



metaUI



Please have a brief
look at the **form**
on the right ...

General

Title

First Name

Last Name

Age

Profession ▼

Company size

☐ < 10

☐ > 10 < 50

☐ > 50

Save



Each field is
represented by a
**bunch of
components.**

Title



```
<field-group>
```

```
  <field label="First Name">
```

```
    <field-input [formControl]="firstName">
```

```
  </field-input>
```

```
  <field-error [control]="firstName">
```

```
  </field-error>
```

```
</field>
```

```
</field-group>
```



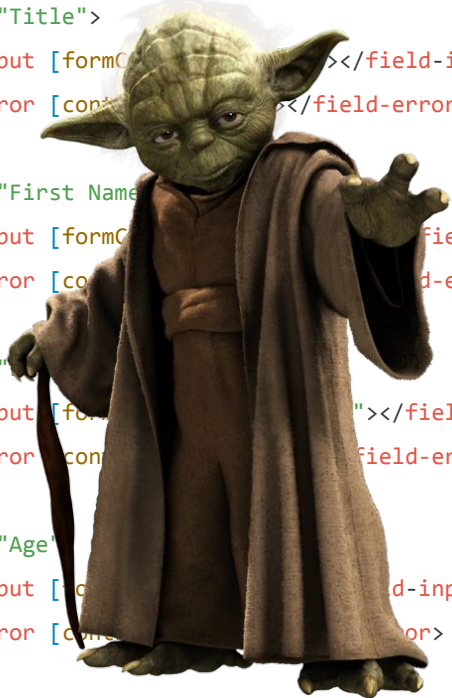
The more
components the
more lines of code
you have to maintain.

```
<field-group label="General">
  <field label="Title">
    <field-input [formControl]="title"></field-input>
    <field-error [control]="title"></field-error>
  </field>
  <field label="First Name">
    <field-input [formControl]="firstName"></field-input>
    <field-error [control]="firstName"></field-error>
  </field>
  <field label="Last Name">
    <field-input [formControl]="lastName"></field-input>
    <field-error [control]="lastName"></field-error>
  </field>
  <field label="Age">
    <field-input [formControl]="age"></field-input>
    <field-error [control]="age"></field-error>
  </field>
  <field label="Profession">
    <field-combobox [formControl]="profession"></field-combobox>
    <field-error [control]="profession"></field-error>
  </field>
</field-group>
```



Scrolling down the
code is like watching
a Star Wars Intro.

```
<field-group label="General">
  <field label="Title">
    <field-input [formControl]="title"></field-input>
    <field-error [control]="title"></field-error>
  </field>
  <field label="First Name">
    <field-input [formControl]="firstName"></field-input>
    <field-error [control]="firstName"></field-error>
  </field>
  <field label="Last Name">
    <field-input [formControl]="lastName"></field-input>
    <field-error [control]="lastName"></field-error>
  </field>
  <field label="Age">
    <field-input [formControl]="age"></field-input>
    <field-error [control]="age"></field-error>
  </field>
  <field label="Profession">
    <field-combobox [formControl]="profession"></field-combobox>
    <field-error [control]="profession"></field-error>
  </field>
</field-group>
```





15 form fields
produce around
100 lines of code.



What impact would it have if you...

...add **i18n**?



What impact would it have if you...
...deal with **role concept**?



What impact would it have if you...
...deal with **dynamic layouts**?



