DEVFEST 2016 - ANGULAR 2 WORKSHOP

~~Milan Lempera @milanlempera~~
Víťa Plšek @vitaplsek
Matěj Horák @horakmat

angular.cz

O nás

Milan Lempera
@milanlempera

- php, javascript, clojure(script)

Víťa Plšek
@vitaplsek

- java, javascript

angular.cz

O nás

Matěj Horák
@horakmat

- java, javascript

Víťa Plšek
@vitaplsek

- java, javascript

ANGULAR.CZ/DEVFEST-2016

PART 1

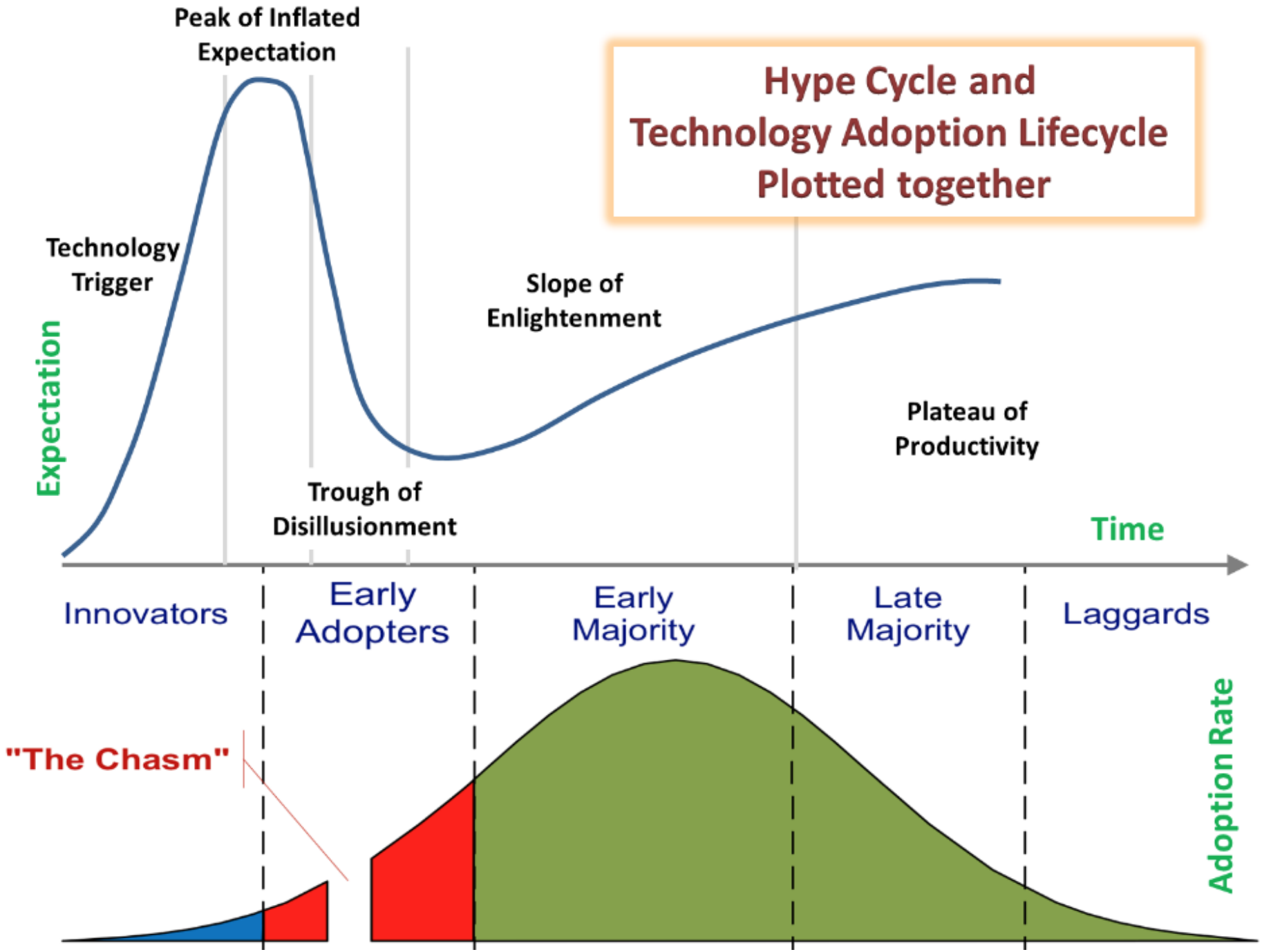
