
Angular 2
Off Tracks
by Gerard Sans (@gerardsans)
Laptops shiny and ready by **9:00**~ish
Instructions **bit.ly/jfokus-ng2-off-tracks**
Jfokus
2017

Agenda

9:00 Welcome

Instructions and setup

Advanced Techniques

Practice, Q&A

Angular 2 Animations

Practice, Q&A

Redux and using ngrx/store

Practice, Q&A

Ahead of time Compilation

Practice, Q&A

12:30 Wrapping up

Google Developer Expert

Google Developers Experts Writer Edit

Blogger

gsans

Coding is fun | Coded something awesome today? | #Angular & #JavaScript are the new kings! M...

Aug 14 · 8 min read

GraphQL and the amazing Apollo Client

Explore an Application built using React and Angular 2

International Speaker


NG
CONF


AC
ANGULAR
CONNECT


Riga
DevDay


FullStack
The conference on JavaScript, Node and Internet of Things
13–15 JUL
2016


DON'T MISS:
droidcon
LONDON
27-28 OCT
2016
LONDON


VOXXEDDAYS
BELGRADE


js.talks();


QCon
LONDON


Jfokus


INTERNET &
MOBILE
WORLD 2016


{ }
Frontend
Union
Conf


ff


JAZOON
TechDays


GDG DevFest
The Netherlands


GDG DevFest
Ukraine 2016


GDG DevFest
London 2016


Angular 2 Trainer

Community Leader

New Features since final

Angular Version

index.html

```
// index.html
<my-app ng-version="2.4.6">
  <div>
    <h2>Hello Angular2</h2>
  </div>
</my-app>
```

Programatically

```
import {VERSION} from '@angular/core';

console.log(VERSION);
// {
//  full: "2.4.6"
//  major: "2"
//  minor: "4"
//  patch: "6"
// }
```

Components inheritance

Introduction

- Derived Class inherits from Base Class
- This complements ES6 inheritance
- Similarly to DI Configuration is done automatically
- The derived class can decide to override some behaviours

Base Class Propagates

- **Decorators** (metadata)
  - class (@Component, @Directive, @Pipe)
  - properties (@Input, @Output)
- **Constructor**: if not defined
- **Lifecycle hooks**: called even if not defined in derived class

Usage

```
<div>
  <person name="John"></person>
  <employee name="Tom" id="45231"></employee>
</div>
```

Implementation

```
@Component({
  selector: 'person',
  template: `<h4>Person: {{name}}</h4>`
})
export class Person {
  @Input() name: string;
}

@Component({
  selector: 'employee',
  template: `<h4>Employee: {{name}}, id: {{id}}</h4>`
})
export class Employee extends Person {
  @Input() id: string;
}
```

Advanced Techniques

Host and Child Elements

Introduction

- Available for Components and Directives
- Access to DOM Elements
  - using ElementRef
  - using @ViewChild, @ContentChild
- @Component, @Directive Metadata
  - host, @HostBinding, @HostListener

ElementRef

```
@Component({
  selector: 'person', // <person name="John"></person>
  template: `<h4>Person: {{name}}</h4>`
})
export class Person {
  constructor(private element: ElementRef) {
    console.log(this.element.nativeElement);
  }
}
```

@ViewChild

```
@Component({
  selector: 'section',
  template: `
    <div>
      <a href="#" #link>Add</a>
      <person name="John></person>
    </div>`
})
export class Section {

    @ViewChild('link') element;
    ngAfterViewInit() {
      //this.element.nativeElement
    }

    @ViewChild(Person) person;
    ngAfterViewInit() {
      //this.person.sayHi();
    }
}
```

@ContentChild

```
@Component({
  selector: 'my-container',
  template: '<ng-content></ng-content>'
})
export class MyContainer {
  @ContentChild('link') element;
  ngAfterContentInit() { } //this.element.nativeElement

  @ContentChild(Person) person;
  ngAfterContentInit() { } //this.person.sayHi();
}
@Component({
  selector: 'my-app',
  template: `
    <my-container>
      <a href="#" #link>Add</a>
      <person name="John></person>
    </my-container>
  `,
})
export class App { }
```

