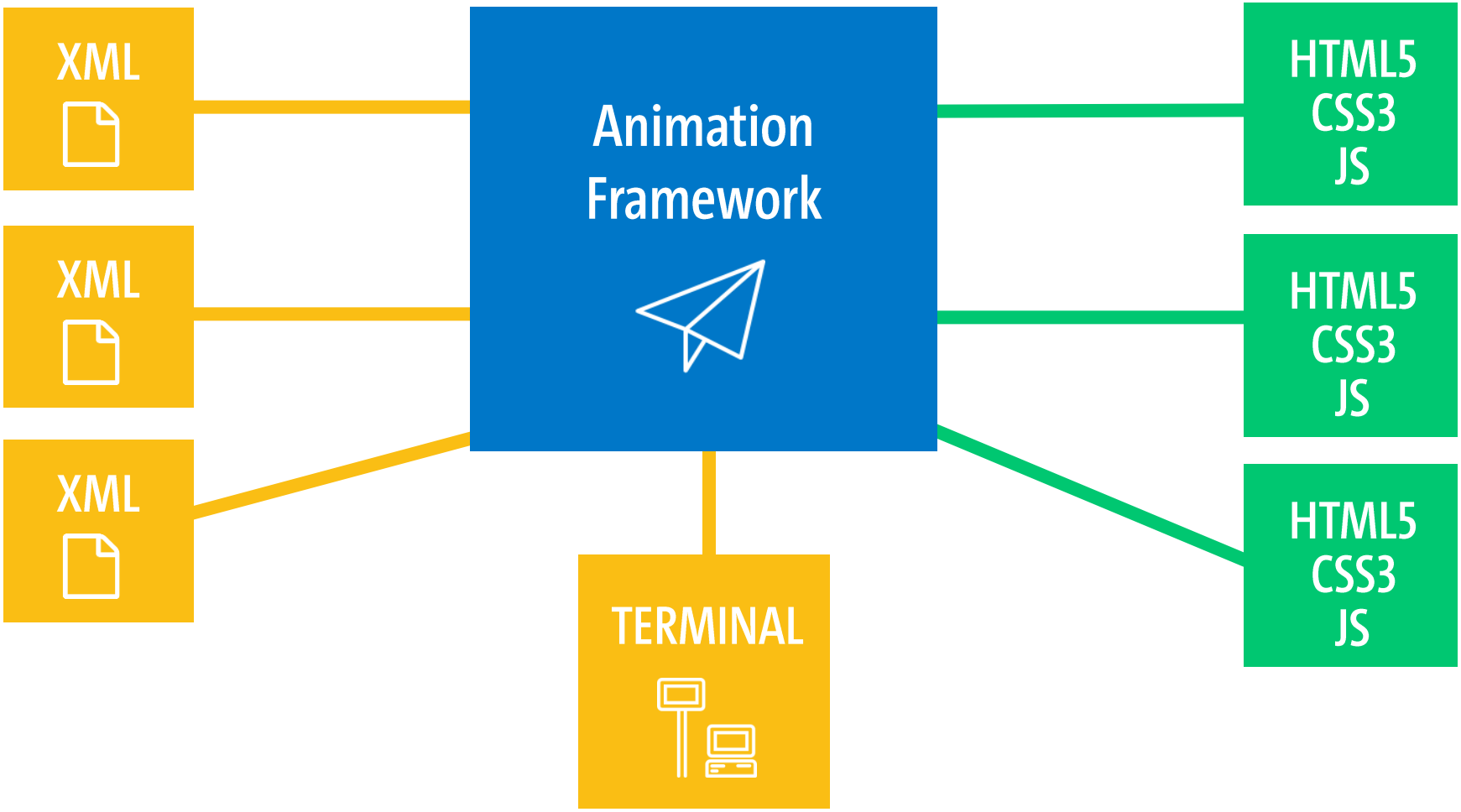
Liebst du Typen? JavaScript in großem Stil mit TypeScript