Angular JS

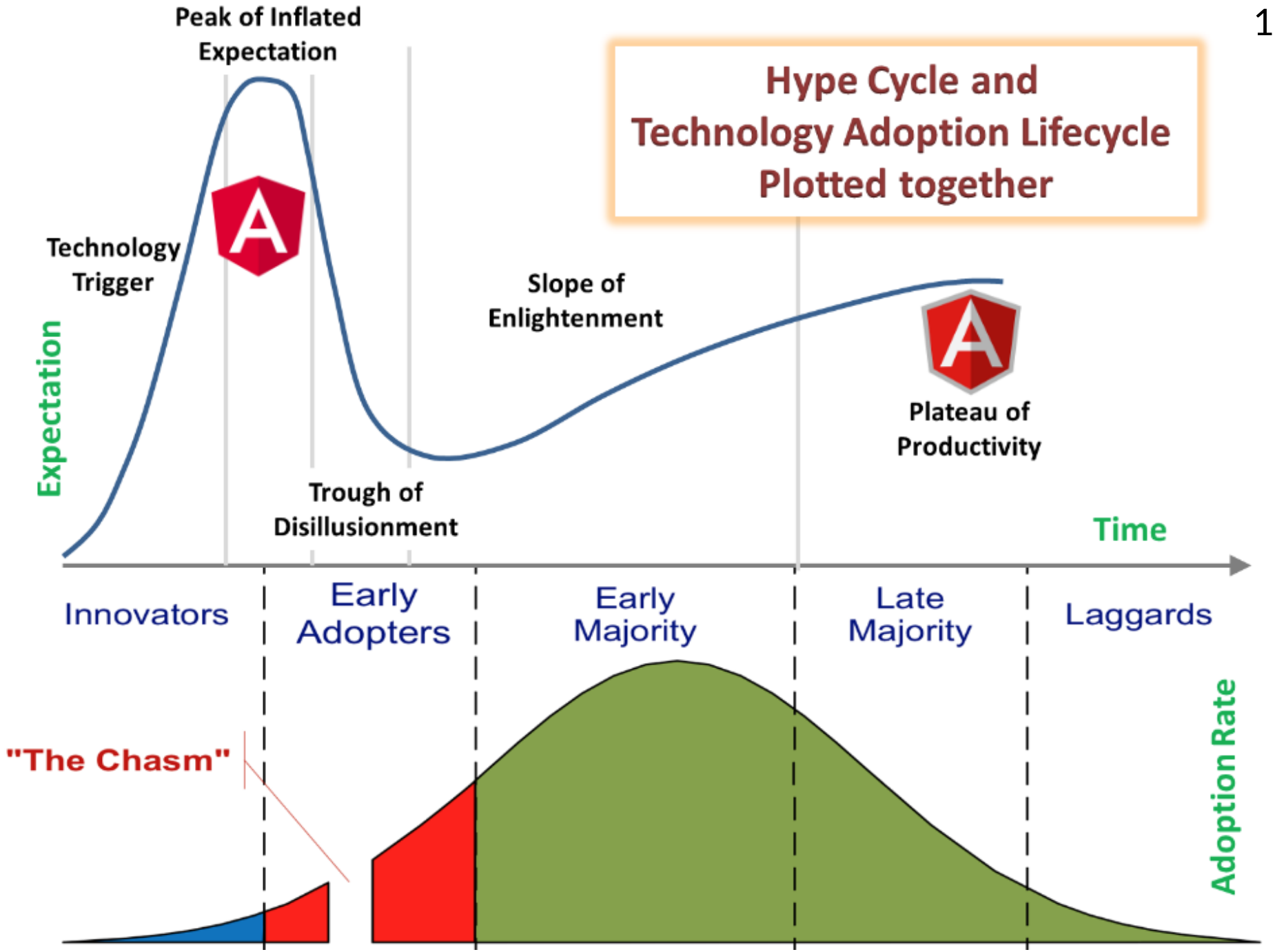
Angular 1.X

since 2009

Angular 2

since 2016


Hype Cycle and
Technology Adoption Lifecycle
Plotted together
Peak of Inflated
Expectation
Technology
Trigger
Slope of
Enlightenment
Plateau of
Productivity
Trough of
Disillusionment
Expectation
Time
Innovators
Early
Adopters
Early
Majority
Late
Majority
Laggards
"The Chasm"
Adoption Rate