HostBinding

```
@Component({
  selector: 'person', // <person name="John"></person>
  template: `<h4>Person: {{name}}</h4>`,
  host: {
    '[style.backgroundColor]': "'red'",
    '[class.active]': 'true',
  },
})
export class Person {
  @Input() name;

  //<host style="color: red;"></host>
  @HostBinding('style.backgroundColor') backgroundColor = 'red';

  //<host class="active"></host>
  @HostBinding('class.active') active = true;
}
```

HostListener

```
@Directive({
  selector: '[log-clicks]', // <element log-clicks></element>
  host: {
    '(click)': 'onClick($event)',
    '(document:click)': 'documentClick($event)',
  },
})
export class LogClicks {

  //<host (click)="onClick($event)"></host>
  @HostListener('click', ['$event'])
  onClick(e){ console.log(e); }

  //<host (document:click)="onClick($event)"></host>
  @HostListener('document:click', ['$event'])
  documentClick(e) { console.log(e); }
}
```

Host CSS Styling

- matches classes, attributes, etc
- `:host(selector) { }`
  - targets host element
- `:host-context(selector) { }`
  - targets host and descendants
- `:host /deep/ selector`
  - targets host and descendants crossing boundaries

Structural Directives

Implementing *ngIf

Introduction

- Structural Directives use HTML5 template tag
- template elements are invisible by default
- its content is placed within a
  - #document-fragment

*ngIf

```
<div *ngIf="true">
  Yay!
</div>

<template [ngIf]="true">
  <div>
    Yay!
  </div>
</template>
```

Structure

```
<template [ngIf]="true">

  <div>
    Yay!
  </div>

</template>
```

*myNgIf

```
import { Directive, Input } from '@angular/core';
import { TemplateRef, ViewContainerRef } from '@angular/core';

@Directive({
  selector: '[myNgIf]'
})
export class MyNgIf {
  constructor(
    private template: TemplateRef<any>,
    private viewContainer: ViewContainerRef
  ) { }

  @Input() set myNgIf(condition: boolean) {
    if (condition) {
      this.viewContainer.createEmbeddedView(this.template);
    } else {
      this.viewContainer.clear();
    }
  }
}
```

Implementing *ngFor

Introduction

- Uses template tag
- Similarly
  - ViewContainerRef, TemplateRef
- Requires Change Detection
  - ChangeDetectorRef
- Requires helper for Arrays
  - IterableDiffers

*ngFor

```
<div *ngFor=”let elem of elems”>
  {{value}}
</div>

<template ngFor #elem [ngForOf]="elems">
  <div>{{value}}</div>
</template>
```

Structure

```
<template ngFor #elem [ngForOf]="elems">

  <div>
   {{value}}
  </div>

</template>
```

*myNgFor (1/2)

```
@Directive({
  selector: '[myNgFor]',
  inputs: ['myNgForOf']
})
export class MyNgFor implements OnInit {
  private differ: IterableDiffer;

  constructor(
    private differs: IterableDiffers,
    private changeDetector: ChangeDetectorRef,
    private template: TemplateRef,
    private viewContainer: ViewContainerRef
  ) { }

  ngOnInit(){
    if (this.myNgForOf && !this.differ) {
      this.differ = this.differs.find(this.myNgForOf).create(this.chang
    }
  }
}
```

*myNgFor (2/2)

```
export class MyNgFor implements DoCheck, OnInit {

  ngDoCheck() {
    if (!this.differ) return;

    let changes = this.differ.diff(this.myNgForOf);
    if (!changes) return;

    changes.forEachAddedItem(added => {
      let view = this.viewContainer
        .createEmbeddedView(this.template);
      view.context.$implicit = added.item;
    });

    changes.forEachRemovedItem(removed => {
      this.viewContainer.remove(removed.previousIndex);
    });
  }
}
```

Exposing Data

```
// <h2 *myNgFor="let name of elems; let ts=timestamp">
//   Hello {{name}}-{{ts | date:"mediumTime"}}
// </h2>

export class MyNgFor implements DoCheck, OnInit {

  ngDoCheck() {
    changes.forEachAddedItem(added => {
      let view = this.viewContainer
        .createEmbeddedView(this.template, {
          timestamp: new Date()
        });
      view.context.$implicit = added.item;
    });
  }
}
```

Animations

Introduction

- Build on top of Web Animations API
- CSS Transitions
- CSS Animations (@keyframes)

