
Advanced
A5
June, 10-11 2017

DEVELOPER 13 YEARS

TRAINER 4 YEARS

WRITER 3 BOOKS

FOUNDER

SPEAKER

- Sergey Morenets, 2

Angular 5. Basics

- ✓ Angular 5.x vs 2.x vs 1.x
- ✓ TypeScript 2.x
- ✓ Components creation
- ✓ Services and DI
- ✓ Data binding
- ✓ Pipes
- ✓ Form validation
- ✓ Directives
- ✓ Working with HTTP
- ✓ Unit-testing

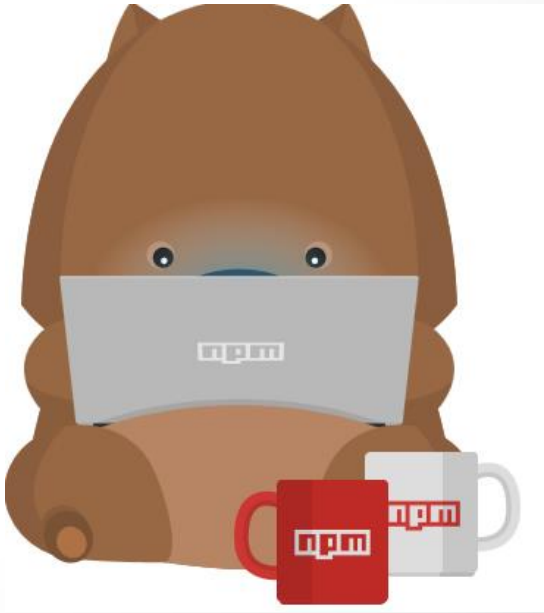
Sergey Morenets, 2018

Agenda

- ✓ Angular 5 features
- ✓ Yarn
- ✓ Angular Material 2
- ✓ SPA applications
- ✓ Angular Universal
- ✓ Optimization. AOT vs JIT
- ✓ Jest
- ✓ Module development
- ✓ Service workers
- ✓ Angular 6 features

NPM

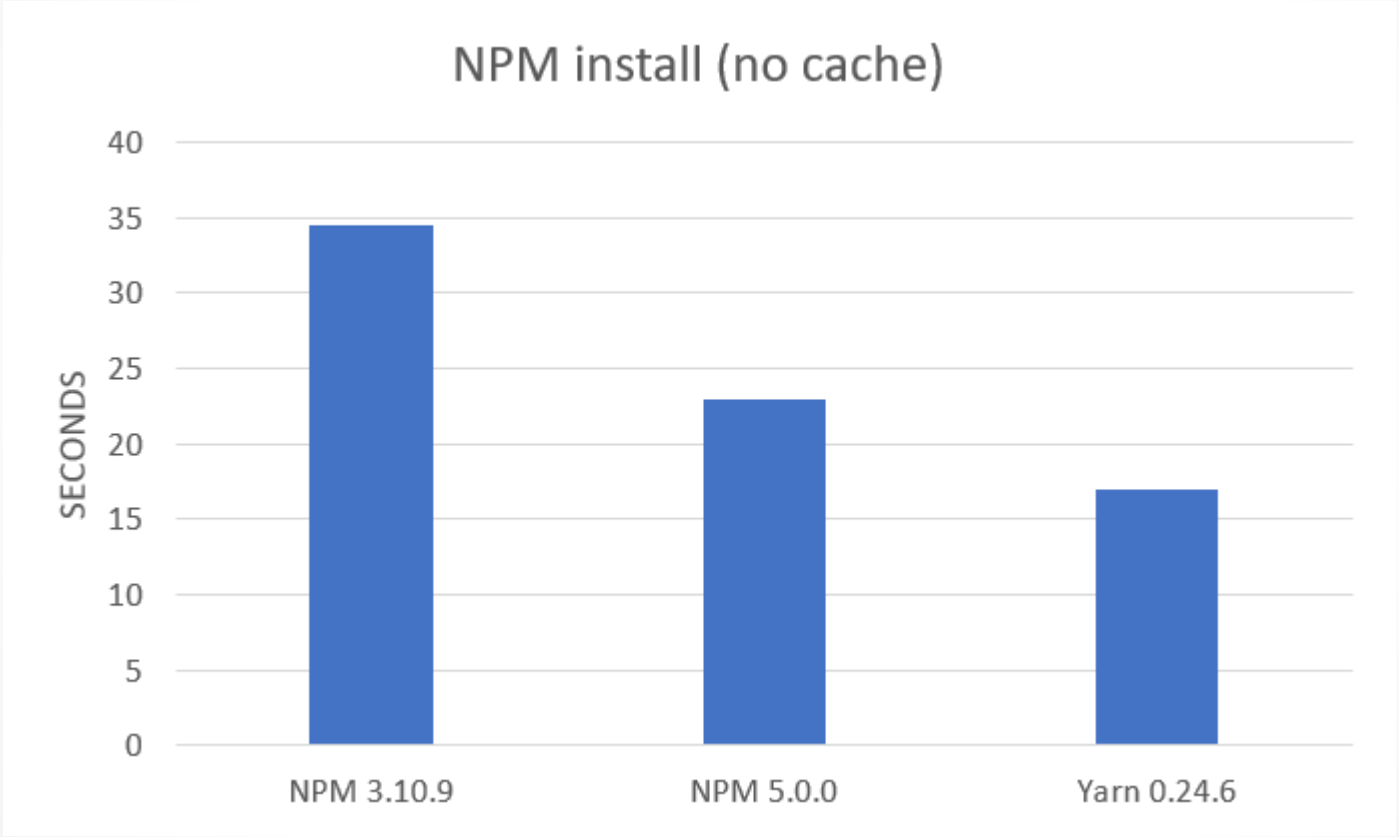
- Package manager for JavaScript
- Bundled together with **Node**
- Package(or module) is directory with files
- Hosts over 250 000 packages
- Manifest is **package.json**



Yarn

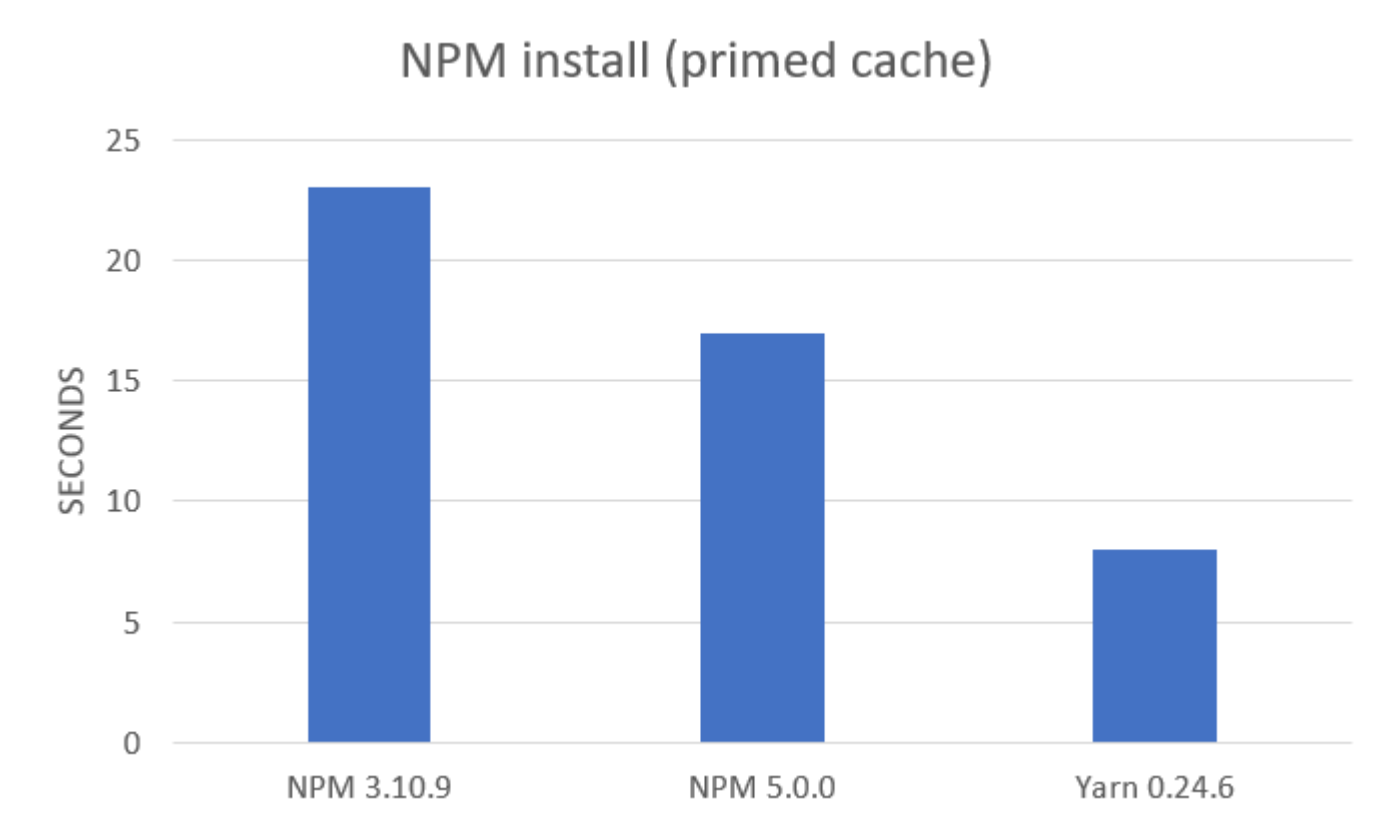
- ✓ Built by Facebook, Google, Exponent and Tilde
- ✓ Fetches modules from **NPM** registry
- ✓ Still uses **package.json** for configuration
- ✓ **Parallel** execution(comparing to sequential in NPM)
- ✓ Offline mode
- ✓ Flat mode

Yarn vs NPM



- Sergey Morenets, 2018

Yarn vs NPM



- Sergey Morenets, 2018

Yarn

- ✓ yarn **init**

  Create empty project

- ✓ yarn [global] **add** <package>@<version> [--dev]

  Adds new dependency

- ✓ yarn **remove** <package>

  Removes dependency

- ✓ yarn **install** or **yarn**

  Installs project dependencies

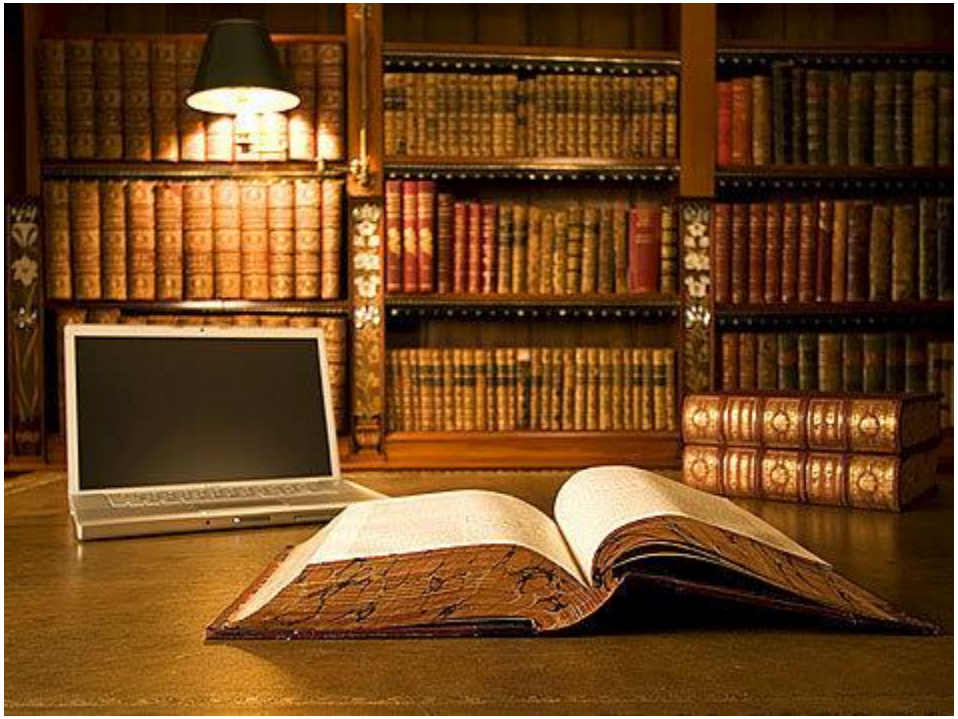
- ✓ *yarn **upgrade** <package>@<version>*

  Upgrades dependency

Task #1. Installation.