Hype Cycle and
Technology Adoption Lifecycle
Plotted together
Peak of Inflated
Expectation
Technology
Trigger
Slope of
Enlightenment
Plateau of
Productivity
Trough of
Disillusionment
Expectation
Time
Innovators
Early
Adopters
Early
Majority
Late
Majority
Laggards
"The Chasm"
Adoption Rate

JavaScript vs TypeScript


```
function area(r) {
  return Math.PI * r * r;
}

let myNumber = 2;

let myString = '2'

area(myNumber);   // Ok
area(myString);   // Ok
```

```
function area(r: number): number {
  return Math.PI * r * r;
}

let myNumber = 2;

let myString = '2'

area(myNumber);   // Ok
area(myString);   // ERROR
```

Arrow functions


```
let area = r => Math.PI * r * r;

let area = (r: number) => Math.PI * r * r;
```

```
let myNumber = 2;                // typ bude odvozen
let otherNumber:number = 2;


myNumber = 'someString';         // ERROR
otherNumber = 'someString';      // ERROR
```

```
// definice typů pole
let arrayOfNumbers: number[];
```

```
// generika
let numbers: Set<number> = new Set();
numbers.add(1);      // OK
numbers.add("a");    // ERROR
```

TypeScript - Class


```
class Greeter {
    private message: string;

    constructor(message: string) {
        this.message = message;
    }

    greet() {
        return "Hello, " + this.message;
    }
}

let greeter = new Greeter("world");
```

TypeScript - Class


```
class Greeter {

    constructor(private message: string) {
    }

    greet() {
        return "Hello, " + this.message;
    }
}

let greeter = new Greeter("world");
```

TypeScript - Interface


```
interface Speaker {
  id?: number;
  name: string;
}
```

"structural subtyping" / "duck typing"

záleží jen na struktuře objektu

```
let speaker1: Speaker = { // OK
  id: 1,
  name: 'Petr Novák'
};

let speaker2: Speaker = { // OK - id není povinné
  name: 'Petr',
  surname: 'Novák'
};

let speaker3: Speaker = { // ERROR - neobsahuje name
  id: 11,
  surname: 'Novák'
};
```

TypeScript - Dekorátory


```
function enableLogging(target) {
  target.loggingEnable = true;
}
```

```
@enableLogging
class Calculator { }
```

```
const calculator = new Calculator();
calculator.loggingEnable // true
```

můžete dekorovat i vlastnosti a metody

Struktura aplikace

- Angular staví aplikaci jako strom komponent
- app-root
  - app-task-list
  - app-task-detail
    - app-task-info
    - app-task-actions

Angular CLI

- devstack
- scafolding

Vygenerování komponenty

```
ng generate component tasks/task-list
```

```
installing component
  create src/app/tasks/task-list/task-list.component.css
  create src/app/tasks/task-list/task-list.component.html
  create src/app/tasks/task-list/task-list.component.spec.ts
  create src/app/tasks/task-list/task-list.component.ts
```

Komponenta


```
import { Component, OnInit } from '@angular/core';


@Component({
  selector: 'app-task-list',
  templateUrl: './task-list.component.html',
  styleUrls: ['./task-list.component.css']
})
export class TaskListComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Použití komponenty v šabloně

```
<app-task-list></app-task-list>
```

Attributy a metody komponenty


```
import { Component, OnInit } from '@angular/core';
import { Task } from './task';

@Component({
  selector: 'app-task-list',
  templateUrl: './task-list.component.html',
  styleUrls: ['./task-list.component.css']
})
export class TaskListComponent implements OnInit {

  tasks: Task[];
  selectedTask: Task;

  getTotalEstimation() {
    return ...
  }

  ...

}
```

Šablona komponenty

Interpolace

```
<h1>{{selectedTask.title}}</h1>
```

```
<span>Total Estimation: {{getTotalEstimation()}}</span>
```

Cykly, podmínky

```
<ul>
  <li *ngFor="let task of tasks">

    {{task.title}}
    <span *ngIf="task.isOverdue">Overdue !</span>

  </li>
</ul>
```

Event binding


```
<button type="button" (click)="upVote(task)">Vote</button>

<span [class]="task.type"></span>
```

```
<input type="text" [(ngModel)]="task.title" />
```

Také u vlastních komponent

```
<app-task-list [tasks]="overdueTasks"
               (onUpVote)="voteForTask($event)">
</app-task-list>
```

Rozhraní komponenty