CSS Transitions

- Define an initial and final state
- Intermediate states are calculated automatically
- We can choose which CSS properties we want to affect
- Not all CSS properties are animatable (list)

transition example

```
.element {
  transition-property: all;
  transition-duration: 5s;
  transition-timing-function: ease-in;
  transition-delay: 1s;
}

.element {
  transition: all 5s ease-in 1s;
}
```

CSS Animations

- Define any number of states between the initial and final state
- Changes from states are calculated automatically
- We can choose which CSS properties we want to affect

animation example

```
.element {
  animation-name: fade;
  animation-duration: 5s;
  animation-delay: 1s;
  animation-iteration-count: infinite;
  animation-timing-function: linear;
  animation-direction: alternate;
}

.element {
  animation: fade 5s 1s infinite linear alternate;
}

@keyframes fade {
  0% {
    opacity: 1;
  }
  100% {
    opacity: 0;
  }
}
```

toggle

```
@Component({
  selector : 'toggle',
  animations: [
    trigger('toggle', [
      state('true', style({ opacity: 1; color: 'red' })),
      state('void', style({ opacity: 0; color: 'blue' })),
      transition('void => *', animate('500ms ease-in-out')),
      transition('* => void', animate('500ms ease-in-out'))
    ])
  ],
  template: `
  <div [@toggle]="show" *ngIf="show">
    <ng-content></ng-content>
  </div>`
})
export class Toggle {
  @Input() show:boolean = true;
}
```

new aliases 2.1+

```
animations: [
  trigger('toggle', [
    state('show', style({ opacity: 1; })),
    state('void', style({ opacity: 0; })),
    transition(':enter', animate('500ms ease-in-out')),
    transition(':leave', animate('500ms ease-in-out'))
  ])
],
```

demo

Router transitions

```
import { Component } from '@angular/core';
import { routerTransition } from './router.animations';

@Component({
  selector: 'home',
  template: `<h1>Home</h1>`,
  animations: [routerTransition()],
  host: {'[@routerTransition]': ''}
})
export class Home { }
```

Router transitions

```
import {trigger, state, animate, style, transition} from '@angular/core

export function routerTransition() {
  return slideToRight();
}

function slideToRight() {
  return trigger('routerTransition', [
    state('void', style({position:'fixed', width:'100%'}) ),
    state('*', style({position:'fixed', width:'100%'}) ),
    transition(':enter', [
      style({transform: 'translateX(-100%)'}),
      animate('0.5s ease-in-out', style({transform: 'translateX(0%)'}))
    ]),
    transition(':leave', [
      style({transform: 'translateX(0%)'}),
      animate('0.5s ease-in-out', style({transform: 'translateX(100%)'}
    ])
  ]);
}
```