1. Install **Node**/npm.Check **NPM** version using “npm –v” command.
2. Install **Python**(2.7 or higher)
3. Install **Git**
4. Install **Yarn**. You can install it using NPM: *npm install yarn --global* . However native OS manager is recommended:
5. Create **new folder** and run *yarn init* command in this folder.

Business domain



- Sergey Morenets, 2018

Task #2. Project installation.

1. Clone project from Git repository: https://github.com/it-discovery/angular
2. Install all the required dependencies: **yarn**.
3. Open project in IDE (**WebStorm** 2018.1 is recommended)
4. Review project components, services, directives.
5. Start project: **ng serve** and observe its functionality

Task #3. Data binding.

1. Create new component with **<button>** and ** ** elements. For example: * <button>Update</button>*
2. Create all possible approaches to change content of ** ** element by clicking 'Update' button using **Angular**.

Common mistakes/flaws

10-11 2017

- Sergey Morenets, 2018

Common mistakes/flaws


```
@Component({
  selector: 'app-books',
  templateUrl: './books.component.html',
  styleUrls: ['./books.component.css']
})
export class BooksComponent {
  books: Array<Book>;

  constructor(private bookService: BookService) {
    this.books = this.bookService.getBooks();
  }
}
```

readonly

Interfaces vs classes

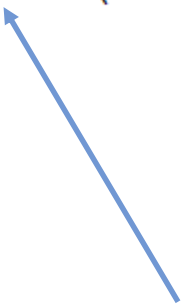

```
export interface User {
  login: string;
  password: string;
}
```

```
export class User {
  login: string;
  password: string;
}
```

Common mistakes/flaws

```
export class BooksComponent {
  books: Array<Book>;

  constructor(private bookService: BookService) {
    this.bookService.findBooks().subscribe(
        next: res => this.books = res);
  }
```



ngOnInit

Common mistakes/flaws


```
export class SampleComponent {
  private subject = new BehaviorSubject<string>( _value: '');

  constructor(private appService: AppService) {
    this.appService.findTypes().subscribe(this.subject);
  }

  getValue() {
    return this.subject.value;
  }
}
```

Common mistakes/flaws


```
export class SampleComponent implements OnDestroy {
  private subject = new BehaviorSubject<string>( _value: '');

  private subscription: Subscription;

  constructor(private appService: AppService) {
    this.subscription = this.appService.findTypes()
      .subscribe(this.subject);
  }

  ngOnDestroy(): void {
    this.subscription.unsubscribe();
  }
```

- Sergey Morenets, 2018

Common mistakes/flaws


```
export class Product {

  name: string;

  price: number;

  discountDate: Date;
}
```

```
isDiscountActive(): boolean {
  return this.discountDate > new Date();
}
```

```
constructor(httpClient: HttpClient) {
  httpClient.get<Product>( url: '/products')
    .filter((item: Product) => item.isDiscountActive())
    .subscribe( next: obj => console.log(obj));
}
```

Common mistakes/flaws

```
export class DataComponent implements OnInit {

  constructor(private activeModal: NgbActiveModal) {
  }
```

```
constructor(private el: ElementRef, private service: HighlightJsService,
            private route: ActivatedRoute,private articleService: ArticleService,
            authService: AuthenticationService,
            private subscribeService: SubscribeService,
            private attachmentService: AttachmentService) {
  super(authService);
```

- Sergey Morenets, 2018

Common mistakes/flaws


```
<div *ngFor="let book of books">
  <app-book [book]="book"></app-book>
</div>
```

```
<div *ngFor="let book of books;trackBy : trackBook()">
  <app-book [book]="book"></app-book>
</div>
```

```
trackBook(index, book: Book): string {
  return book ? book.title : undefined;
}
```

- Sergey Morenets, 2018

Angular 5 features

- ✓ Service workers
- ✓ Form changes
- ✓ HTTP Client
- ✓ Router changes
- ✓ i18N

Angular 5. Form changes


```
this.passwordCtrl = new FormControl('', {
  validators: Validators.required,
  updateOn: 'blur'
});
```

blur, **change**, submit

```
this.userForm = new FormGroup({
  username: '',
  password: ''
}, { updateOn: 'blur' });
```

Angular 5. HTTP Client


```
const headers = new HttpHeaders().set('Authorization', '
secret');
const params = new HttpParams().set('page', '1');
return this.http.get('/api/users', { headers, params });
```

Angular 4.3

```
const headers = { 'Authorization': 'secret' };
const params = { 'page': '1' };
return this.http.get('/api/users', { headers, params });
```

Angular 5.x

- Sergey Morenets, 2018

Router changes

- ChildActivationStart and ChildActivationEnd events
- Possibility to reload a route

```
providers: [
  // ...
  RouterModule.forRoot(routes, {
    onSameUrlNavigation: 'reload'
  })
]
```

i18 changes


```
import { registerLocaleData } from '@angular/common';
import localeFr from '@angular/common/locales/fr';

registerLocaleData(localeFr);
```

```
@Component({
  selector: 'ns-locale',
  template: `
    <p>The locale is {{ locale }}</p>
    <!-- will display 'en-US' -->

    <p>{{ 1234.56 | number:'1.0-3':'fr-FR' }}</p>
    <!-- will display '1 234,56' -->
  `
})
class DefaultLocaleComponentOverridden {
  constructor(@Inject(LOCALE_ID) public locale: string) { }
```

Pipes

- ✓ Used to transform and filter data
- ✓ Raw data formatting
- ✓ Called ‘filters’ in Angular 1.x
- ✓ Can be used in HTML/application code

Built-in pipes

Name	Description	Example
json	Converts object into JSON text fomat	{{book | **json**}}
slice	Filters collection(array) creating new sub-collection	*ngFor="let book of books | **slice** : 0: 5"
number	Formats number using current regional settings	{{ amount | **number** }} {{ amount | **number** : '.2-2'}}
percent	Format number into percentage format	{{ amount | **percent** }}
currency	Applies currency symbol to the number	{{ amount | **currency** }} {{ amount | **currency** : 'UAH'}} {{ amount | **currency** : 'EUR' : true}}

User-defined pipes


```
import {PipeTransform, Pipe} from '@angular/core';

@Pipe({name : 'stub'})
export class StubPipe implements PipeTransform {
  transform(value, args) {
    return value;
  }
}
```

```
export class StubPipe implements PipeTransform {
  transform(value: any, ...args: Array<any>): any {
    for (const item of args) {
      console.log(item);
    }
    return value;
  }
}
```

Task #4. Custom pipe

1. Create new pipe “**sort**” that will sort elements of array (ascending or descending). The format of this pipe will be the following:
2. items | sort : ‘title’ -- sorts **items** array by **title** property in ascending order
3. Create new ‘**ui’** module and put this pipe there.
4. Apply this pipe to the **BooksComponent** to sort books by title or author.

RxJS

- ✓ Push-based collection of events
- ✓ Implementation of observer design pattern
- ✓ Observable object sends notifications
- ✓ Observer object receive notifications

RxJS

Type	Description
Observable	Push-based collection of values
Observer	Consumer of push-based notifications from Observer
Subscription	Result of subscription of Observer on Observable
Scheduler	Allows to schedule/order tasks execution
Subject	Can act as both data producer/consumer

Observable. Creation


```
private execute() {
  let observable: Observable<Number> = Observable.empty();
  observable = Observable.of(1, 2);
  observable = Observable.range(1, 10);
  observable = Observable.from([1, 2, 3]);
  observable = Observable.throw(new Error("Unexpected"));
}
```

```
observable.subscribe((text) => console.log(
    "Event received " + text));
```

```
observable.subscribe((text) => console.log(
    "Event received " + text),
  (err) => console.log("Error " + err),
  () => console.log("Process completed"));
```

Observable. Creation


```
observable = Observable.interval(2000);
observable.subscribe(x => console.log(x));
```

0, 1, 2, 3, ...

```
const subsription: Subscription = Observable.from([1, 2, 3])
    .subscribe(x => console.log(x));
subsription.unsubscribe();
```

```
const subject: Subject<number> = new Subject();
subject.subscribe(x => console.log(x));
subject.subscribe(x => this.exec(x));

observable.subscribe(subject);
```

Observable. Creation


```
observable = Observable.create(
  (observer: Observer<string>) =>

    setTimeout(function () {
      observer.next('OK');
      observer.complete();
    }, 3000));
```

```
observable.timeout(1000).
      subscribe(x => console.log(x),
      err => console.log('Error ' + err));
```

- Sergey Morenets, 2018

Observable. Operations


```
observable = Observable.range(1, 10);

observable.filter((x:number) => x%2 == 0).subscribe(
    (text) => console.log("Even number " + text));
```