10 domain objects

→ 3 pages for each

→ 30 pages

→ over 3000 lines of code

Gregor Woiwode

Software Architect



co-IT.eu GmbH



gregor.woiwode@co-IT.eu



@GregOnNet

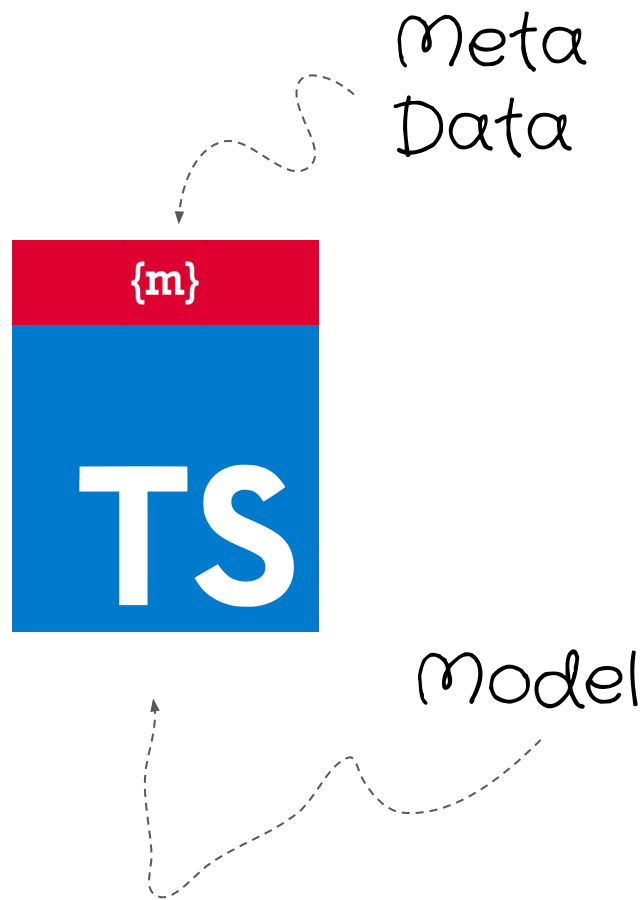




metaUI

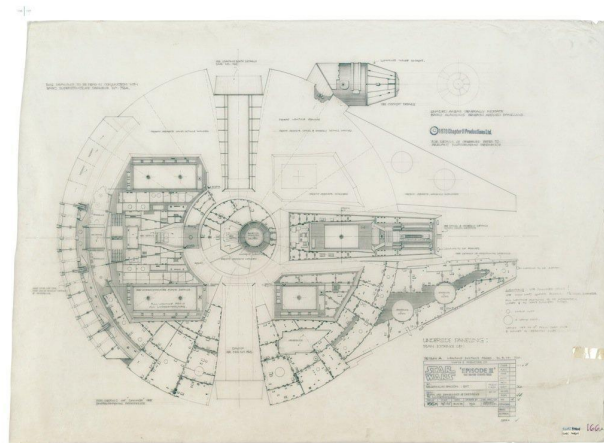


metaUI uses your
model + Meta
data describing
the object





metaUI is a set of
APIs and
components





metaUI derives the
whole UI for you
without templates.

SAP Home

Overview Sales Resource Management Self Services Analytics and Reporting

Featured

Recent Activities 3 of 6

- Search Sales Performance 1 Day ago App
- My Accounts 1 Day ago App
- Open Opportunities 1 Day ago App

10 Years of Sustainability at PumpTech

Weather Forecast
Orlando, Florida
80.6°F

My Work Environment

Sales Orders Predicted Delayed 3 of 3

| Sales Order | Predicted Delay | Customer | Amount |
|-------------|-----------------|-------------------|-------------|
| 03493834 | 5 Days | Worlka Industries | \$15,843.23 |
| 04547903 | 3 Days | Acme Corp. | \$73,111.35 |
| 04573620 | 1 Day | Stark Industries | \$19,230.29 |

Approve 7 % Discount for Quote Pump MultiEco 331 [Approve](#)
from Anna Smith, Account DelBont Industries

| Product | Unit | Amount |
|-----------------------|------|-------------|
| Pump MultiEco 331 | 5 PC | \$22,785.00 |
| O-ring Pressure Seals | 5 PC | \$1,111.35 |
| Inlet Filter | 5 PC | \$4,180.35 |