Johannes Dienst

Ein komplexes Projekt




JAVA DEVELOPER
J'raxis
THINKING JAVASCRIPT IS LIKE JAVA

JavaScript

- Dynamisch typisiert
- Vererbung mit Prototypen
- this ist nicht gleich this

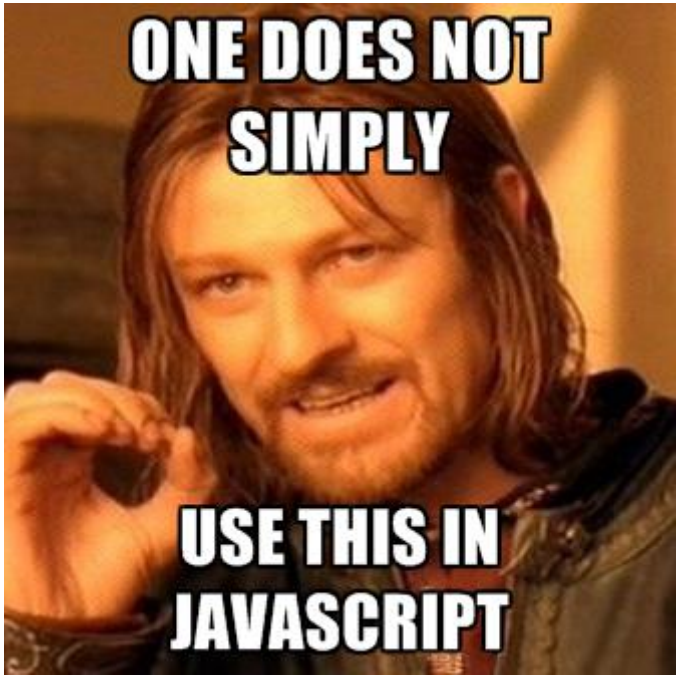
Dynamische Typisierung

```
function getDate() {
    var date = new Date();
    var dateArray = [];
    dateArray.push(date.getDay());
    dateArray.push(date.getMonth());
    dateArray.push(date.getFullYear());
    return dateArray;
}
```

Prototypen

```
var Monkey = (function (_super) {

    __extends(Monkey, _super);

    function Monkey(name) {
        _super.call(this, name);
    }

    Monkey.prototype.move = function () {
        alert("Hopping...");
        _super.prototype.move.call(this, 5);
    };

    return Monkey;

})(Mammal);
```