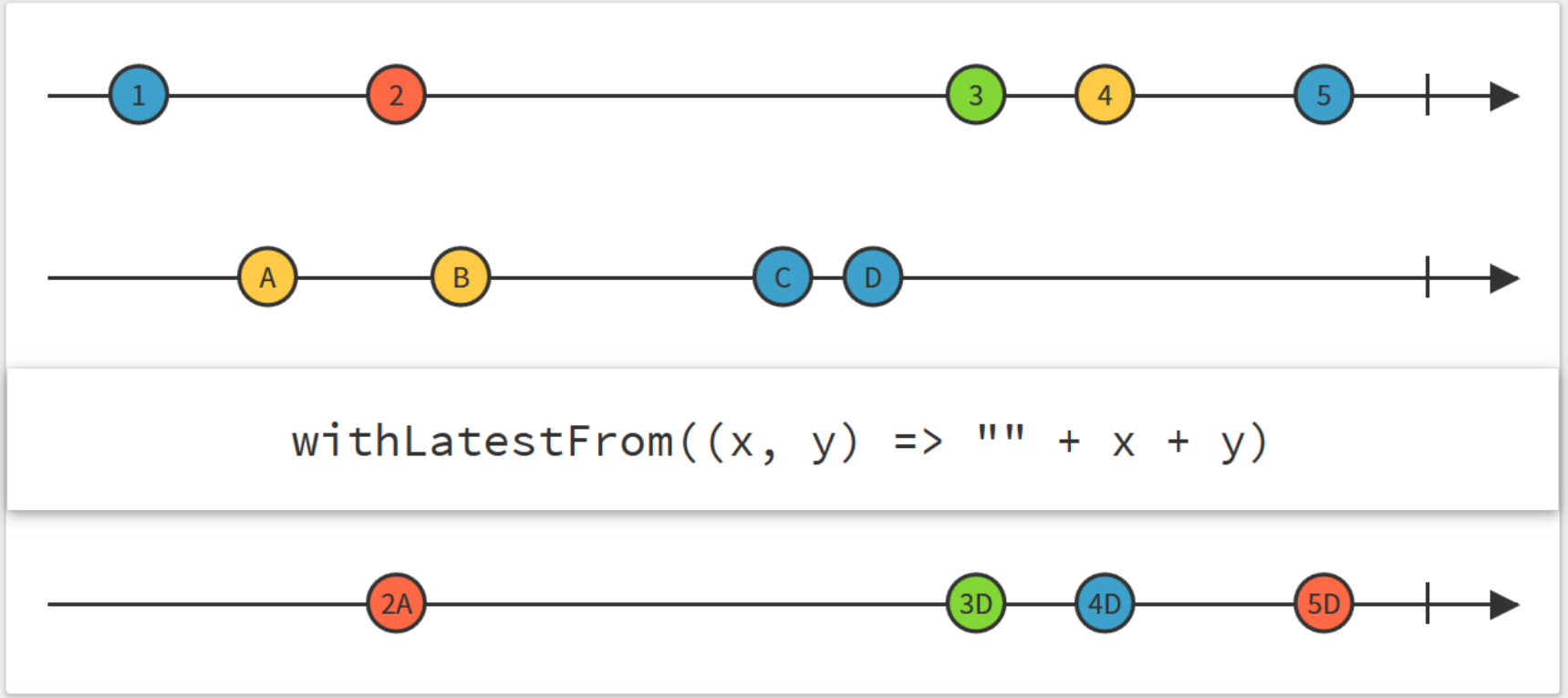
```
observable.map((x:number) => x * 10).subscribe(
  (text) => console.log(text));
```

```
observable = Observable.from([1, 2, 3]);
observable.reduce( (x: number, y: number) => x + y)
  .subscribe(x => alert(x));
```

```
observable.max((x: number, y: number) => x > y ? x : y);
```

- Sergey Morenets, 2018

RxMarbles



- Sergey Morenets, 2018

Task #5. Observable

1. Create few Observable objects, using **from(), range()** and **of ()** functions.
2. Subscribe to observable to print received events or generated errors.
3. Create Observable that pushes **Latin** lowercase letters (from ‘a’ to ‘z’) and verify its behavior.

Task #6. Global Event Bus

1. Create new class **ApplicationEvent** with source and message properties.
2. Create new service **AsyncEventBus**. It will serve as global event bus and allow components to:
    1. Send events of **ApplicationEvent** type
    2. Subscribe to **ApplicationEvent** events
3. Use **RxJs** functionality to generate/subscribe for events.

Custom validator


```
export interface ValidatorFn {
    (c: AbstractControl): {
        [key: string]: any;
    };
}
```

```
export class EmailValidator {
  static getEmailValidator() {
    return function emailValidator(c: FormControl):
      { [s: string]: boolean } {
      if (!c.value.match(/^\w+@\S+\.\S+$/)) {
        return {invalidChars: true};
      }
    }
  }
}
```

Custom validator


```
constructor(formBuilder: FormBuilder) {
  this.userForm = formBuilder.group({
    username: formBuilder.control('', [Validators.required,
      Validators.minLength(3), EmailValidator.getEmailValidator()]),
    password: formBuilder.control('', Validators.required)
  });
}
```

- Sergey Morenets, 2018

Asynchronous validator

```
constructor(formBuilder: FormBuilder) {
  this.userForm = formBuilder.group({
    username: formBuilder.control('', [Validators.required,
      Validators.minLength(3), EmailValidator.getEmailValidator()]),
    password: formBuilder.control('', Validators.required)
  });

  this.userForm.get('username').valueChanges.
      subscribe(x => alert(x));
}
```


returns **Observable**

Asynchronous validator


```
validateUniqueName(value: string) {
  if (value === 'a@a.com' || value === 'user2') {
    return false;
  }
  return true;
```

```
const username: AbstractControl = this.userForm.get('username');

username.valueChanges.debounceTime(500)
  .map(value => this.validateUniqueName(value))
  .subscribe(flag => {
    if (!flag) {
      username.setErrors({asyncInvalid: true});
    } else {
      username.setErrors(null);
    }
  });
```

Asynchronous validator


```
validateUniqueName(c: AbstractControl): Observable<ValidationErrors> {
  return Observable.of('user1@test.com', 'user2@test.com', null)
    .filter(item => item === c.value || item == null)
    .map(item => {
        if (item == null) {
          return null;
        } else {
          return {asyncInvalid: true};
        }
      }
    ).first();
```

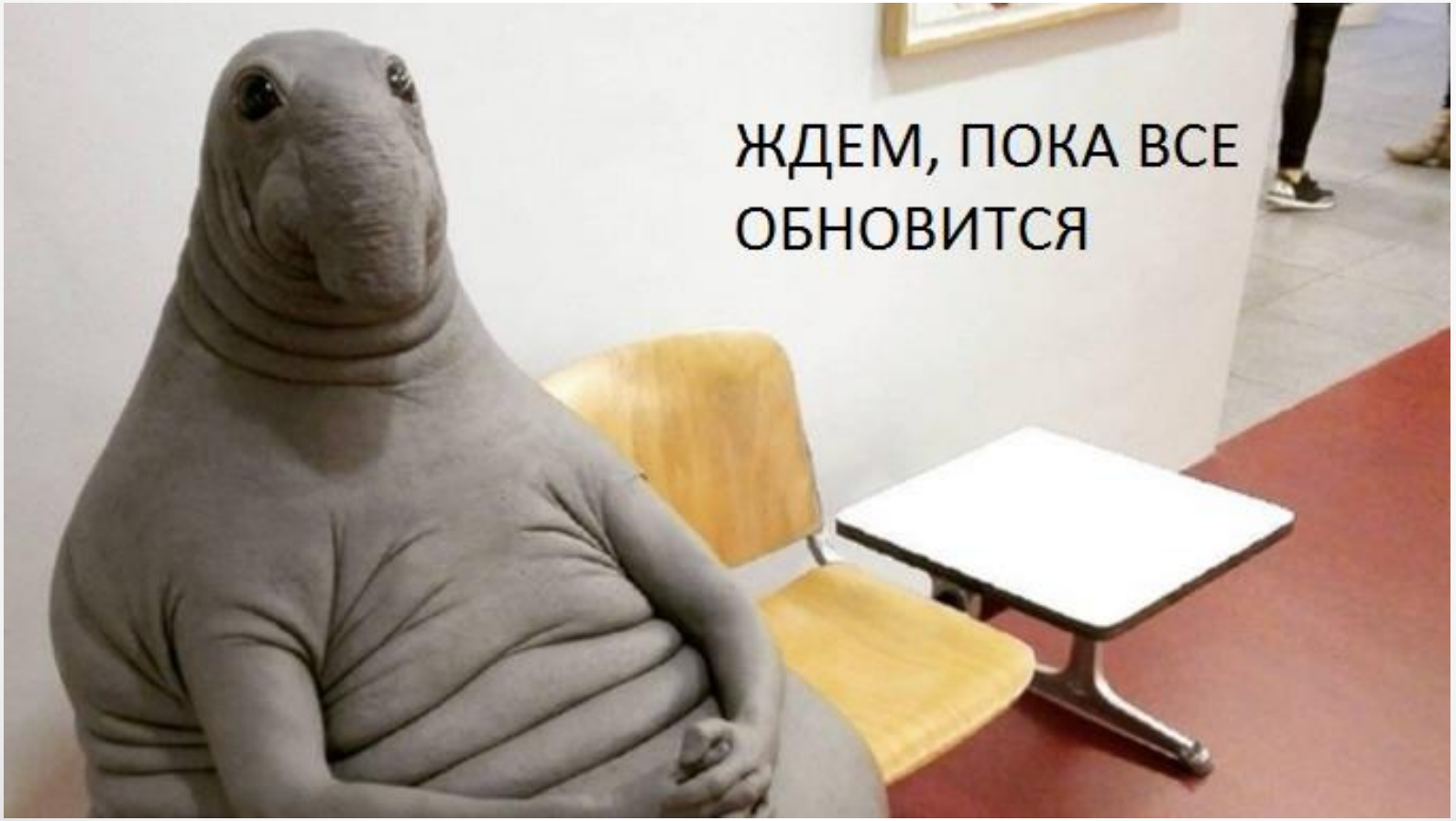
```
this.userForm.get('username').setAsyncValidators(
  this.validateUniqueName);
```

- Sergey Morenets, 2018

Task #7. Custom & async validators

1. Create your own validator function that checks that author field contains at least two words.
2. Add this function to the validators section of the **FormControl.control** method.
3. Check that submission and validation works.
4. Add new **asynchronous** validator that verifies if there already exists book with such title. Use book service to check book existence.