My Accounts
189 Active

Take Courses
8

My Groups
Slack

Project Beta Costs
Me, Ann, John + 3 more Tue Feb 12

Team Alfa
Me, Ann, John + 8 more Tue Feb 5

Holiday Destinations
Me, Ann, Jessica Tue June 5



metaUI reduces
the amount of
code you have to
write by 70%.

```
<m-context  
  [object]="object"  
  [operation]="operation"  
  layout="two-column">  
  
  <m-include-component></m-include-component>  
  
</m-context>
```




```
<field-group label="General">
  <field label="Title">
    <field-input [formControl]="title"></field-input>
    <field-error [control]="title"></field-error>
  </field>
  <field label="First Name">
    <field-input [formControl]="firstName"></field-input>
    <field-error [control]="firstName"></field-error>
  </field>
  <field label="Last Name">
    <field-input [formControl]="lastName"></field-input>
    <field-error [control]="lastName"></field-error>
  </field>
  <field label="Age">
    <field-input [formControl]="age"></field-input>
    <field-error [control]="age"></field-error>
  </field>
  <field label="Profession">
    <field-combobox [formControl]="profession"></field-combobox>
    <field-error [control]="profession"></field-error>
  </field>
</field-group>
```



```
<m-context
  [object]="object"
  [operation]="operation"
  layout="two-column">

  <m-include-component></m-include-component>

</m-context>
```



Most of the IO code written in traditional frameworks is a mechanical application of (unstated) rules rooted in the domain object data model.”

— Craig Federighi



Object Style Sheet (OSS) describes your UI.

```
class=PersonalInfo {  
  field=title {  
    label: "Title";  
  }  
  
  field=firstName {  
    label: "First Name";  
  }  
}
```



```
.box {  
  .shape {  
    background-color: #9d9d9d;  
  }  
}
```

SASS

```
class=PersonalInfo {  
  field=title {  
    label: "Title";  
  }  
  
  field=firstName {  
    label: "First Name";  
  }  
}
```

OSS



Features



Dry Principle



Customization capability



UI Library agnostic

The background features three large, abstract geometric shapes. A teal shape is in the top-left corner, a purple shape is in the top-right corner, and a light blue shape is in the bottom-right corner. All shapes have rounded corners and are set against a white background.

Reviewing the
basic approach...

Entity

+

Rule

Entity

```
import { Entity } from '@ngx-metaui/rules';

export class Character implements Entity {

  className(): string {
    return 'Character';
  }

  identity(): string {
    /* ... */
  }

  getTypes(): any {
    return {
      /* ... */
    };
  }
}
```


Entity

```
import { Entity } from '@ngx-metaui/rules';  
  
export class Character implements Entity {  
  
  constructor(  
    public id: string  
  ) {}  
  
  getTypes(): any {  
    return {  
      id: String  
    };  
  }  
}
```

Needed for Introspection

Rule

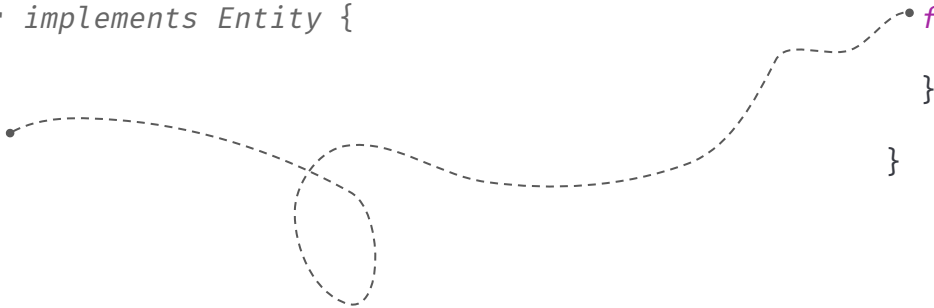
```
class=Character {  
  field=id {  
    label:"Id";  
  }  
}
```

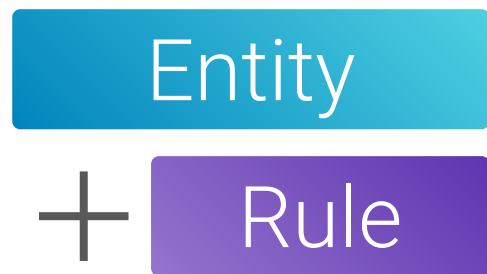
Entity

Rule

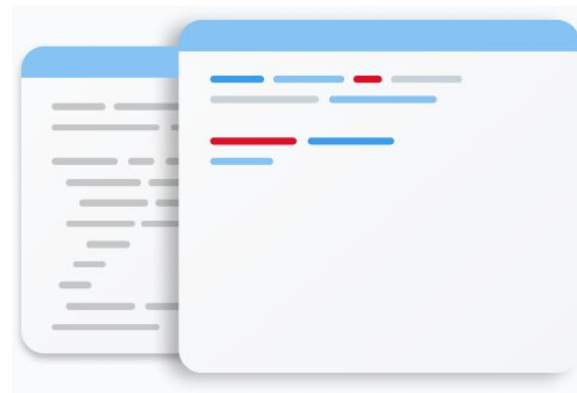
```
import { Entity } from '@ngx-metaui/rules';  
  
export class Character implements Entity {  
  
  constructor(  
    public id: string  
  ) {}  
  
  getTypes(): any {  
    return {  
      id: String  
    };  
  }  
}
```

```
class=Character {  
  field=id {  
    label:"Id";  
  }  
}
```





