",
    "Ext.button.Button"
  ],
  extend: "Ext.toolbar.Toolbar",

  items: [
    {
      xclass: "Ext.button.Button",
      text: "click me"
    }
  ]
});
```

JANGAROO IS DEAD, LONG LIVE JANGAROO!

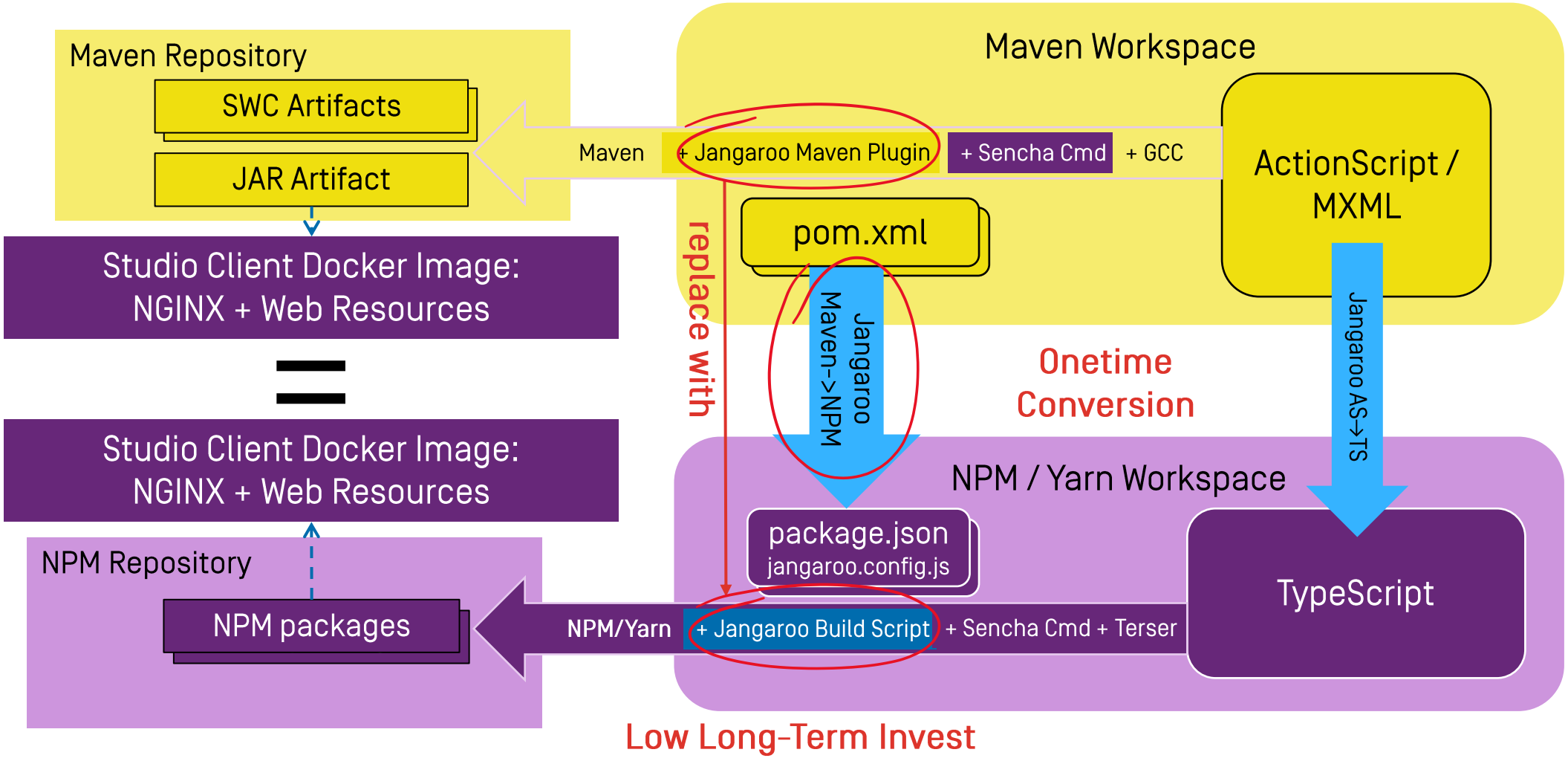
STUDIO COMPILER CHAIN



Ext AS -> Ext TS


Jangaroo Ext AS API Generator (Java)
Ext AS
ActionScript / MXML
Ext JSDoc (JSON)
replace with
Jangaroo AS→TS
Jangaroo Ext TS API Generator (Node.js)
Ext TS
TypeScript

MAVEN -> NPM


Maven Repository
SWC Artifacts
JAR Artifact
Studio Client Docker Image:
NGINX + Web Resources
=
Studio Client Docker Image:
NGINX + Web Resources
NPM Repository
NPM packages
Maven Workspace
Maven
+ Jangaroo Maven Plugin
+ Sencha Cmd
+ GCC
ActionScript /
MXML
pom.xml
replace with
Jangaroo
Maven->NPM
Jangaroo AS->TS
Onetime
Conversion
NPM / Yarn Workspace
package.json
jangaroo.config.js
TypeScript
NPM/Yarn
+ Jangaroo Build Script
+ Sencha Cmd + Terser
Low Long-Term Invest

OVERVIEW

Migration Tools

> Compiler AS3/MXML -> TypeScript

> Maven (pom.xml) -> NPM (package.json/jangaroo.config.js)

Replacements

> Compiler AS3/MXML -> Ext JS

> Ext AS

> Maven Plugin

> Babel Plugin TS->Ext JS

> Ext TS

> NPM Build Scripts + Babel Plugin

DEMO TIME!

POSSIBLE MANUAL MIGRATION EFFORTS

ToDos after applying the Migration Tool

- AS->TS
  - Tool cannot reliably tell Config type from class type
  - Slight differences between APIs
    - Ext AS -> Ext TS
    - jangaroo-browser -> TypeScript lib-dom
- pom.xml -> package.json / jangaroo.config.js
  - Using arbitrary Maven features => manual effort

TAKEAWAY: NEW WORKSPACE

- Based on standards: NPM, Yarn, Babel, TypeScript
  - No more Maven, ActionScript/MXML
- Works in many IDEs (IDEA, WebStorm, VSCode)
  - No more “IDEA Ultimate only”
- Watch task, incremental builds, debugging

TAKEAWAY: WORKSPACE CONVERSION

Converting your CM10 Studio Client workspace to CM11:

- Fully automated, usually builds & runs right after conversion
- Manual efforts for green TypeScript sources
- Maybe manual efforts for previously undetected errors
- Maybe manual effort for special cases ("native" JS code, 3rd-party JS integration, unusual Maven configuration)

THANK YOU!

Your questions?

Dr. Frank Wienberg
Software Architect

frank.wienberg@coremedia.com
Skype: frank.wienberg