demo 1

demo 2

Redux and ngrx

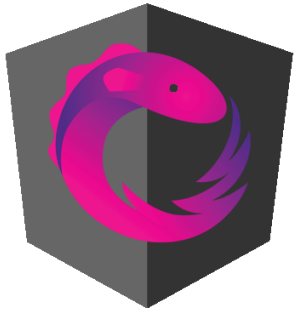
Angular Data Layer

DATA CLIENTS

GraphQL

Firebase

STATE MANAGEMENT



ngrx

Redux

Dan Abramov

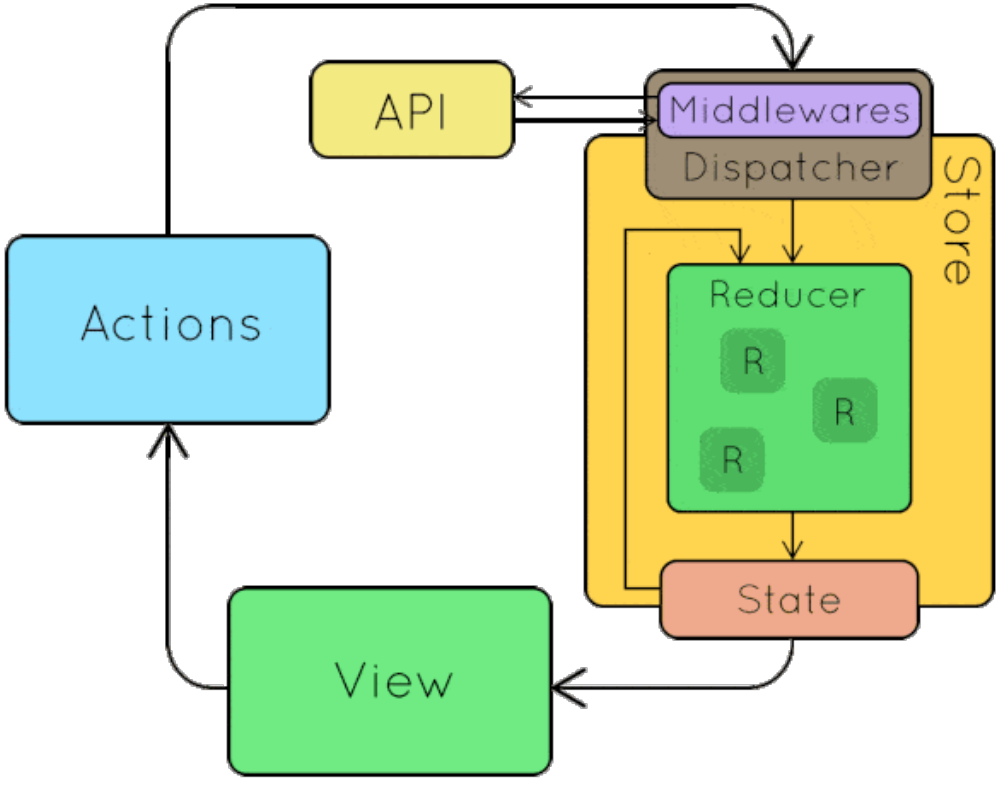
@gaearon

Libraries

Main Principles

- Unidirectional data flow
- Single Immutable State
- New states are created without ***side-effects***

Unidirectional data flow



source: blog

Single Immutable State

- Helps tracking changes by reference
- Improved Performance
- Enforce by convention or using a library. Eg: Immutable.js

Immutable by Convention

- New array using ***Array Methods***
  - map, filter, slice, concat
  - Spread operator (ES6) [...arr]
- New object using ***Object.assign*** (ES6)

Using Immutable.js

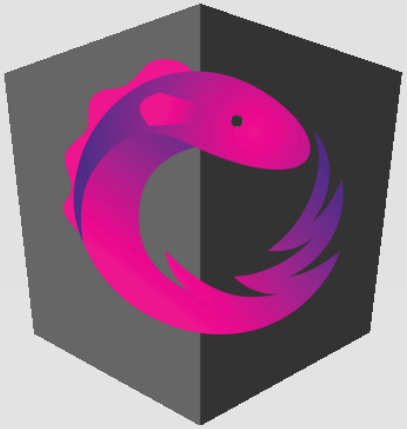
```
let selectedUsers = Immutable.List([1, 2, 3]);
let user = Immutable.Map({ id: 4, username: 'Spiderman'}):

let newSelection = selectedUsers.push(4, 5, 6); // [1, 2, 3, 4, 5, 6];
let newUser = user.set('admin', true);
newUser.get('admin') // true
```

Reducers

- ***Reducers*** create new states in response to ***Actions*** applied to the current ***State***
- Reducers are ***pure functions***
- Don't produce ***side-effects***
- Composable

Middlewares

- Sit between ***Actions*** and ***Reducers***
- Used for logging, storage and ***asynchronous operations***
- Composable



Rob Wormald

@robwormald

ngrx/store

- Re-implementation of Redux on top Angular 2 and RxJS 5
- ngrx suite: store, effects, router, db

Setup

```
import { App } from './app';
import { StoreModule } from "@ngrx/store";
import { rootReducer } from './rootReducer';

@NgModule({
  imports: [
    BrowserModule,
    StoreModule.provideStore(rootReducer)
  ],
  declarations: [ App ],
  bootstrap: [ App ]
})
export class AppModule {}

platformBrowserDynamic().bootstrapModule(AppModule);
```