Rule
Processor





Rule Processor

- Knows about types
- Knows how types are rendered

```
import { Entity } from  
'@ngx-metaui/rules';
```

```
export class Character implements
```

```
  constructor(  
    public id: string  
  ) {}
```

```
  getTypes(): any {  
    return {  
      id: String  
    };  
  }  
}
```

```
}
```

```
<m-context
```

```
  [object]="entity"
```

```
  [operation]="operation"
```

```
  layout="two-column">
```

```
<m-include-component></m-include-component>
```

```
</m-context>
```



Rule processor



Entity

```
<m-context
```

```
  [object]="entity"
```

```
  [operation]="operation"
```

```
  layout="two-column">
```

```
  <m-include-component></m-include-component>
```

```
</m-context>
```



Setup
metaUI



```
ng add @ngx-metaui/rules
```




Setup metaUI

```
@NgModule({  
  declarations: [AppComponent],  
  imports: [  
    BrowserModule,  
    BrowserAnimationsModule,  
    AppRoutingModule,  
  
    MetaUIRulesModule.forRoot(),  
    MaterialRulesModule.forRoot()  
  ],  
  bootstrap: [AppComponent]  
})  
export class AppModule {}
```

The background features three large, abstract geometric shapes: a teal triangle in the top-left, a purple triangle in the top-right, and a blue triangle in the bottom-right. The text is positioned on the left side of the slide.

Review **1st**
refactoring...



OSS
field

```
field=avatarUrl {  
  label:"Profile Url";
```

```
}
```



OSS

field - Link your Component

```
field=avatarUrl {  
  label:"Profile Url";  
  component: ImageComponent;  
  
}
```



OSS

field - Bind component values

```
field=avatarUrl {  
  label:"Profile Url";  
  component: ImageComponent;  
  bindings: {  
    url: $value;  
    alt: ${object.firstName + ' ' + object.lastName + ' avatar image'};  
  };  
}
```



OSS

@field - Create computed properties

Entity

```
@field=fullName {  
  type: String;  
  value: ${object.firstName + ' ' + object.lastName};  
}
```

```
export class Character  
  implements Entity  
{  
  constructor(  
    public firstName: string,  
    public lastName: string  
  ) {}  
}
```



OSS

trait - Build reusable rules

Rule

```
@field=fullName {  
  trait: derived, heading1;  
  type: String;  
  value: ${object.firstName + ' ' + object.lastName};  
}
```

```
@trait=heading1 {  
  wrapperComponent:GenericContainerComponent;  
  wrapperBindings: { tagName:h1; };  
}
```

[Source](#)



OSS

trait - Build reusable rules

Traits behave like mixins you
know from *Sass* .

The background features three large, abstract geometric shapes: a teal triangle in the top-left, a purple triangle in the top-right, and a blue triangle in the bottom-right. The text is positioned on the left side of the slide.

Review **2nd**
refactoring...



OSS

operation - Define your view

```
class=Character {  
  operation=(view) {  
    zNone => *;  
    zLeft => fullName => avatarUrl;  
  }  
  
  operation=(edit) {  
    @field=title {  
      trait: derived, heading1;  
      type: String;  
      value: 'Edit';  
    }  
  
    zNone => *;  
    zLeft => title => firstName => lastName => lightsaberColor;  
  }  
  
  /* add as many operations you like */  
}
```



OSS

operation - Define your view

```
class=Character {  
  operation=(view) {  
    zNone => *;  
    zLeft => fullName => avatarUrl;  
  }  
}
```



Awaiting **3rd**
refactoring... :-)



MetaUI TV



github.com/GregOnNet/metaui-playground



Thank you

www.metaui.io