Change detection



- Sergey Morenets, 2018

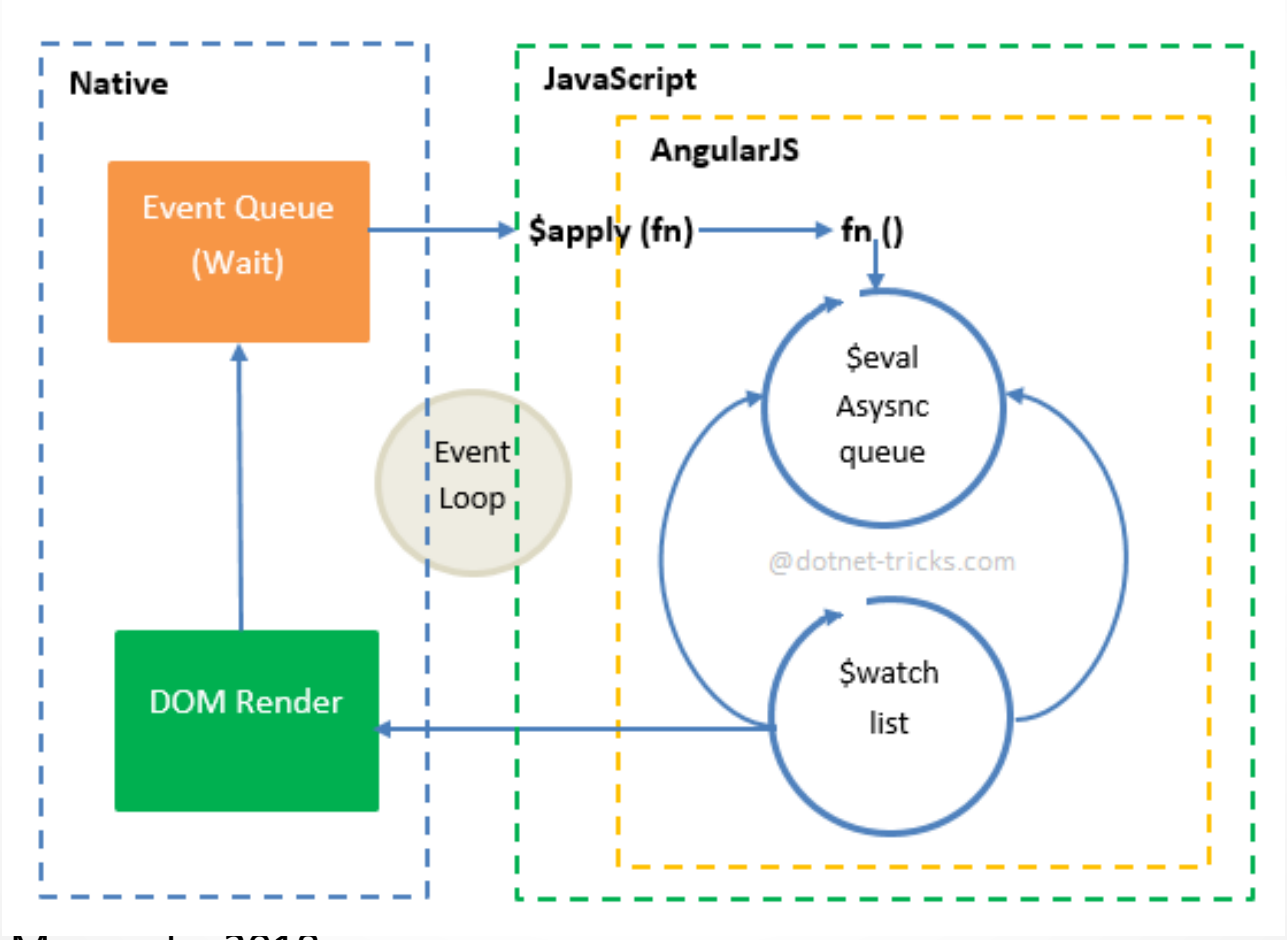
Change detection

- Sergey Morenets, 2018

Angular 1.Change detection

- Invoked by directives(**ng-model**), services**($http, $timeout**) or **$scope.$apply**()
- **Watcher** created for every dynamic expression
- After each event Angular triggers **digest cycle**
- During digest cycle every expression is evaluated and compared with old value
- This cycle is repeated until results are stable

Angular 1.Change detection



- Sergey Morenets, 2018

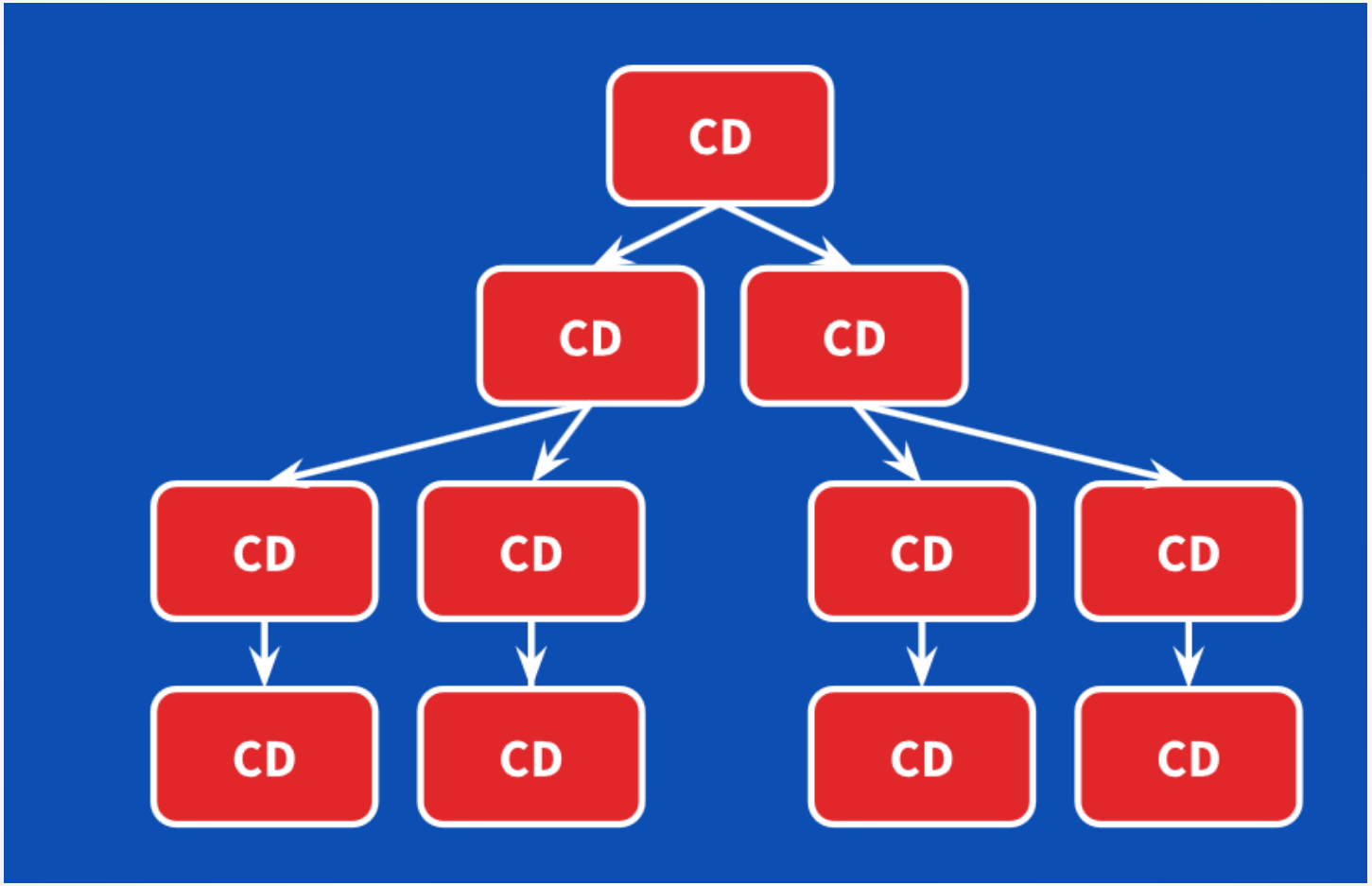
Change detection

- Component state can change by events, timers and other asynchronous tasks
- Each model change should reflect in DOM update
- Each component has its own change detector
- Angular uses **zones** to get notified about changes
- Zone is language feature in **Dart** ported to JavaScript as Zone.js
- A zone is an execution context that persists across asynchronous tasks(similar to **ThreadLocal** in Java)
- Zone.js patches all asynchronous runtimes and provides hooks(before, after, exception)

Change detection

- ✓ Component can update state of its children but not ancestors
- ✓ If child component updates its parent then exception is thrown(in **development** mode)
- ✓ Single traversal across the component tree
- ✓ No more infinitive loops
- ✓ The whole process is faster

