



# i18n

## in großen Angular Projekten

TypeScript



## **Philipp Burgmer**

<> [burgmer@w11k.de](mailto:burgmer@w11k.de) | [@philippburgmer](https://twitter.com/philippburgmer)

Software-Entwickler, Trainer

Web-Technologien, Sicherheit

TypeScript, Angular </>

## **W11K GmbH - The Web Engineers**

- <> Gegründet 2000
- Auftrags-Entwicklung / Consulting
- Web / Java
- Esslingen / Siegburg </>

## **theCodeCampus.de - Weiter.Entwickeln.**

- <> Schulungen seit 2007
- theCodeCampus seit 2013
- Projekt-Kickoffs & Unterstützung im Projekt </>

- <> Was ist Internationalisierung
- <> Funktionale & Nicht-Funktionale Anforderungen
- <> Verschiedenen Ansätzen
- <> Überblick & Ideen geben
- <> Große Projekte
- <> Angular Projekte

If you internationalize, you develop your application in a way that ensures it will work well and can be easily adapted for users from any culture, region, or language.

W3C

# Was ist l10n?

## **Internationalization**

<> Anwendung so entwickeln, dass sie angepasst werden kann </>

## **Localization**

<> Tatsächliche Anpassung an eine Sprache / Kultur

Setzt i18n voraus </>

# Nicht Funktionale Anforderungen

<> Datenformate

<> Performance

<> Modularisierung / Wiederverwendbarkeit

<> Gesetze & Restriktionen

# Funktionale Anforderungen

- <> Formatierungen anpassen
- <> Texte übersetzen
- <> Grafische Darstellung anpassen
- <> APIs



# APIs

- <> Anforderung: Navigation in App einbauen
- <> Kein Problem, Google Maps sei Dank
- <> Neue Anforderung: Nach Asien expandieren

- <> Google Maps in China nicht verfügbar
- <> Baidu Maps → andere API
- <> LeafLet als Zwischenschicht
- <> Einheitliche API für verschiedene Content-Provider

# i18n

ist also mehr als nur ...

# Formattierungen

- <> Wichtig: von Anfang an richtig machen**
  - Konsequent sein, kein Mischmasch
  - Oft Kleinigkeiten, die kaum auffallen
  - Intuitiv sein, nicht Gewohnheiten ausnutzen
- <> Englische Texte, aber ...**
  - Deutsche Zahlen
  - Deutsches Datum

<> Tausender- und Dezimal-Trennzeichen

<> Position des Währungszeichens

<> Angular

- NumberPipe, CurrencyPipe, PercentPipe
- Benutzen [ECMAScript Internationalization API](#) (Polyfill: [Intl.JS](#))
- `lang`-Attribute z.B. an `html` setzen oder per Code
- Wird nur ausgeführt wenn Zahl sich ändert
- Sprache kann zur Laufzeit geändert werden

```
1 <div>{{ pi | number:'1.1-5' }}</div>
```

<> Reihenfolge der Elemente

<> Anpassung an Zeitzone

<> Kalendersystem (z.B. für Nordkorea oder Indien)

<> Angular

- DatePipe
- Input: Date-Object, ISO-String, number
- Konfiguration: ShortCuts oder Pattern
- Nur Gregorianischer Kalender

```
1 <div>{{ today | date:'shortDate' }}</div>  
2 <div>{{ today | date:'dd.MM.yyyy' }}</div>
```



<> Beim Sortieren von Datensätzen lokale Besonderheiten beachten

- z.B. Umlaute in Deutschland

<> Angular

- Keine orderBy-Pipe mehr in Angular 2+
- Manuelles Sortieren im Code
- `Collator` aus Internationalization API verwenden

```
1 var strings = ["a", "c", "ä", "b"];
2 strings.sort(); // ["a", "b", "c", "ä"]
3 strings.sort(new Intl.Collator("de").compare) // ["a", "ä", "b", "c"]
```

# Texte übersetzen

<> ID einbetten

<> Wird durch Text ersetzt

```
1 <div translate>common.yes</div>  
2 <a title="{{ 'title' | translate }}">Link</a>
```

<> Wiederverwendung bei allgemeinen Wörtern üblich

- Setzt nur ein Wort für 'Ja' in allen Sprache voraus
- Kontextabhängig?

- <> Text einbetten (z.B. Englisch), wird per Attribute markiert
- <> ID wird generiert
- <> Keine Wiederverwendung
- <> Zum Übersetzen werden original Texte und IDs benötigt
  - Text steht im Code → schlecht für Übersetzer
  - Wann wird ID generiert

```
1 <div i18n>Yes</div>
```

<> ID steht im Code

<> Übersetzungen stehen in Datei / Datenquelle

- Mapping ID zu Text

<> ID wird zur Laufzeit durch Text ersetzt

- Vorteil: Wechseln der Sprache
- Nachteil: Performance Overhead
- Wie oft wechselt ein realer Benutzer die Sprache?