```
@Component({
  selector: 'app-task-list',
  templateUrl: './task-list.component.html',
  styleUrls: ['./task-list.component.css']
})
export class TaskListComponent implements OnInit {

  @Input() tasks: Task[];
  @Output() onUpVote = new EventEmitter<Task>();

  upVote(task) {
    this.onUpVote.emit(task);
  }

  ...
}
```

```
<app-task-list [tasks]="overdueTasks"
               (onUpVote)="voteForTask($event)">
</app-task-list>
```

Vyzkoušejte si to

PART 2

Moduly

- Moduly vystavují komponenty a udávají závislost na jiné moduly

```
@NgModule({
  declarations: [
    AppComponent,
    TaskDetailComponent,
    TaskListComponent,
    Error404Component
  ],

  imports: [
    BrowserModule,
    HttpModule,
  ],

  bootstrap: [AppComponent]
})
export class AppModule {}
```

Routing

- transparentní chování aplikace v prostředí webu
- stav a URL adresa jsou vzájemně provázané
  - historie (vpřed, zpět)
  - možnost poslat/uložit URL

Router Module


```
RouterModule.forRoot([
    {path: '', component: TaskListComponent},
    {path: 'task/:id', component: TaskDetailComponent},
    {path: '**', component: Error404Component}
  ]
)
```

Vykreslení v šabloně

```
<html>
  <head>
  </head>
  <body>

    <router-outlet></router-outlet>

  </body>
</html>
```

Moduly a routing


```
@NgModule({
  imports: [
    RouterModule.forRoot([
        {path: '', component: TaskListComponent},
        {path: 'task/:id', component: TaskDetailComponent},
        {path: '**', component: Error404Component}
      ]
    )
  ],
  exports: [
    RouterModule
  ]
})
export class AppRoutingModule {}
```

Moduly a routing


```
@NgModule({
  declarations: [
    AppComponent,
    TaskDetailComponent,
    TaskListComponent,
    Error404Component
  ],

  imports: [
    BrowserModule,
    HttpModule,
    AppRoutingModule    // routing modul
  ],

  bootstrap: [AppComponent]
})
export class AppModule {}
```

Přechod mezi routami


```
<a routerLink="/tasks">Seznam</a>

<a [routerLink]="['task', task.id]">Detail</a>
```

Injectables


```
import { Task } from "./task";
import { Injectable } from "@angular/core";

@Injectable()
export class TaskDataService {

  getTasks(): Task[]{
    ...
  }

}
```

Začlenení do modulu


```
@NgModule({
  declarations: [
    AppComponent,
    TaskDetailComponent,
    TaskListComponent,
    Error404Component
  ],

  imports: [
    BrowserModule,
    HttpModule,
    AppRoutingModule
  ],

  providers: [TaskDataService],   // injektovatelná závislost

  bootstrap: [AppComponent]
})
export class AppModule {}
```

Injektáž do komponenty

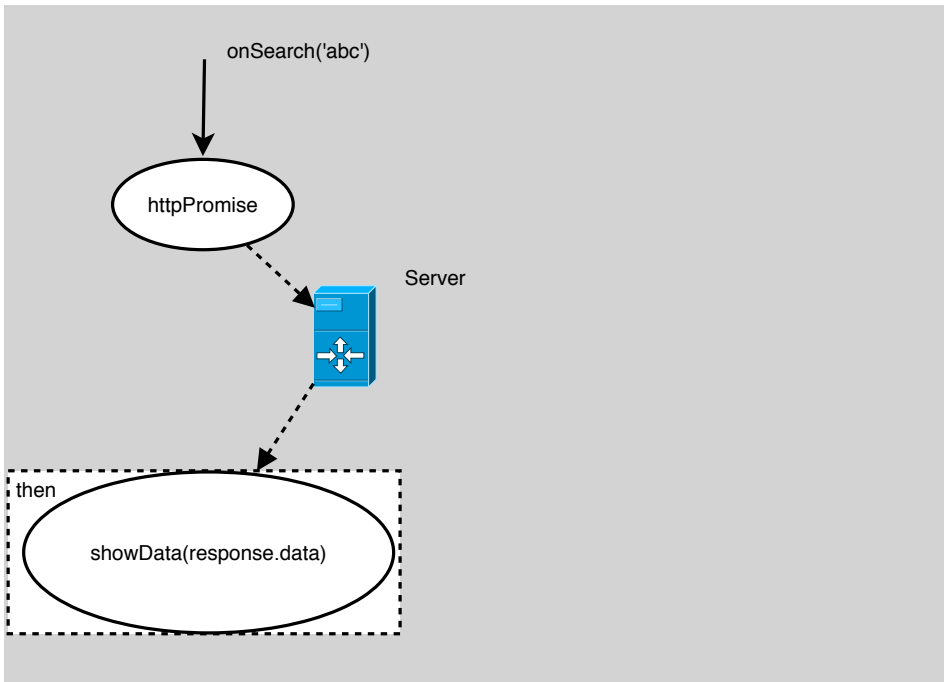