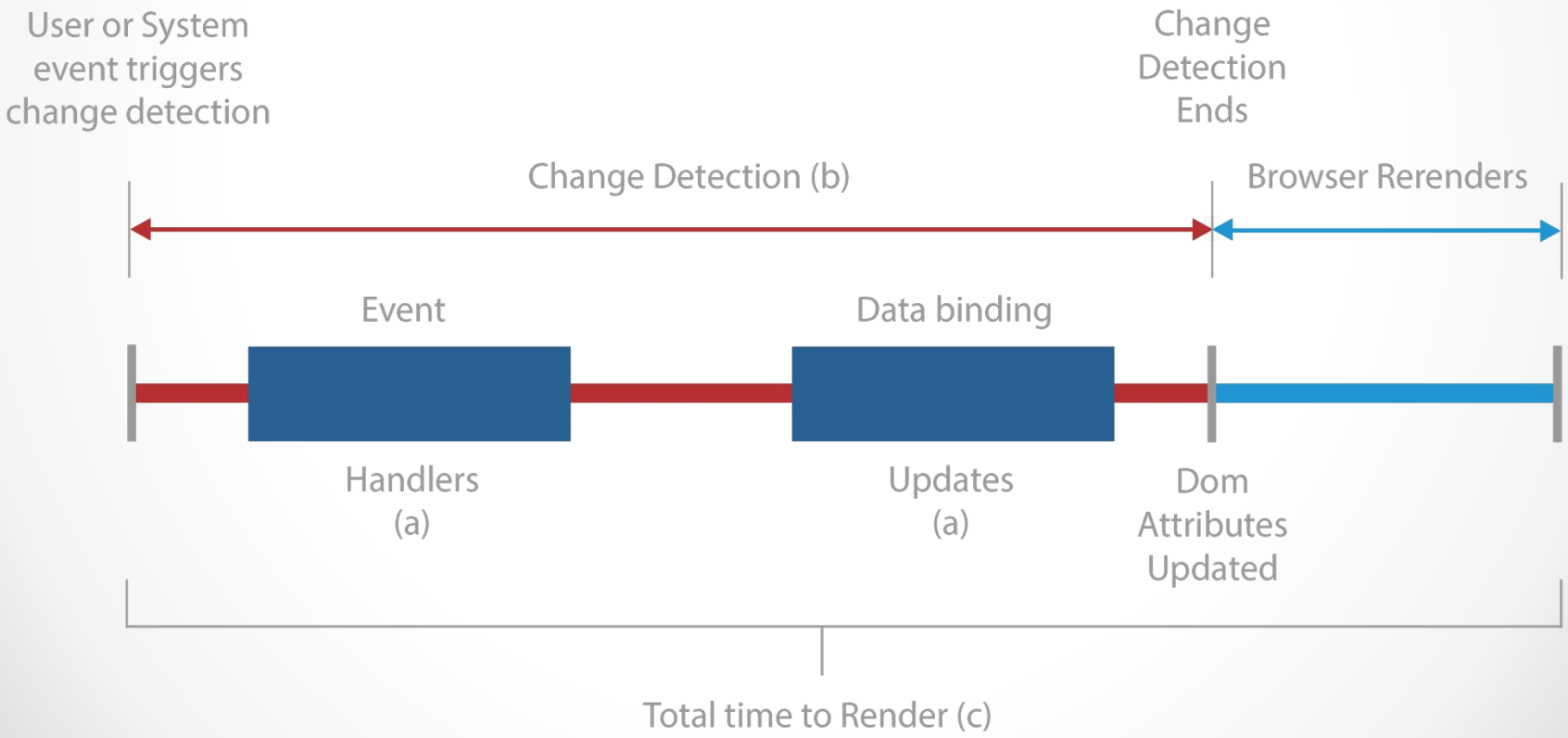
Change detection



- Sergey Morenets, 2018

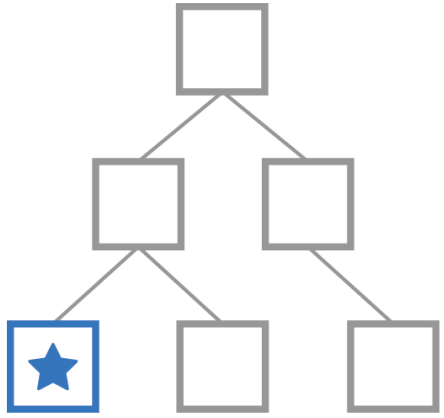
Runtime performance

- ✓ Total rendering time should be less than 17 ms to achieve 60FPS

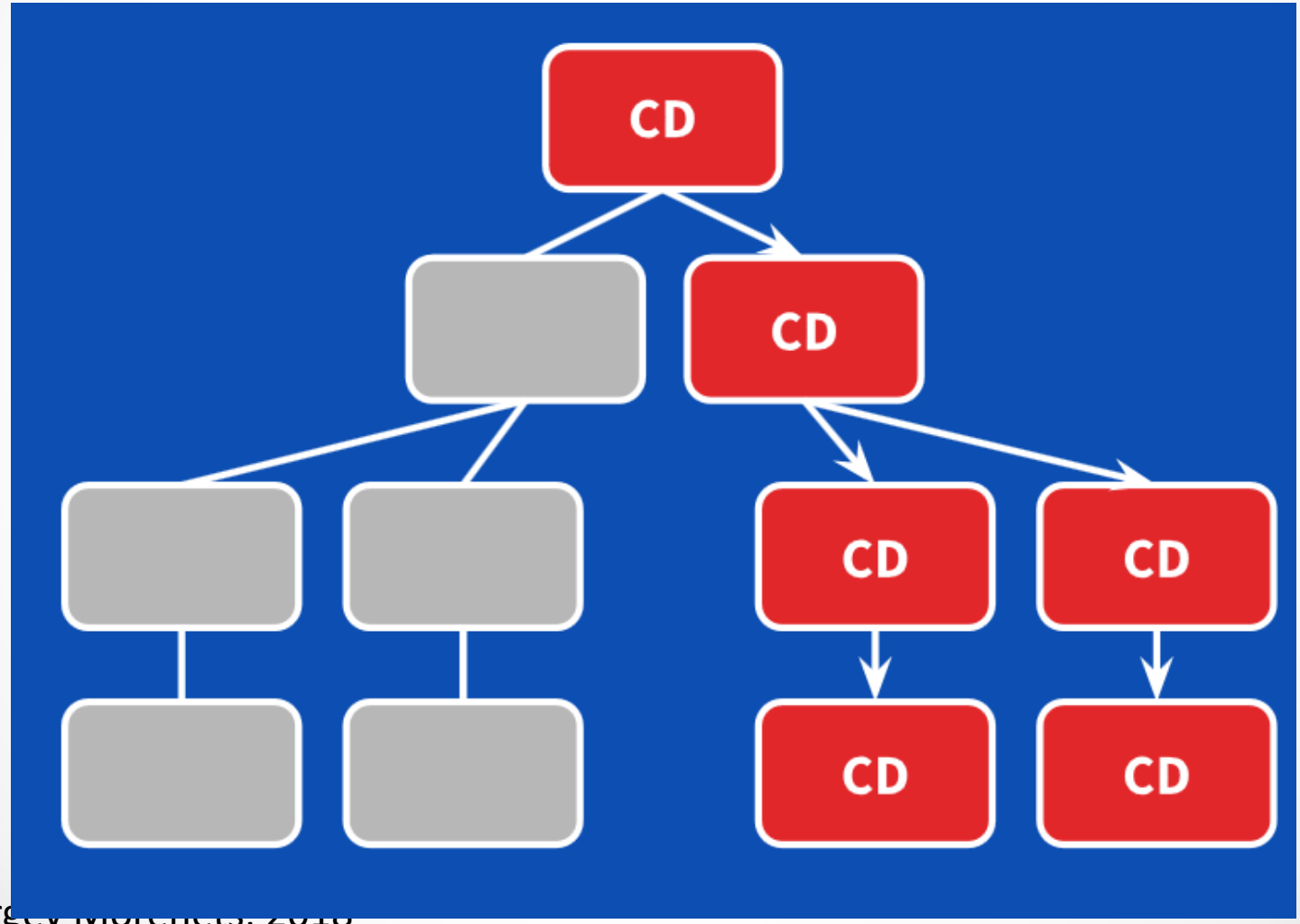


Runtime performance

- ✓ Fast event handlers
- ✓ Reduce number of callback executions inside change detection cycle
- ✓ Reduce time of change detection cycle



Change detection



- Sergey Morenets, 2018

Change detection


```
@Component({
  selector: 'language',
  templateUrl: './language.component.html',
  styleUrls: ['./language.component.css']
})
export class LanguageComponent implements OnInit {
  @Input()
  label: string;

  lastModified(): string {
    return new Date().toISOString();
  }
```

```
<div>{{lastModified()}}</div>
```

Change detection


```
@Component({
   selector: 'language',
   templateUrl: './language.component.html',
   styleUrls: ['./language.component.css'],
   changeDetection: ChangeDetectionStrategy.OnPush
})
export class LanguageComponent implements OnInit {
   @Input()
   label: string;

   lastModified(): string {
      return new Date().toISOString();
   }
```

- Sergey Morenets, 2018

Task #8. Change detection

1. Create new component that will contain **@Input** property.
2. Add a method to new component that will increment and display a **counter**. Invoke it in the component template.
3. Add an event handler in the parent component that will change a model. Modify **changeDetection** attribute of **@Component** decorator in the child component
4. Run application and make sure it works properly.

Task #9. Global event bus. Sync version

1. Create new interface **IEventConsumer** with single method
2. Create new interface **IEventBus** with subscribe/send methods. subscribe method should accept parameter of **IEventConsumer** type. Let AsyncEventBus class implement **IEventBus** interface.
3. Create new synchronous implementation of IEventBus that simply stores consumers in the **Array<IEventConsumer>**

Component sample


```
@Component({
  selector: 'app-book5',
  template: '<input [(ngModel)]="text"/>{{text}}'
})
export class Book5Component {
  text = 'value';
}
```

```
export class Book5Component implements AfterViewChecked {
  text = 'value';

  ngAfterViewChecked(): void {
    this.text = 'new value';
  }
}
```

Change detection


```
▶ERROR Error:        ng:///EventModule/Bo…nent.ngfactory.js:7
ExpressionChangedAfterItHasBeenCheckedError: Expression has
changed after it was checked. Previous value: 'model: value'.
Current value: 'model: new value'.
```

```
export class Book5Component implements AfterViewChecked {
  text = 'value';

  constructor(private cdr: ChangeDetectorRef) {
  }

  ngAfterViewChecked(): void {
    this.text = 'new value';
    this.cdr.detectChanges();
  }
}
```

Change detection


```
export class Book5Component implements OnInit {
  text = 'value';

  constructor(private cdr: ChangeDetectorRef) {
  }

  ngOnInit(): void {
    this.cdr.detach();
    this.text = 'update';
    setInterval( callback: () => {
      this.cdr.reattach();
    }, ms: 5000);
  }
}
```

Disable change detection

Enable change detection

Task #10. Programmatic change detection

1. Try to enable/disable component check detection using detach/reattach functions of **ChangeDetectorRef** object.
2. Update getBooks() function in **BookService** class so that it returns list of books in the random order.
3. Modify BooksComponent so that it refresh list of books each 3 seconds.

Component management


```
@Component({
  selector: 'app-book',
  templateUrl: './book.component.html',
  styles: []
})
export class BookComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

```
<app-book></app-book>
```

- Sergey Morenets, 2018

Component management


```
@ViewChild('parent', {read: ViewContainerRef})
private parent: ViewContainerRef;

constructor(private componentFactoryResolver:
              ComponentFactoryResolver) {
}
```

parent.component.ts

```
ngOnInit(): void {
  const bookComponent = this.componentFactoryResolver
    .resolveComponentFactory(BookComponent);
  this.parent.createComponent(bookComponent);
}
```

```
<ng-template #parent></ng-template>
```

parent.component.html

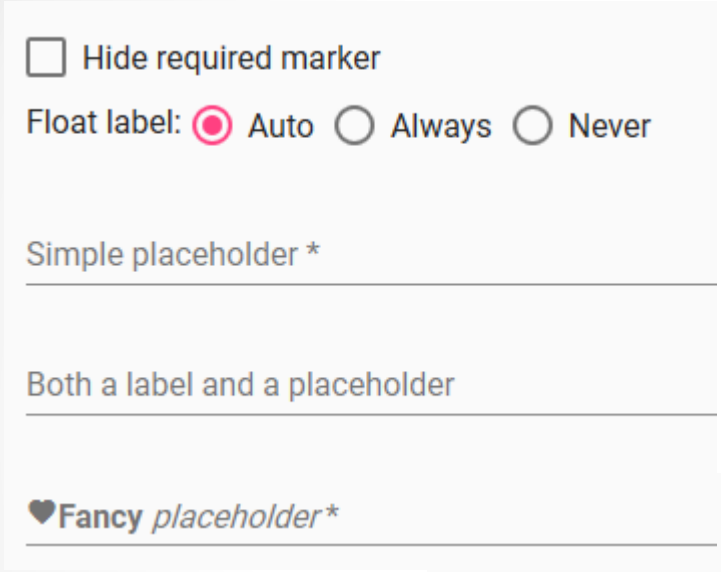
```
entryComponents: [BookComponent]
```

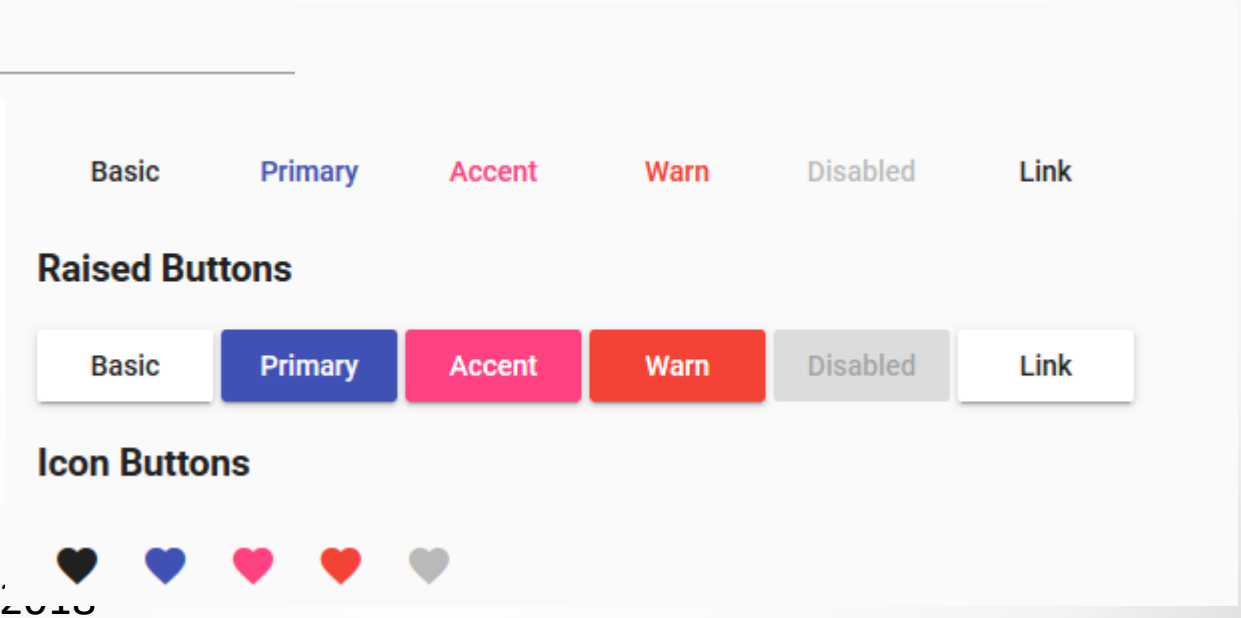
app.module.ts

Task #11. Component management

1. Create new component **BannerHeaderComponent**
2. Create new components BestBuyComponent and DiscountsComponent that will display ad content.
3. Create new service **BannerService** that should return random banner component and change it every 5 seconds.
4. Update BannerHeaderComponent to use BannerService and dynamically change banners provided by the service.

Angular Material 2





- Sergey Morenets, 2018

Angular Material 2. Overview

- ✓ Reference implementation of **Google Material** Design
- ✓ Modern high-quality UI component set (26 components) designed for desktop/mobile platforms and tested on all modern browsers
- ✓ Support themes
- ✓ Optimized for performance
- ✓ Fully integrated with Angular
- ✓ Internationalized

Angular Material 2. Installation

- ✓ npm install --save @angular/material
- ✓ npm install --save @angular/animations
- ✓ (optional) npm install --save hammerjs
- ✓ Select theme
- ✓ (optional) Add Material icons

Angular Material 2. Installation

app.module.ts