ADD_TODO Action

```
// add new todo
{
  type: ADD_TODO,
  id: 1,
  text: "learn redux",
  completed: false
}
```

todos Reducer

```
const initialState = [];

const todos = (state = initialState, action:Action) => {
  switch (action.type) {
    case TodoActions.ADD_TODO:
      return state.concat({
          id: action.id,
          text: action.text,
          completed: action.completed });
    default: return state;
  }
}

// {
//  todos: [], <-- todos reducer will mutate this key
//  currentFilter: 'SHOW_ALL'
// }
```

currentFilter Reducer

```
const currentFilter = (state = 'SHOW_ALL', action: Action) => {
  switch (action.type) {
    case TodoActions.SET_CURRENT_FILTER:
      return action.filter
    default: return state;
  }
}

// {
//  todos: [],
//  currentFilter: 'SHOW_ALL' <-- filter reducer will mutate this key
// }
```

rootReducer

```
import { combineReducers } from '@ngrx/store';

const combinedReducer = combineReducers({
  todos: todos,
  currentFilter: currentFilter
});

export rootReducer = (state, action) => combinedReducer(state, action);
```

New State

```
{
  todos: [{
    id: 1,
    text: "learn redux",
    completed: false
  }],
  currentFilter: 'SHOW_ALL'
}

// {
//  todos: [], <-- we start with no todos
//  currentFilter: 'SHOW_ALL'
// }
```

Stateless Todo Component

```
// <todo id="1" completed="true">buy milk</todo>
@Component({
  inputs: ['id', 'completed'],
  template: `
    <li (click)="onTodoClick(id)"
      [style.textDecoration]="completed?'line-through':'none'">
      <ng-content></ng-content>
    </li>`,
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class Todo {
  constructor(
    private _store: Store<TodosState>,
    private todoActions:TodoActions
  ) { }

  private onTodoClick(id){
    this._store.dispatch(this.todoActions.toggleTodo(id));
  }
}
```

Redux Dev Tools

Features

- Save/Restore State
- Live Debugging
- Time travel
- Dispatch Actions

demo

Angular 2 in Production

Lazy Loading

Lazy Loading

```
// app.routing.ts
const aboutRoutes: Routes = [
  { path: 'about', loadChildren: './app/about.module.ts' },
];
export const AboutRouting = RouterModule.forChild(aboutRoutes);

//about.module.ts
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { AboutRouting } from './about.routing';
import { About } from './about.component';

@NgModule({
  imports: [ CommonModule, AboutRouting ],
  declarations: [ About ]
})
export default class AboutModule { }
```

Preloading

default preloading

```
// app.routing.ts
import { Routes, RouterModule, PreloadAllModules } from '@angular/route

const appRoutes: Routes = [
  { path: 'about', loadChildren: './app/about.module.ts' },
];

export const AppRouting = RouterModule.forRoot(appRoutes, {
  useHash: true,
  preloadingStrategy: PreloadAllModules
});
```

custom preloading

```
// app.routing.ts
import { Routes, RouterModule } from '@angular/router';
import { CustomPreload } from './custom.preload';

const appRoutes: Routes = [
  { path: 'about', loadChildren: ..., data: { preload: true } },
];

export const AppRouting = RouterModule.forRoot(appRoutes, {
  useHash: true,
  preloadingStrategy: CustomPreload
});
```

custom preloading

```
// custom.preload.ts
import { PreloadingStrategy } from '@angular/router';

export class CustomPreload implements PreloadingStrategy {
  preload(route: Route, preload: Function): Observable<any> {
    return route.data && route.data.preload ? preload() : Observable.of
  }
}
```