```
export class TaskListComponent implements OnInit {

  constructor(private taskDataService: TaskDataService) {
  }

  ngOnInit(): void {
    this.tasks = this.taskDataService.getTasks();
  }

  ...
}
```

Získání parametrů routy


```
export class TaskDetailComponent implements OnInit {

  constructor(private route: ActivatedRoute) {
  }

  ngOnInit(): void {
    let idParam = this.route.snapshot.params['id'];
    let id = parseInt(idParam);

    ...
  }

  ...
}
```

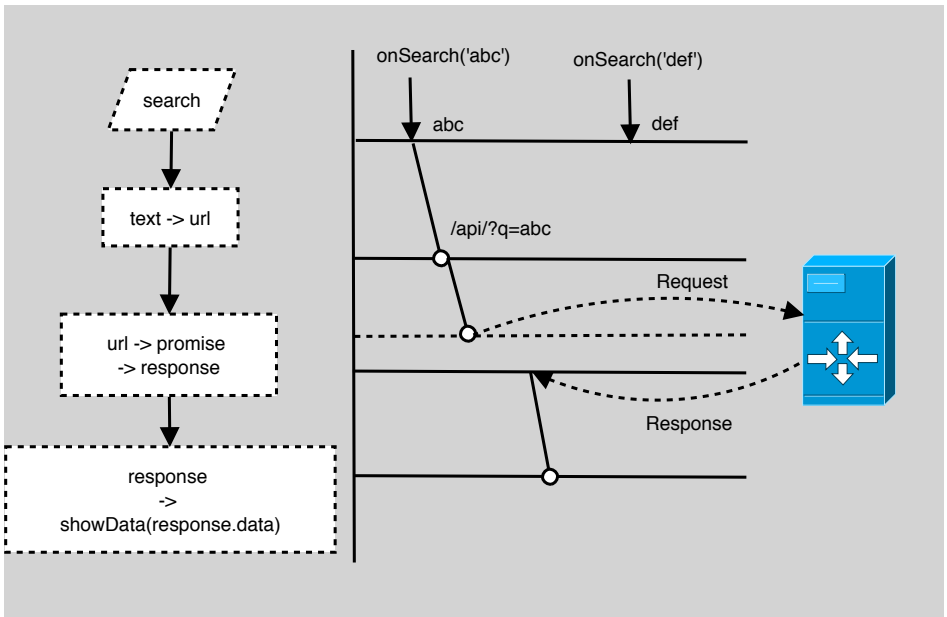
Asynchronní data


```
function onSearch(text) {

  let url = '/api/?q=' + text;

  $http
  .get(url)
  .then(function(response) {
    showData(response.data);
  });

}
```



Asynchronní data


```
let search = new Subject()
  .map((text) => '/api/?q=' + text)
  .flatMap((url) => $http.get(url))
  .subscribe((response) {
    showData(response.data);
  })

function onSearch(text) {

  return search.next(text);

}
```



Observable + subscribe


```
@Component({
  ...
})
export class SomeComponent implements OnInit {

  titleObservable: Observable<string>;

  title: string;

  ngOnInit() {

    this.titleObservable
      .subscribe((title) => this.title = title);
  }

}
```

```
{{ title }}
```

Pipe


```
@Component({
  ...
})
export class SomeComponent  {

  title: string = "Angular"
}
```

```
{{ title }}                -> Angular

{{ title | uppercase }}    -> ANGULAR
```

vestavěné: date, uppercase, lowercase, currency, percent, async

Observable + async pipe


```
@Component({
  ...
})
export class SomeComponent implements OnInit {

  titleObservable: Observable<string>;