```
imports: [
  MdButtonModule,
  MdCheckboxModule,
  MdIconModule
],
```

```
<button md-button>Update</button>
<button md-raised-button>Raised button</button>
<button md-fab><md-icon>check</md-icon></button>
<button md-mini-fab><md-icon>check</md-icon></button>
```

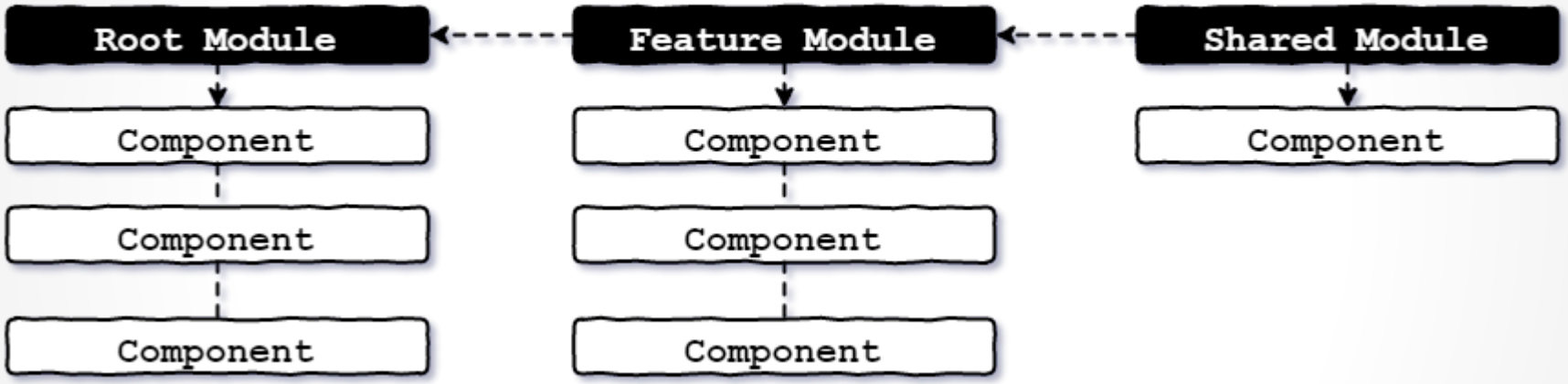
Task #12. Angular Material 2

1. Install **Angular Material**: *npm install --save @angular/material*
2. Install **Animations** module: *npm install --save @angular/animations*
3. Import **BrowserAnimationsModule** into root module.
4. Import HammerJS: *npm install --save hammerjs*
5. (optional) Add Material icons into **index.html**: <link href="https://fonts.googleapis.com/icon?f... ons" rel="stylesheet">

Task #13. Angular Material. Dialogs

1. Review Dialog component API: https://material.angular.io/components/dialog/overview
2. Change **BookRegistration** component so that it complies with Dialog layout and add directives: mat-dialog-title, mat-dialog-actions, mat-dialog-content and mat-dialog-close.
3. Change **BooksComponent** so that it opens new dialog when user wants to submit new book.

Angular2 Modules



Task #14. Module development

1. Create new module **News**. Create **NewsComponent** component there.
2. Create new module **Books**. Move all book-related files there (components, directives, pipes and services). Do not forget to put them into “declarations” section of the **books** module.
3. Put all the externally used components/pipes/directives into “exports” section of the **books** module.

Routing


```
import {Routes} from '@angular/router';
import {PanelHeaderComponent} from "./app.component";
import {SimpleComponent} from "./simple/simple.component";

export const ROUTES: Routes = [
  {path: '', component: PanelHeaderComponent},
  {
    path: 'simple', component: SimpleComponent
  }
];
```

app.routes.ts

Routing


```
imports: [
  BrowserModule,
  FormsModule,
  HttpModule,
  ReactiveFormsModule,
  CoreModule,
  RouterModule.forRoot(ROUTES)
```

app.module.ts

Routing


```
<header>My header</header>

<router-outlet></router-outlet>

<footer>
  <a href="" routerLink="/"
        routerLinkActive="selected">Panel header</a>
  <a href="" routerLink="/simple"
     routerLinkActive="selected">Simple</a>

</footer>
```

http://localhost:4200/

home.component.html

http://localhost:4200/simple

Routing. Hash tag


```
imports: [
  BrowserModule,
  FormsModule,
  HttpModule,
  ReactiveFormsModule,
  CoreModule,
  RouterModule.forRoot(ROUTES)
],
providers: [{
    provide: LocationStrategy,
    useClass: HashLocationStrategy
}],
```

http://localhost:4200/#/

http://localhost:4200/#/simple

app.module.ts

Routing. Programmable


```
constructor(private router: Router) {

}

navigate() {
  this.router.navigate(['/simple']);

}
```

home.component.ts

Routing. Parameters


```
export const ROUTES: Routes = [
  {path: '', component: PanelHeaderComponent},
  {
    path: 'simple/:title', component: SimpleComponent
  }
];
```

app.routes.ts

Routing. Parameters


```
<a href="" routerLink="/"
      routerLinkActive="selected">Panel header</a>
<a href="" routerLink="/simple/test"
   routerLinkActive="selected">Simple</a>
```

home.component.html

Routing. Programmable


```
constructor(private appService: AppService,
            private router: Router) {
  this.execute();
}

navigate() {
  this.router.navigate(['/simple', 'Hello!']);

}
```

home.component.ts

Routing. Parameters


```
export class SimpleComponent implements OnInit{
  private title:string;

  constructor(private route:ActivatedRoute) {
  }

  ngOnInit(): void {
    this.title = this.route.snapshot.params['title'];
  }
}
```

simple.component.cs

Routing. Lazy loading


```
const routes: Routes = [
  {
    path: 'customers',
    loadChildren: 'app/customers/customers.module#CustomersModule'
  },
  {
    path: 'orders',
    loadChildren: 'app/orders/orders.module#OrdersModule'
  },
  {
    path: '',
    redirectTo: '',
    pathMatch: 'full'
  }
];
```

app.routes.ts

Routing. Events

- ✓ NavigationStart
- ✓ NavigationEnd
- ✓ NavigationCancel
- ✓ NavigationError
- ✓ RouteConfigLoadStart
- ✓ RouteConfigLoadEnd

Task #15. Angular Router

1. Create new module **news**. Create **NewsComponent** component there.
2. Add new **Routes** constant that will store **routes** and linked components. Put it in **app.routes.ts** file. There will be three routes for navigation: news, article and single article.
3. Add RouterModule.forRoot(Routes) to root module
4. Add **<router-outlet></router-outlet>** tag to the main page.

Progressive web apps(PWA)

- Sergey Morenets, 2018

Progressive web apps

- Progressive web apps is user experiences which combine the best of the web and the best of the apps
- **Progressive** — Work for every user, regardless of browser choice
- **Responsive** — Fit any form factor: desktop, mobile, tablet
- **Connectivity independent** — Service workers allow work offline, or on low quality networks.
- **App-like** — Feel like an app to the user with app-style interactions and navigation.
- **Fresh** — Always up-to-date thanks to the service worker update process.

Progressive web apps

- ✓ PWA is user experiences which combine the best of the web and the best of the apps
- ✓ **Safe** — Served via HTTPS to prevent snooping
- ✓ **Discoverable** — Are identifiable as “applications” thanks to W3C manifests and service worker registration scope allowing search engines to find them.
- ✓ **Re-engageable** — Make re-engagement easy through features like push notifications.
- ✓ **Installable** — Allow users to “keep” apps they find most useful
- ✓ **Linkable** — Easily shared via a URL and do not require complex installation.

PWA Manifest

```
{
  "short_name": "PWA",
  "name": "My PWA",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "192x192",
      "type": "image.png",
    }
  ],
  "start_url" : "./index.html",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff",
}
```

manifest.json

Service workers

- Service worker is a script that your browser runs in a background
- It doesn’t need web page or user interaction
- Provides features similar to offline work/push notification and background sync

```
if ('serviceWorker' in navigator) {
    // Register a service worker hosted at the root of the
    // site using the default scope.
    navigator.serviceWorker.register('/sw/js').then(function(registration) {
        console.log('Service worker registration succeeded:', registration);
    }).catch(function(error) {
        console.log('Service worker registration failed:', error);
    });
} else {
    console.log('Service workers are not supported.');
}
```

- Sergey Mo

Angular. Service workers

```
{
  "index": "/index.html",
  "assetGroups": [{
    "name": "app",
    "installMode": "prefetch",
    "resources": {
      "files": [
        "/favicon.ico",
        "/index.html"
      ],
      "versionedFiles": [
        "/*.bundle.css",
        "/*.bundle.js",
        "/*.chunk.js"
      ]
    }
  }, {
    "name": "assets",
    "installMode": "lazy",
    "updateMode": "prefetch",
    "resources": {
      "files": [
        "/assets/**"
```

Resources that are part of the application (including 3rd party resources)

Cache all the resources even if they are not requested

File patterns to cache

Files that contain hash information in the file name

ngsw-config.json

Angular. Service workers


```
imports: [
  BrowserModule,
  ServiceWorkerModule.register('/ngsw-worker.js', { enab
led: environment.production })
],
```

app.module.ts

Cache strategy