AoT Compilation

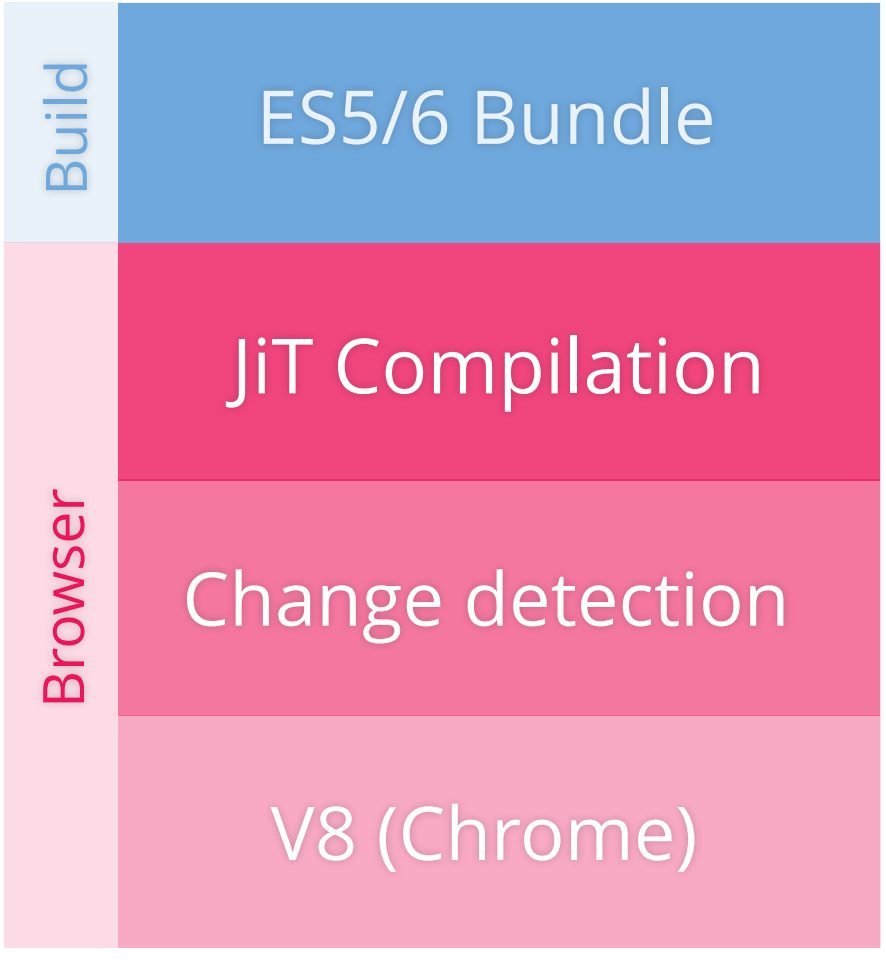
Overview

- Render templates during Build
- Reduce Angular bundle size
- Speed up First Load
- Reduce Application size