}
```

```
{{ titleObservable | async }}
```

Vyzkoušejte si to

PART 3 - BONUS

Angular 2 Forms

- template based

  podobné jako v Angular 1
- reactive

  změny data jako Observable

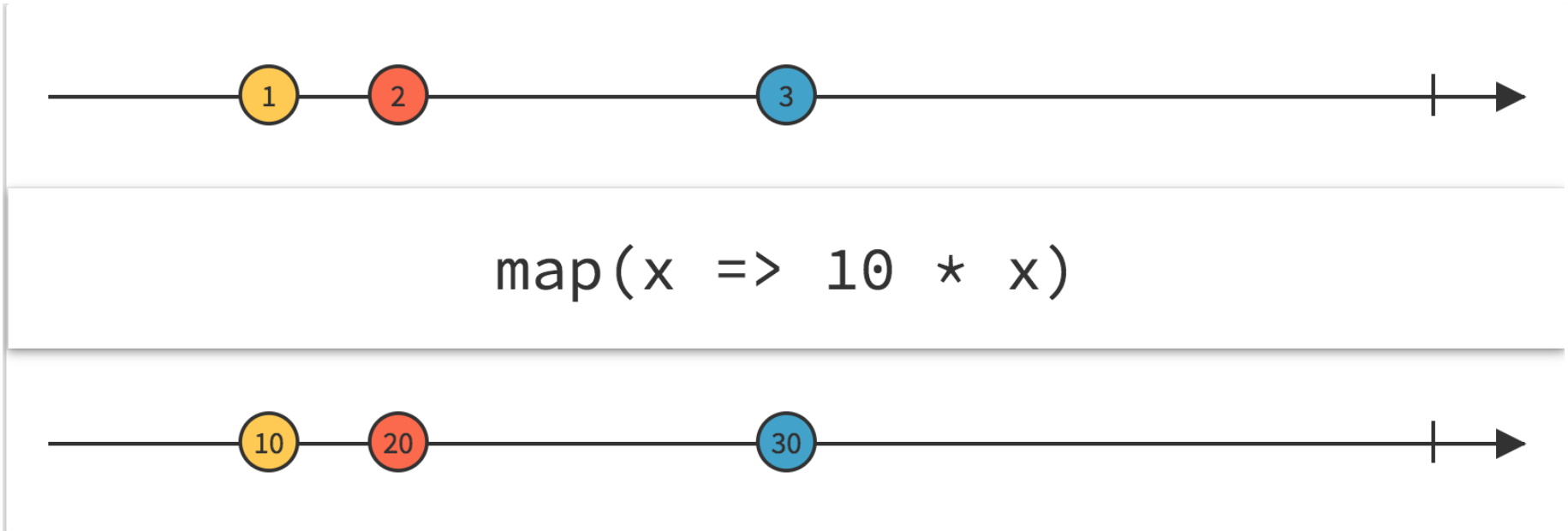
Angular 2 Reactive Forms


```
@Component()
export class SearchComponent implements OnInit {

  searchControl = new FormControl();

  ngOnInit() {
    this.searchControl.valueChanges.subscribe(value => {
      // do something with value here
    });
  }
}
```