```
"dataGroups": [
  {
    "name": "races-api",
    "urls": ["/api/races"],
    "cacheConfig": {
      "strategy": "performance",
      "maxSize": 10,
      "maxAge": "30m",
      "timeout": "5s"
    }
  }
```

Resources that don't belong to application and don't have versions

Default strategy to prefer cache entries if they are available

Cache all requests for 30 minutes

Task #16. Service workers

1. Create new Angular application with service worker support using command *ng new SWApplication –service-worker*
2. Review service worker configuration in **src/ngsw-config.json**
3. Review files that were changed to adapt service worker functionality: .angular-cli.json and src/app/**app.module.ts**
4. Run your application: ng serve and open URL http://localhost:4200

Task #17. HTTP service

1. Add **HttpModule** to the “imports” attribute of @NgModule.
2. Create JSON file **books.json** in **src** folder that will contain list of book objects. Add it to the **“apps”-> “assets”** section in **.angular-cli.json** file.
3. Update **BookService** so it loads books from this books.json file.
4. Build Angular project using **ng build** command.Verify that **books.json** exists in the **dist** folder.

Task #18. Service workers. External API

1. Copy all the service workers related files from SWApplication to Book application.
2. Add service-worker dependency: *yarn add @angular/service-worker*
3. Update serviceWorker property: *ng set apps.0.serviceWorker=true*
4. Build your application: *ng build –prod* and run it to verify that Service Workers work properly.

Optimization

- ✓ AOT(Ahead-of-time) compilation
- ✓ Rollup

Compilation

- Compilation increases efficiency of our JS applications
- Angular 5 can generate JS VM-friendly code at runtime or build time
- It allows to perform change detection and data inline caching much faster

JIT compilation

- Transpiling TS code into JS code
- Bundling/minification
- Deployment
- JS assets are downloaded
- Angular **bootstraps**
- **JIT** process starts and generates JS code for each component
- Application rendering

JIT shortcomings

- ✓ Performance degrade
- ✓ Application includes Angular compiler and related libraries
- ✓ Template-related errors are discovered at run-time

AOT compilation

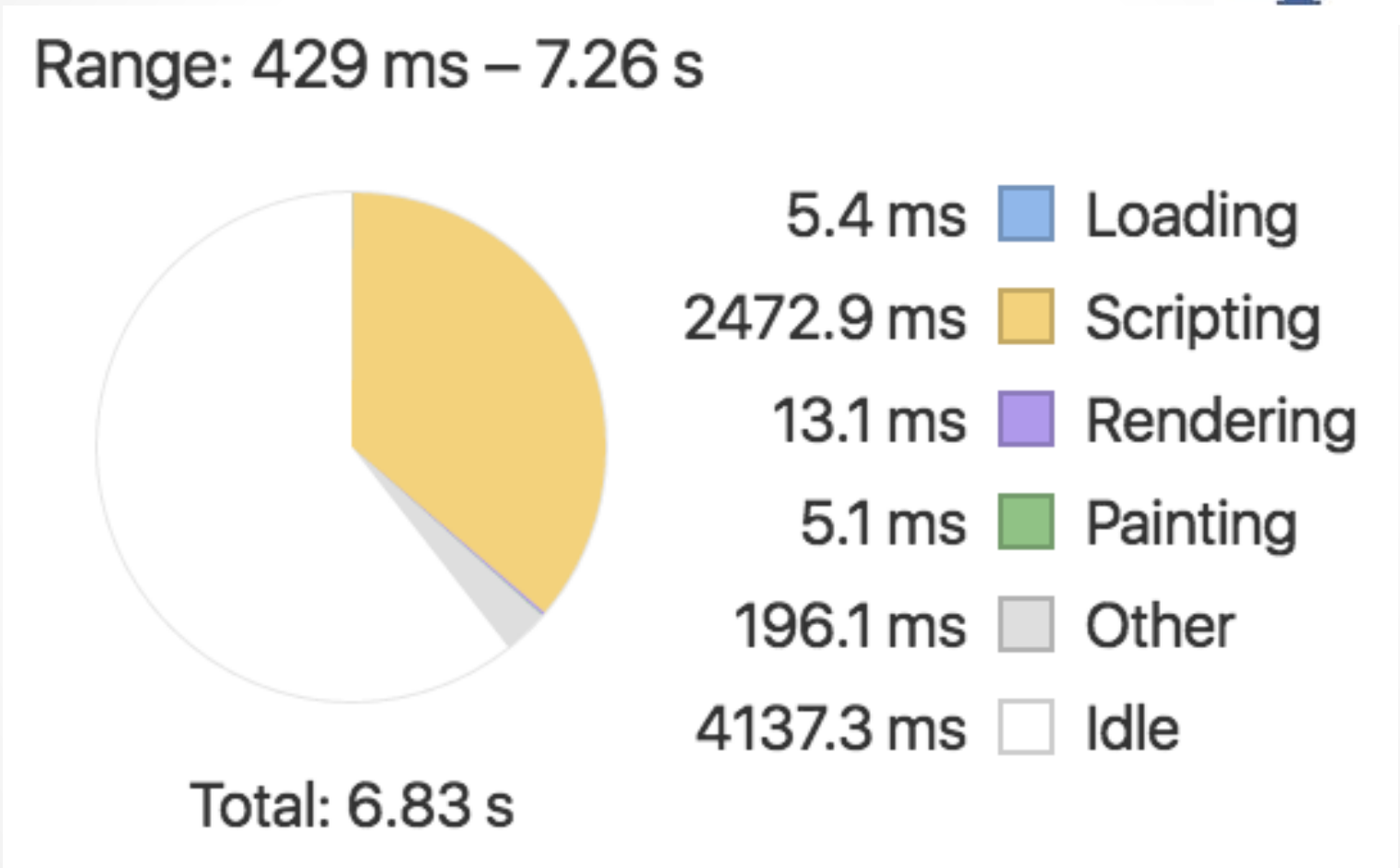
- ✓ Templates compilation into TS code
- ✓ Production mode enabled
- ✓ Transpiling TS code into JS code
- ✓ Bundling/minification
- ✓ Uglification
- ✓ Dead code elimination
- ✓ Deployment
- ✓ JS assets are downloaded
- ✓ Angular bootstraps
- ✓ Application rendering

```
ng build --prod
```

AOT benefits

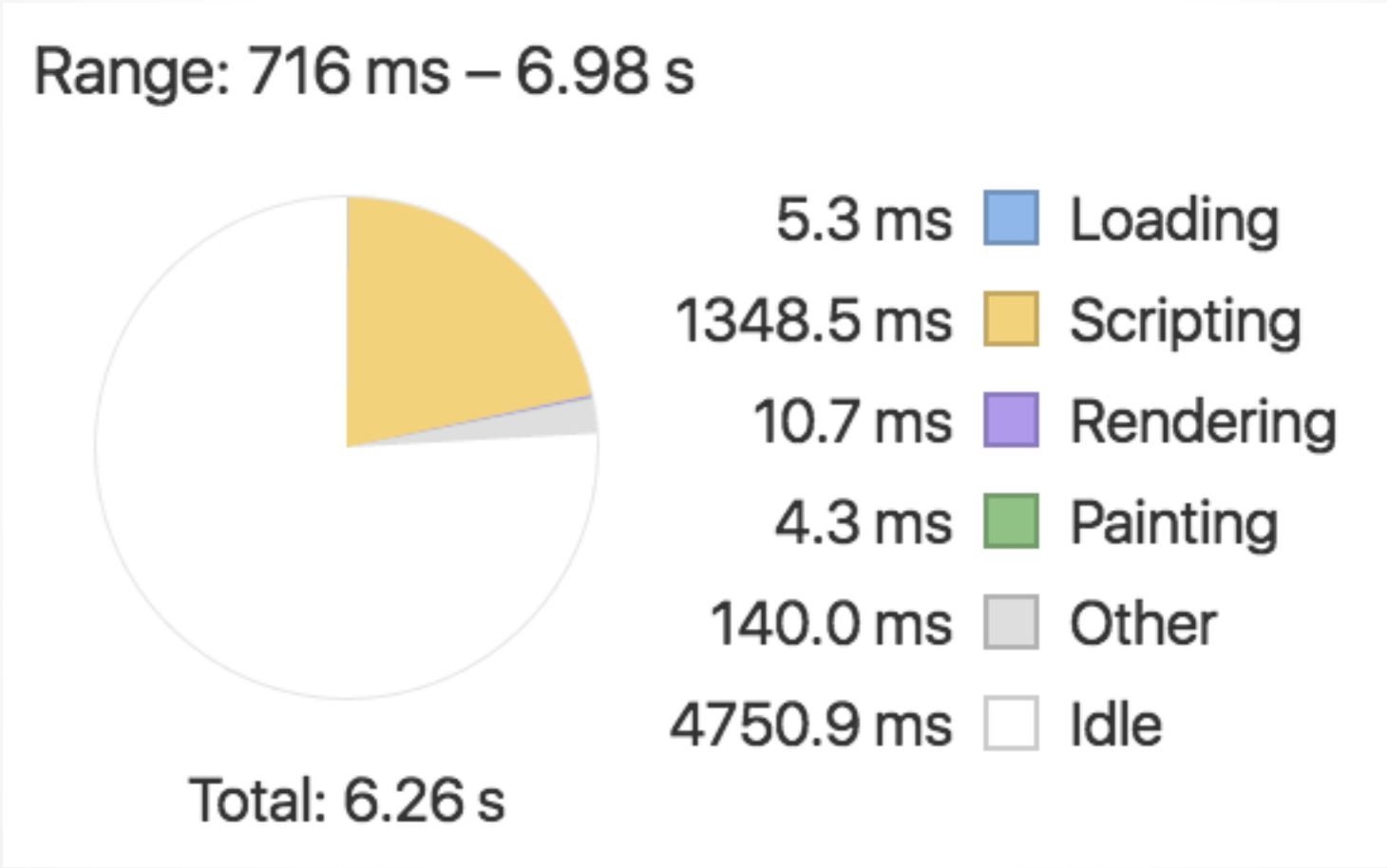
- ✓ Faster rendering
- ✓ Less asynchronous requests because of templates/styles inlining
- ✓ Smaller artifact size
- ✓ Detection of template errors at build time
- ✓ Improved security due to less JavaScript evaluation

JIT compilation



- Sergey Morenets, 2018

AOT compilation



- Sergey Morenets, 2018

Rollup

- ✓ Module bundler for JavaScript, drop-in replacement for CommonJS and AMD
- ✓ Highest level of optimization
- ✓ Resulting code after AOT will be tree-shaken and minified
- ✓ During the build Rollup statically analyzes the code, imported dependencies and remove unnecessary code

rollup.js

CommonJS


```
// import the entire utils object with CommonJS
var utils = require( 'utils' );
var query = 'Rollup';
// use the ajax method of the utils object
utils.ajax( 'https://api.example.com?search=' + query ).then( handleResponse );
```

Rollup


```
// import the ajax function with an ES6 import statement
import { ajax } from 'utils';
var query = 'Rollup';
// call the ajax function
ajax( 'https://api.example.com?search=' + query ).then( handleResponse );
```

- Sergey Morenets, 2018

Task #19. AOT optimization

1. Run **ng build --dev** and **ng build --prod** commands and compare size of the generated output in the **dist** folder.
2. Run command: *npm install @angular/compiler-cli @angular/platform-server –save*
3. Copy file **tsconfig-aot.json** from Tasks/Task folder into root project folder. Review file content.

Angular 6

- ✓ Angular Elements
- ✓ Ivy Renderer

What's new?

There are so many new things in webpack 4, that I can't list them all or this post would last forever. Therefore I'll share a few things, and to see all of **the changes from 3 to 4, please review the release notes & changelog**.

webpack 4, is FAST (up to 98% faster)!

We were seeing interesting reports of build performance from the community testing our beta, so I shot out a poll so we could verify our findings:

https://github.com/angular/angular-cli/pull/9459

- Sergey Morenets, 2018

Webpack 4

- ✓ Build (minification) parallelism
- ✓ New plugin system
- ✓ Optimized for Node v8 and ES6 modules
- ✓ Simplified configuration
- ✓ Production/development modes
- ✓ Tree sharking
- ✓ Upgrade to UglifyJS2

Webpack 4. Modes


```
// webpack.config.js
module.exports = {
  mode: 'production',
};
```

Enables development mode: minification, removal of dev code, optimized assets

Webpack 4. Modes