<> Angular

- ngx-translate

```
1 <h1><span translate>Hello</span> Karlsruhe</h1>
```

<> Code wird von Tool geparsed

- Texte werden extrahiert, IDs generiert → in Datei gespeichert

<> Texte werden übersetzt

<> Beim Build wird eine lokalisierte Version pro Sprache gebaut

- Deployment: my-app.com/es

<> Angular

- Angular i18n

```
1 <h1><span i18n>Hello</span> Karlsruhe</h1>
```

# Demo

- <> Oft Entwickler-Formate wie Java-Property- oder JSON-Dateien
- <> Übersetzer arbeiten mit eigenen Tools → andere Formate
  - XLIFF, XMB oder GetText weit verbreitet und standardisiert
  - Mehr Features: z.B. Hinweise zum Kontext

```
1 <h1><span i18n="Greet the Audience">Hello</span> Karlsruhe</h1>
```



- <> Satzbau und Reihenfolge von Wörtern nicht immer gleich
- <> Platzhalter und Interpolation benötigt
- <> Angular
  - Über reguläre Interpolation für Datenzugriff

```
1 <h1 i18n="Greet the Audience">Hello {{ city }}</h1>
```

## <> Plural wird in machen Sprachen anders gebildet

- Nicht nur Unterscheidung zwischen 0, 1 und >1
- Wann werden Zahlen und wann deren Wörter ausgegeben?

## <> Notation für Auswahl benötigt

## <> Angular

- ICU Format

```
1 <h1 i18n="Greet the Audience">
2   {listeners, plural,
3     =0 {Hello to myself}
4     =1 {Hello, I'm glad you're here}
5     other {Hello}
6   }
7 </h1>
```

## <> Auswahl aus Alternativen

- z.B. Geschlecht
- Sehr schwierig vorausszusagen was welche Sprache braucht
- Iterativ mit Übersetzern arbeiten

```
1 <div i18n>  
2   It's a {gender, select, m {male} f {female}}  
3 </div>
```

## <> Fehlermeldungen sollten ebenfalls übersetzt werden

- Mapping von technischen Bezeichnern (z.B. HTTP Status)

## <> Problem bei Angular i18n

- Unterstützt derzeit nur Text im Template
- Keine Übersetzung im TypeScript Code ([Issue](#))

## <> ngx-translate: Service

```
1 export class GreetingComponent {
2   constructor(translate: TranslateService) {
3     translate.get('HELLO', {value: this.city}).subscribe((res: string) => {
4       console.log(res); //=> 'Hello Karlsruhe'
5     });
6   }
7 }
```

# Modularisierung

## <> Dopplungen vermeiden

- `de` und `de_at`
- Vererbung
- `de_at` enthält nur was speziell ist

## <> In Java üblich

## <> In Angular: bisher nicht umgesetzt

- Können eventuell Übersetzungstools abbilden
- ngx-translate: beim Laden der Daten

- <> Angular Anwendungen werden in Feature-Modulen organisiert
- <> Übersetzungen sind derzeit ein Monolith
- <> ngx-translate: Merge beim Laden theoretisch möglich
- <> Angular i18n: derzeit nicht möglich (Issue)

# Graphische Anpassungen



## <> Zunehmend Web-Fonts im Einsatz

- Enthaltene Zeichen beachten
- Font-Family beachten
- Lokalisierte Anwendung intensiv testen (lassen)

## <> Farben haben unterschiedliche Bedeutungen

## <> Icons und Emojis sollten angepasst sein (z.B. Hautfarbe)

## <> Unterschiedliche Darstellung von Sonderzeichen (Anführungszeichen)

- « Bonjour Karlsruhe »

```
1 <q lang="fr">Bonjour Karlsruhe</q>
```

```
1 q:lang(fr) { quotes: '\00AB\2005' '\2005\00BB'; }
```

<> Nicht alle Sprachen werden von links nach rechts geschrieben

- Arabisch von rechts nach links
- Chinesisch teilweise von oben nach unten

<> In HTML schon eingebaut: `dir` Attribute mit `ltr` oder `rtl`

- Achtung: CSS Regeln teilweise angepasst (`text-align`)
- Teilweise nicht (`margin`)

```
1 <html lang="en" dir="ltr"></html>
```

<> UI auf unterschiedlich lange Texte vorbereiten

<> CSS Selektoren auf Attribute wie `[dir="rtl"]` und `[lang="zh"]`

- Ränder und Abstände vertauschen
- Elemente vertauschen / anders platzieren
- Achtung: potenziell langsam

<> Flex-Box

- `justify-content: flex-start` und `align-content: flex-end`
- Statt `float: left` und `float: right`

**<> i18n kann deutlich mehr enthalten als man annimmt**

- Anpassungen auf vielen Ebenen
- Weitere Bereiche: z.B. Validierung

**<> Angular**

- Angular i18n sehr interessant
- Wenn keine High-End-Performance lieber ngx-translate

Philipp Burgmer  
burgmer@w11k.de  
Twitter: @philippburgmer  
GitHub: pburgmer

[www.thecodecampus.de](http://www.thecodecampus.de)  
@theCodeCampus