```
<input type="search" [formControl]="seachControl">
```

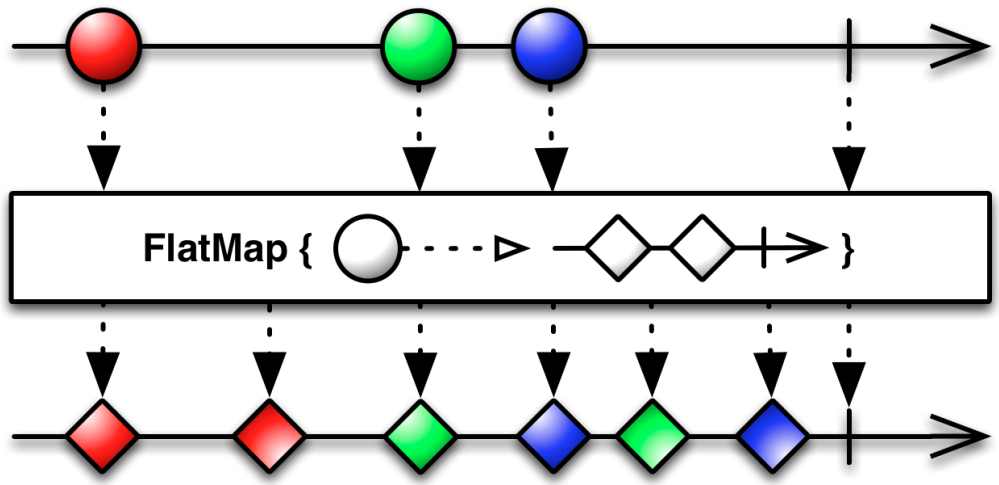
Observables - operátory

RxJS .map



rxmarbles.com

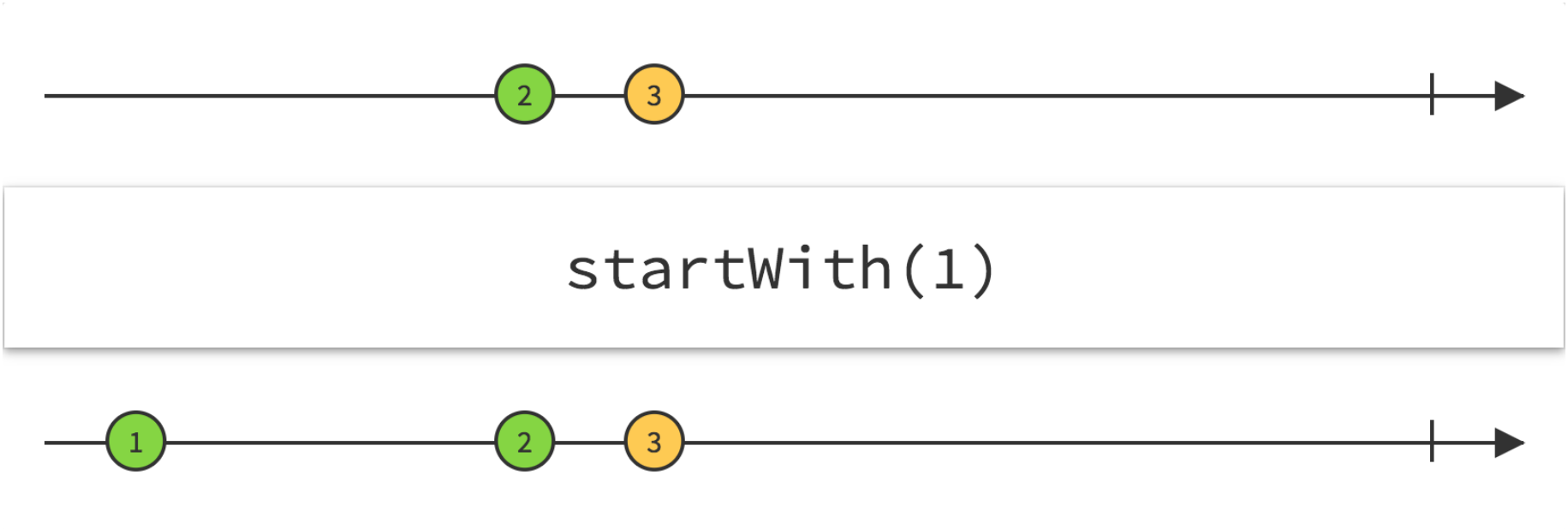
RxJS .flatMap



Podobné jako map, ale místo Observable/Promise vrací rovnou hodnoty, nemusíte tedy zanořovat subscribe/then v map.