Das gefürchtete this



Das gefürchtete this: Ein Beispiel

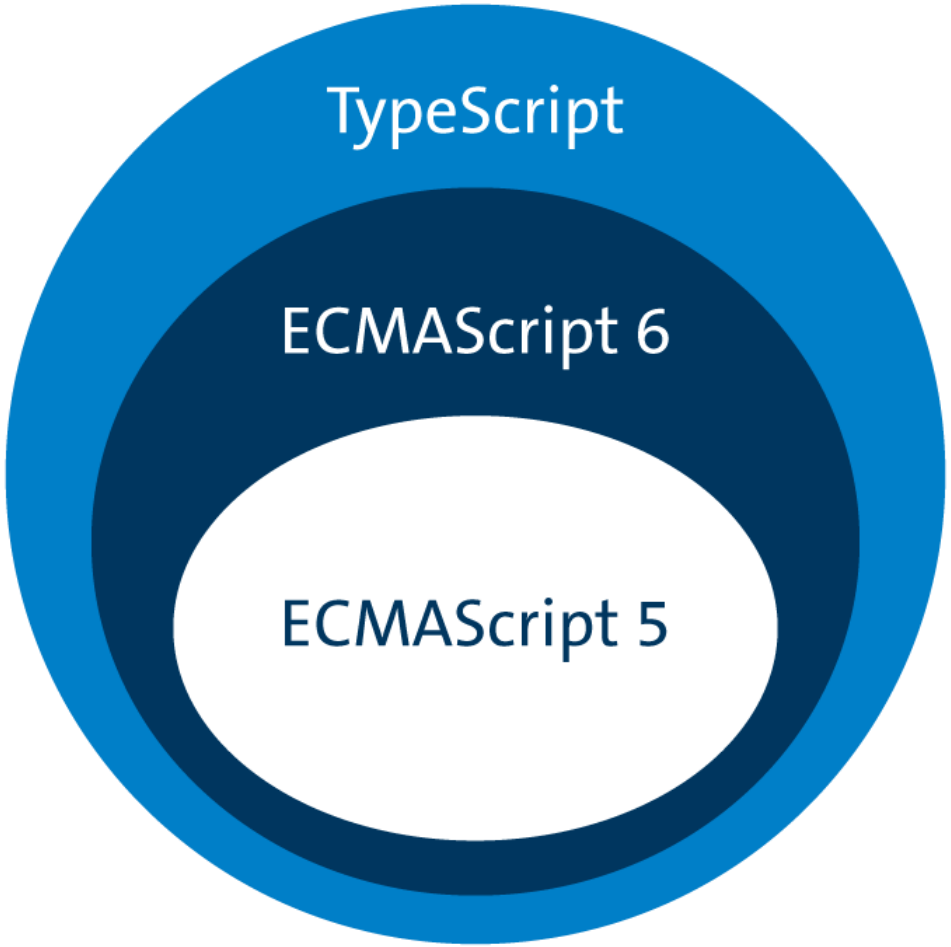
```
MovieManager.prototype.init = function(aMovieInstances) {

    var that = this;

    this._validator.validate(
        function() { that.setReady(); },
            aMovieInstances);
};
```

JavaScript ist cool

- Riesige Community
- Mächtige Frameworks
- Läuft praktisch überall

TypeScript

- Aktive Community
- Native Unterstützung von Frameworks
- Kompiliert zu JavaScript


TypeScript
ECMAScript 6
ECMAScript 5

JavaScript

- Dynamisch typisiert
- Vererbung mit Prototypen
- this ist nicht gleich this

Statische Typisierung

- Principle of Least Astonishment
- Interface Segregation Principle
- Komplexe Refaktorisierungen

Statische Typisierung

```
function getDate(): Array<number> {
    var date = new Date();
    var dateArray = [];
    dateArray.push(date.getDay());
    dateArray.push(date.getMonth());
    dateArray.push(date.getFullYear());
    return dateArray;
}


// Prüfung zur Compile-Zeit

var name: string = "Banana“;

name = 27; // Fehler!!
```