```
// Original code
function map(array, iteratee) {
  let index = -1;
  const length = array == null ? 0 : array.length;
  const result = new Array(length);

  while (++index < length) {
    result[index] = iteratee(array[index], index, array);
  }
  return result;
}
```

```
// Minified code
function map(n,r){let t=-1;for(const a=null==n?0:n.length,l=Array(a);++t<a;)l
```

- Sergey Morenets, 2018

Bundle-level minification


```
// comments.js
import './comments.css';
export function render(data, target) {
  console.log('Rendered!');
}
```

```
// bundle.js (part of)
"use strict";
Object.defineProperty(__webpack_exports__, "__esModule", { value: true });
/* harmony export (immutable) */ __webpack_exports__["render"] = render;
/* harmony import */ var __WEBPACK_IMPORTED_MODULE_0__comments_css__ = __
/* harmony import */ var __WEBPACK_IMPORTED_MODULE_0__comments_css_js___
__webpack_require__.n(__WEBPACK_IMPORTED_MODULE_0__comments_css__);

function render(data, target) {
  console.log('Rendered!');
}
```

```
"use strict";function t(e,n){console.log("Rendered!")}
Object.defineProperty(n,"__esModule",{value:!0}),n.render=t;var o=r(1);r.
```

Loader-level minification


```
/* comments.css */
.comment {
  color: black;
}
```

```
// minified bundle.js (part of)
exports=module.exports=__webpack_require__(1)(),
exports.push([module.i,".comment {\r\n  color: black;\r\n}",""]);
```

```
rules: [
  {
    test: /\.css$/,
    use: [
      'style-loader',
      { loader: 'css-loader', options: { minimize: true } },
    ],
  },
],
```

Tree-shaking with ES modules


```
// comments.js
export const render = () => { return 'Rendered!'; };
export const commentRestEndpoint = '/rest/comments';

// index.js
import { render } from './comments.js';
render();
```

```
// bundle.js (part that corresponds to comments.js)
(function(module, __webpack_exports__, __webpack_require__) {
  "use strict";
  const render = () => { return 'Rendered!'; };
  /* harmony export (immutable) */ __webpack_exports__["a"] = render;

  const commentRestEndpoint = '/rest/comments';
  /* unused harmony export commentRestEndpoint */
```

Image optimization

- ✓ url-loader inlines small static files into the app
- ✓ Bad practice for large files as it increase memory consumptions and parsing time

```
// index.js
import imageUrl from './image.png';
// → If image.png is smaller than 10 kB, `imageUrl` will include
// the encoded image: '…'
// → If image.png is larger than 10 kB, the loader will create a new file,
// and `imageUrl` will include its url: `/2fcd56a1920be.png`
```

Image optimization

- **Image-webpack-loader** compresses images
- Doesn't embed files so it should be pre-configured with url-loader

```
// webpack.config.js
module.exports = {
  module: {
    rules: [
      {
        test: /\.(jpe?g|png|gif|svg)$/,
        loader: 'image-webpack-loader',
        // This will apply the loader before the other ones
        enforce: 'pre',
      },
    ],
  },
};
```

Module concatenation

- ✔ Aka scope hoisting

```
// webpack.config.js (for webpack 4)
module.exports = {
  optimization: {
    concatenateModules: true,
  },
};
```

Module concatenation


```
// bundle.js (part  of)
/* 0 */
(function(module, __webpack_exports__, __webpack_require__) {

  "use strict";
  Object.defineProperty(__webpack_exports__, "__esModule", { value: true });
  var __WEBPACK_IMPORTED_MODULE_0__comments_js__ = __webpack_require__(1);
  Object(__WEBPACK_IMPORTED_MODULE_0__comments_js__["a" /* render */])();

}),
/* 1 */
(function(module, __webpack_exports__, __webpack_require__) {

  "use strict";
  __webpack_exports__["a"] = render;
  function render(data, target) {
    console.log('Rendered!');
  }

})
```

Without concatenation

Module concatenation


```
// bundle.js (part of; compiled with ModuleConcatenationPlugin)
/* 0 */
(function(module, __webpack_exports__, __webpack_require__) {

  "use strict";
  Object.defineProperty(__webpack_exports__, "__esModule", { value: true });  });

  // CONCATENATED MODULE: ./comments.js
  function render(data, target) {
    console.log('Rendered!');
  }

  // CONCATENATED MODULE: ./index.js
  render();

})
```

With concatenation

Task #20. Angular 6

1. Replace version of all the Angular-related dependencies in package.json to **6.0.0-rc.1**.
2. Run *yarn upgrade* command to update project dependencies.
3. Run application *ng serve* and open URL http://localhost:4200. Verify that application works properly.
4. Build your application: *ng build --prod*. How has bundles size changed?

Angular Universal

- ✓ Improved performance
- ✓ SEO optimization
- ✓ Social sites friendly
- ✓ Pre-rendering (both content & state)
- ✓ Event registration
- ✓ Node.js & ASP.NET server run-time support

Angular Universal. Advices

- ✓ Avoid **window**, document, navigator and other browser types as they don't exist on the server
- ✓ Avoid setTimeout() function
- ✓ Don't manipulate **DOM** structure directly. Use Renderer type for that
- ✓ Keep your directives stateless

Task #21. Angular Universal

1. Create Universal configuration in your application using command: *ng generate universal universal-app*
2. Review new files: src/tsconfig.server.json, src/main.server.ts, src/app/app.server.module.ts
3. Review changed files: package.json and .angular-cli.json
4. Build your application: *ng build --prod --app=universal-app --output-hashing=false.* Review contents of **dist-server** folder.

Jasmine vs Jest

- ✓ Based on top of Jasmine and Jasmine(Mocha)-compatible
- ✓ Watch mode
- ✓ Mock by default
- ✓ Bundled with JSDOM
- ✓ Parallel test runners (fast)
- ✓ Promise support
- ✓ Snapshot testing

Snapshot testing


```
exports[`renders correctly 1`] = `
<a
  className="normal"
  href="http://www.facebook.com"
  onMouseEnter={[Function]}
  onMouseLeave={[Function]}
>
  Facebook
</a>
`;
```

- Sergey Morenets, 2018

Snapshot testing. Unit-test


```
describe('CalcComponent', () => {
  let component: CalcComponent;
  let fixture: ComponentFixture<CalcComponent>;

  beforeEach(
    async(() => {
      const configure: ConfigureFn = testBed => {
        testBed.configureTestingModule({
          declarations: [CalcComponent]
        });
      };
```

- Sergey Morenets, 2018

Snapshot testing. Unit-test


```
    configureTests(configure).then(testBed => {
      fixture = testBed.createComponent(CalcComponent);
      component = fixture.componentInstance;
      fixture.detectChanges();
    });
  })
);

it('should snap', () => {
  expect(fixture).toMatchSnapshot();
});
```

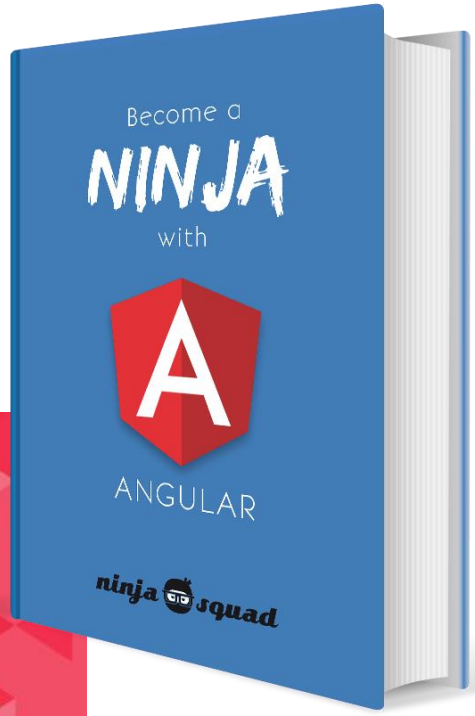
- Sergey Morenets, 2018

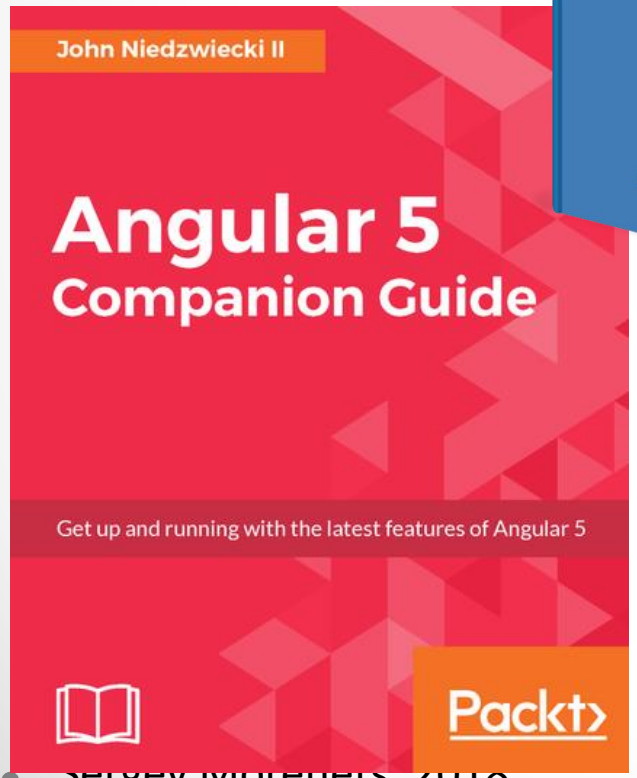
Task #22. Jest

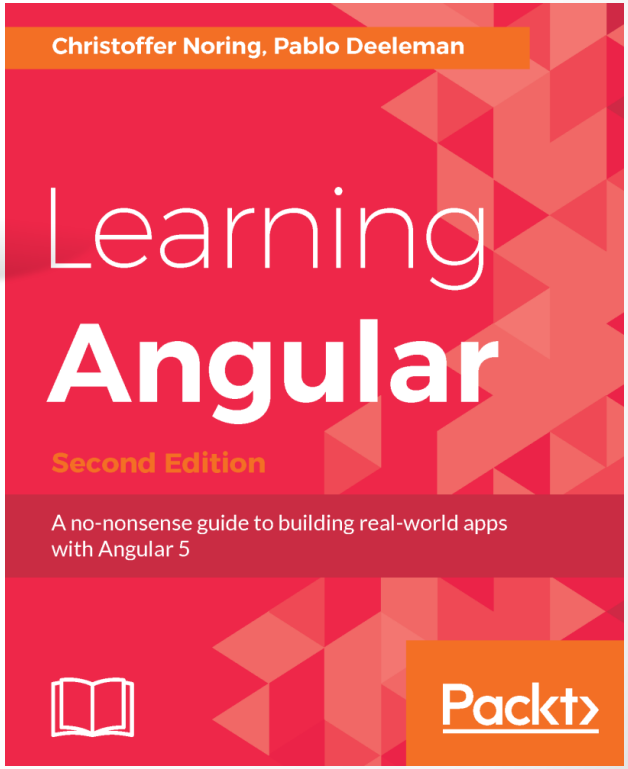
1. Install Jest and related libraries: *yarn add --dev jest jest-preset-angular @types/jest*
2. Add new entry under the **scripts** node of package.json:
3. Create new file setupJest.ts in the src folder and put this line here:

import 'jest-preset-angular';

Books







- Sergey Morenets, 2018

- Sergey Morenets, sergey.morenets@gmail.com