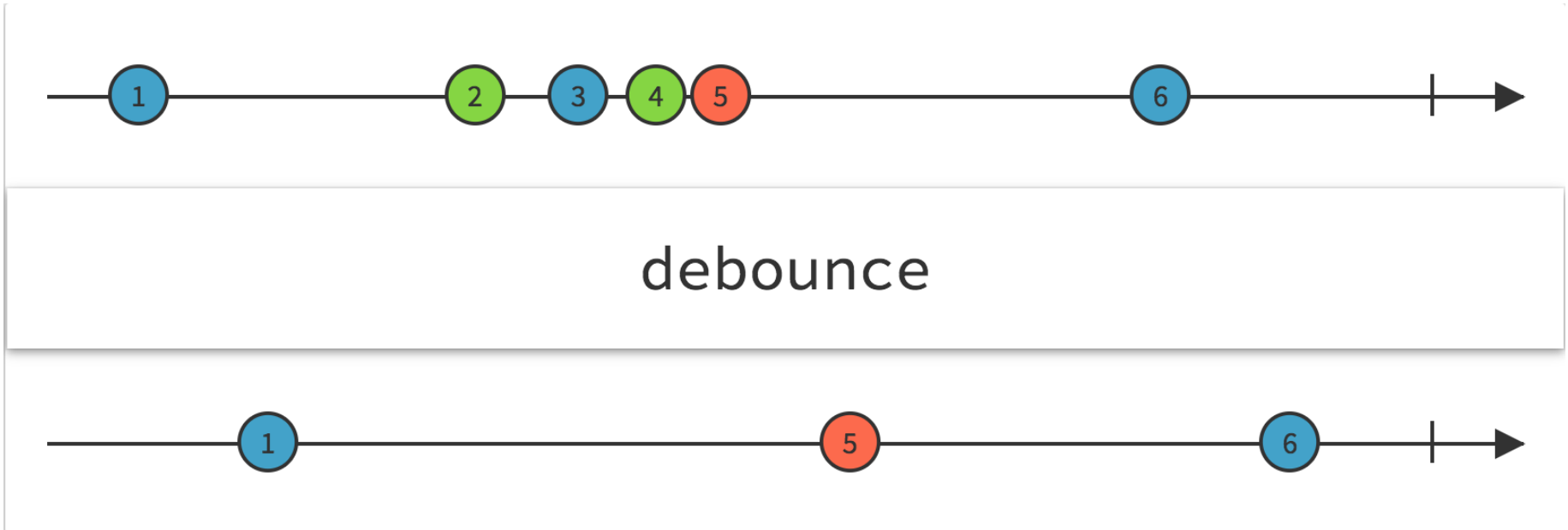
reactivex.io/documentation/operators/flatmap.html

RxJS .startWith



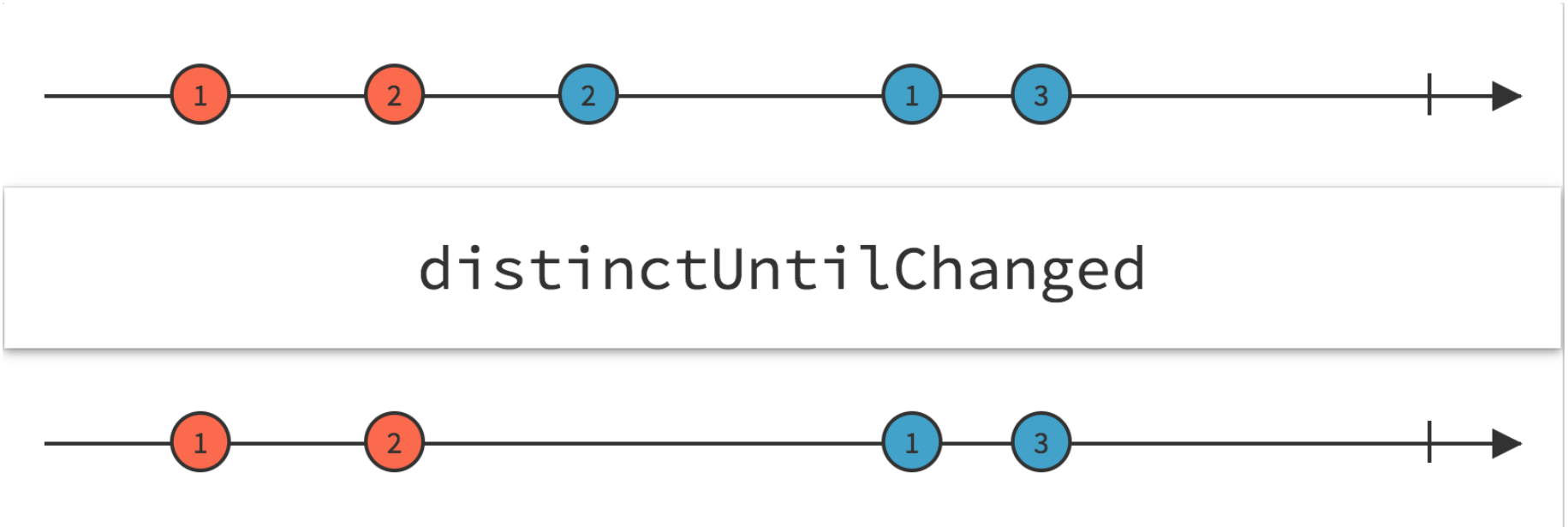
rxmarbles.com

RxJS .debounce



rxmarbles.com

RxJS .distinctUntilChanged



rxmarbles.com

a mnoho dalších - reactivex.io/rxjs/manual/overview.html#categories-of-operators

Vyzkoušejte si to

ANGULAR.CZ/DEVFEST-2016