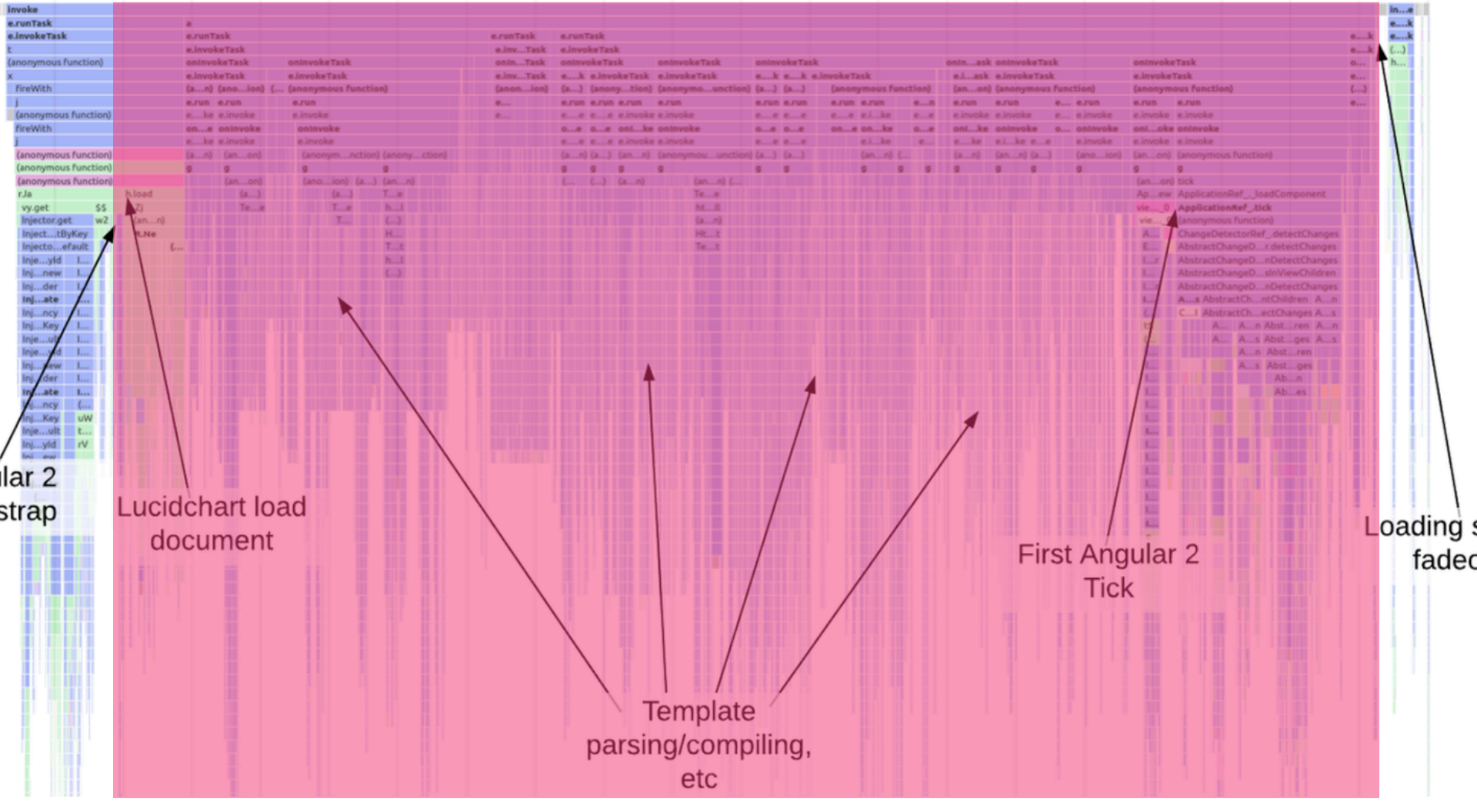
Template Compilation

- Template Renderer
- Change detection
- Just-in-Time Compilation
- Ahead-of-Time Compilation