JavaScript

- Dynamisch typisiert
- Vererbung mit Prototypen
- this ist nicht gleich this

Klassen

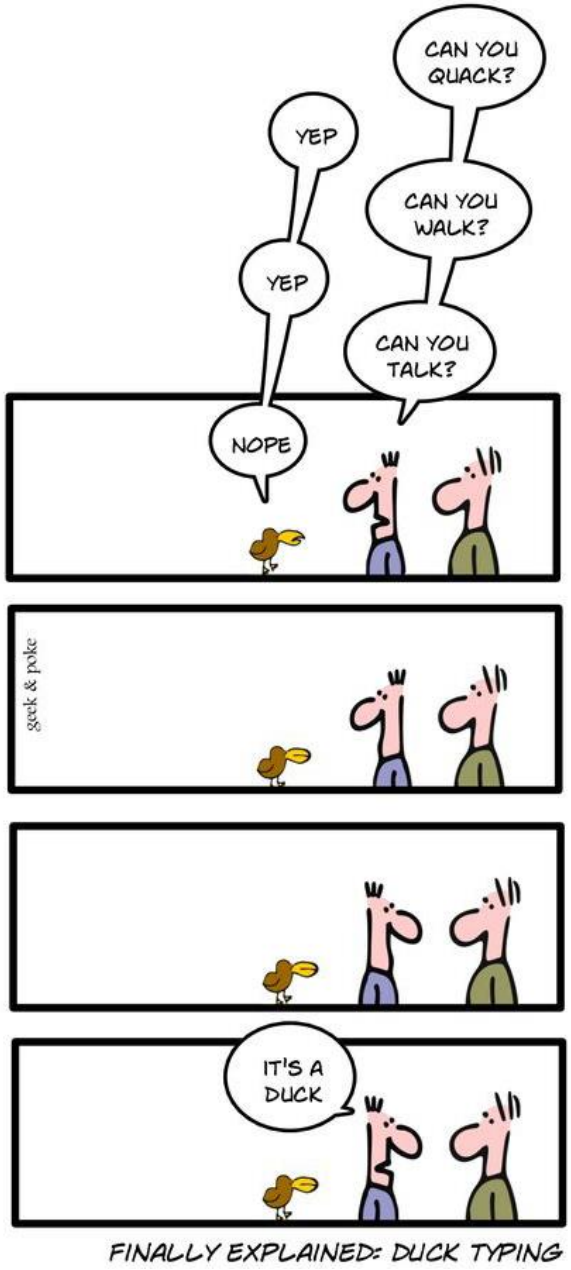
```
class Fish
{

    constructor(name: string) { }

    move(meters: number)
    {
        alert("Swimming... " + meters);
    }
}
```

Interfaces

```
interface Animal
{
    move(meters: number): void;
}

class Mammal implements Animal
{

    constructor(name: string) { }

    move(meters: number)
    {
        alert("Moving... " + meters);
    }
}
```

Klassen - Vererbung

```
class Monkey extends Mammal
{

    constructor(name: string) { super(name); }

    move()
    {
        alert("Hopping...");
        super.move(5);
    }
}
```



Quelle: http://geek-and-poke.com/

Klassen – Structural Typing

```
class Horse
{

  constructor(public name: string) { }

  move(meters: number)
  {
    alert("Galloping...");
    alert(this.name + " moved " + meters + "m.");
  }
}
```

Klassen - Instanziieren

```
var sam = new Monkey("Sammy the Monkey");
var tom: Mammal = new Horse("Tommy the Palomino");

sam.move();
tom.move(34);
```

JavaScript

- Dynamisch typisiert
- Vererbung mit Prototypen
- this ist nicht gleich this

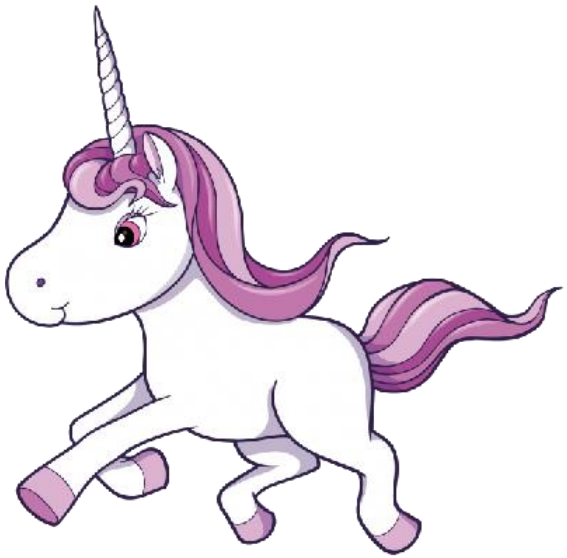
Sofortige Bindung von this

```
init(aMovieInstances: Array<any>)
{

    this._validator.validate(
        ()=> { this.setReady(); },
        aMovieInstances);
    );

}
```

Und das Projekt

Fazit

- Statische Typisierung
- Klassen und Interfaces
- this mit früher Bindung



Danke

JohannesDienst

jdienst@multamedio.de

JohannesDienst.net