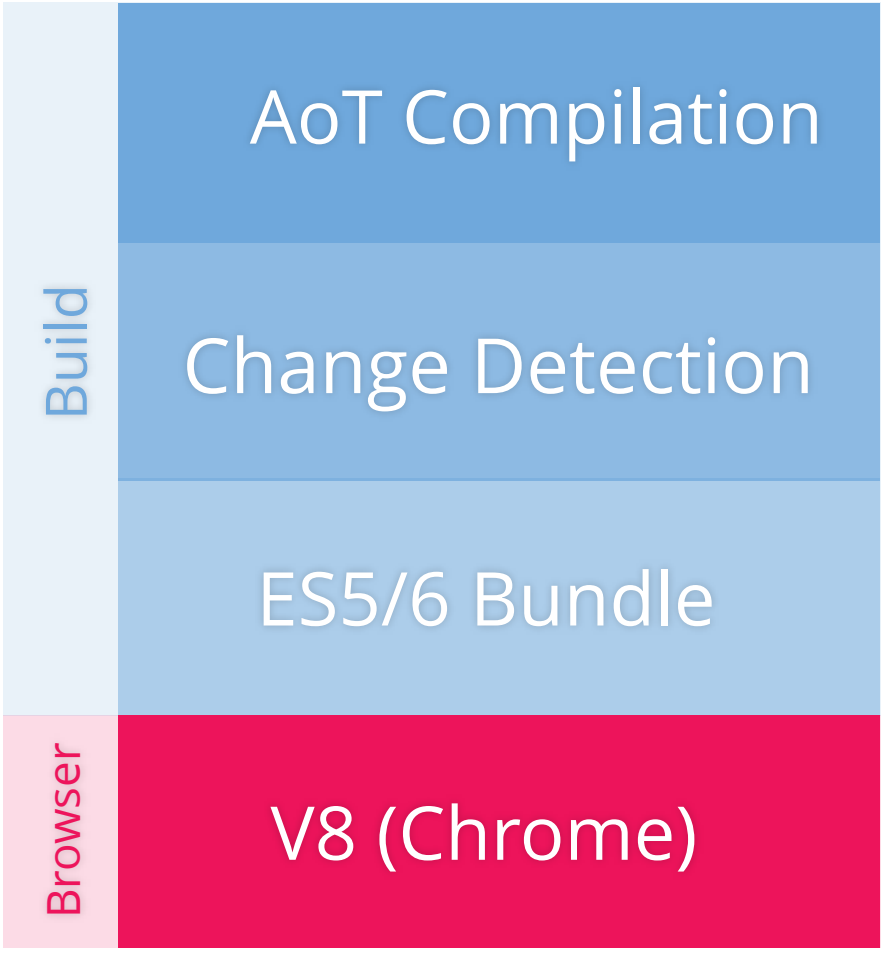
JiT Compilation



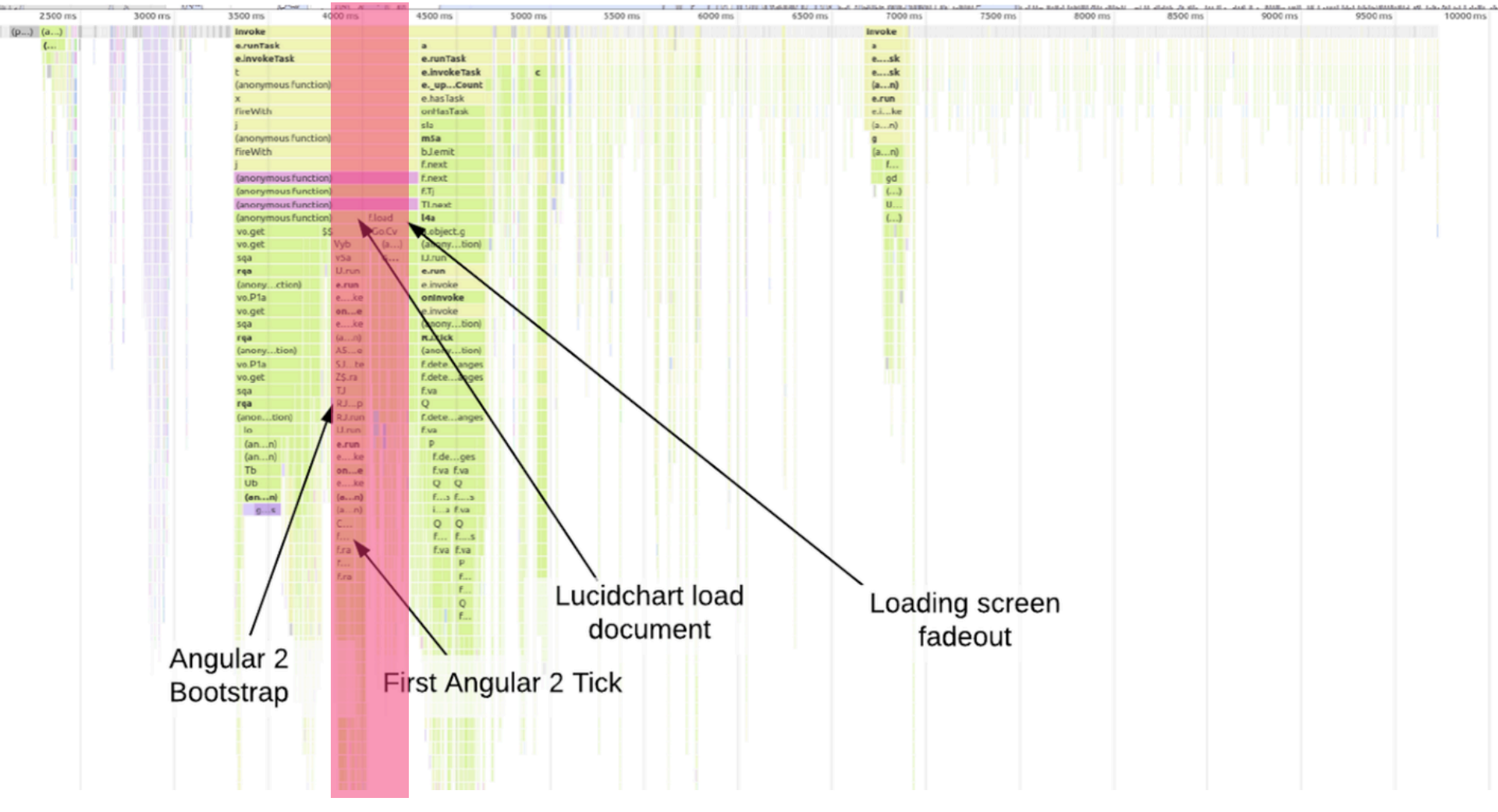
Lucidchart using JiT



AoT Compilation



Lucidchart using AoT



Thanks!