

ANGULAR 2 WORKSHOP

Milan Lempera @milanlempera

Víťa Plšek @vitaplsek

Matěj Horák @horakmat

angular.cz

Matěj Horák
@horakmat

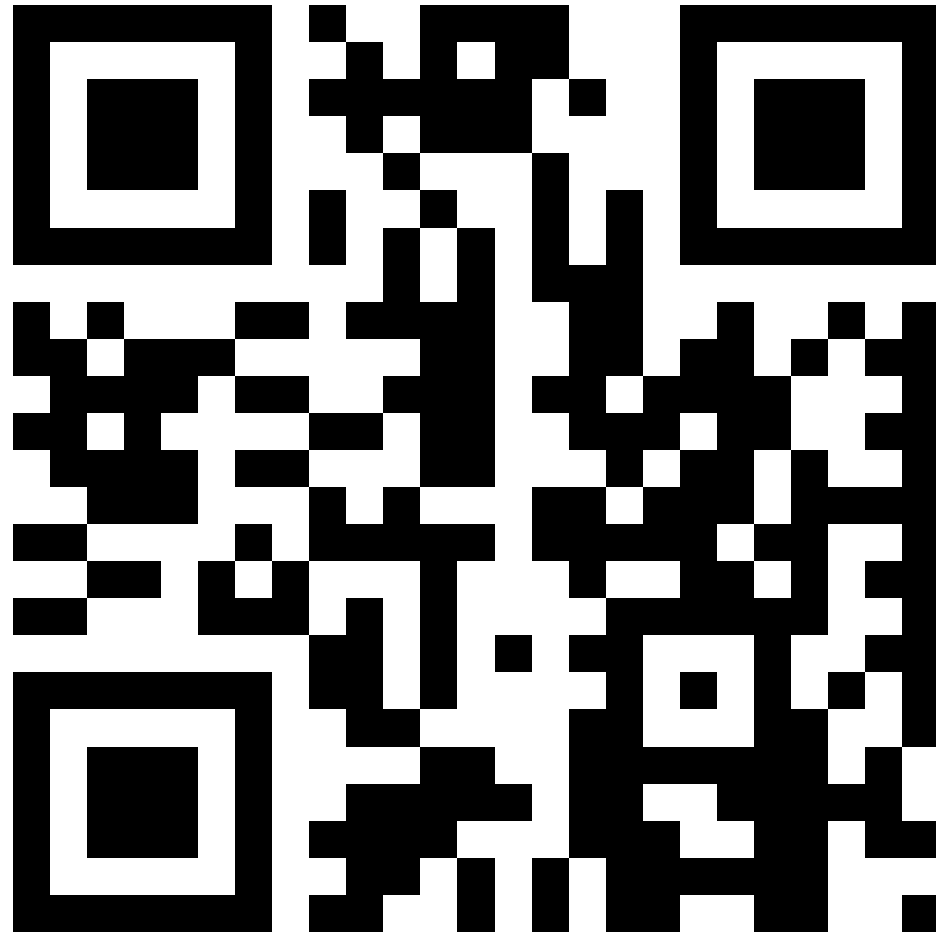
- java, javascript

Víťa Plšek
@vitaplsek

- java, javascript

angular.cz/gdg-brno-a2

0-2



JAK TO BUDE PROBÍHAT

0-3

- povídání
- cvičení
- ukázka
- přestávka

3x po cca hodině

PART 1

1-0

Angular JS

1-1



Angular 1.X

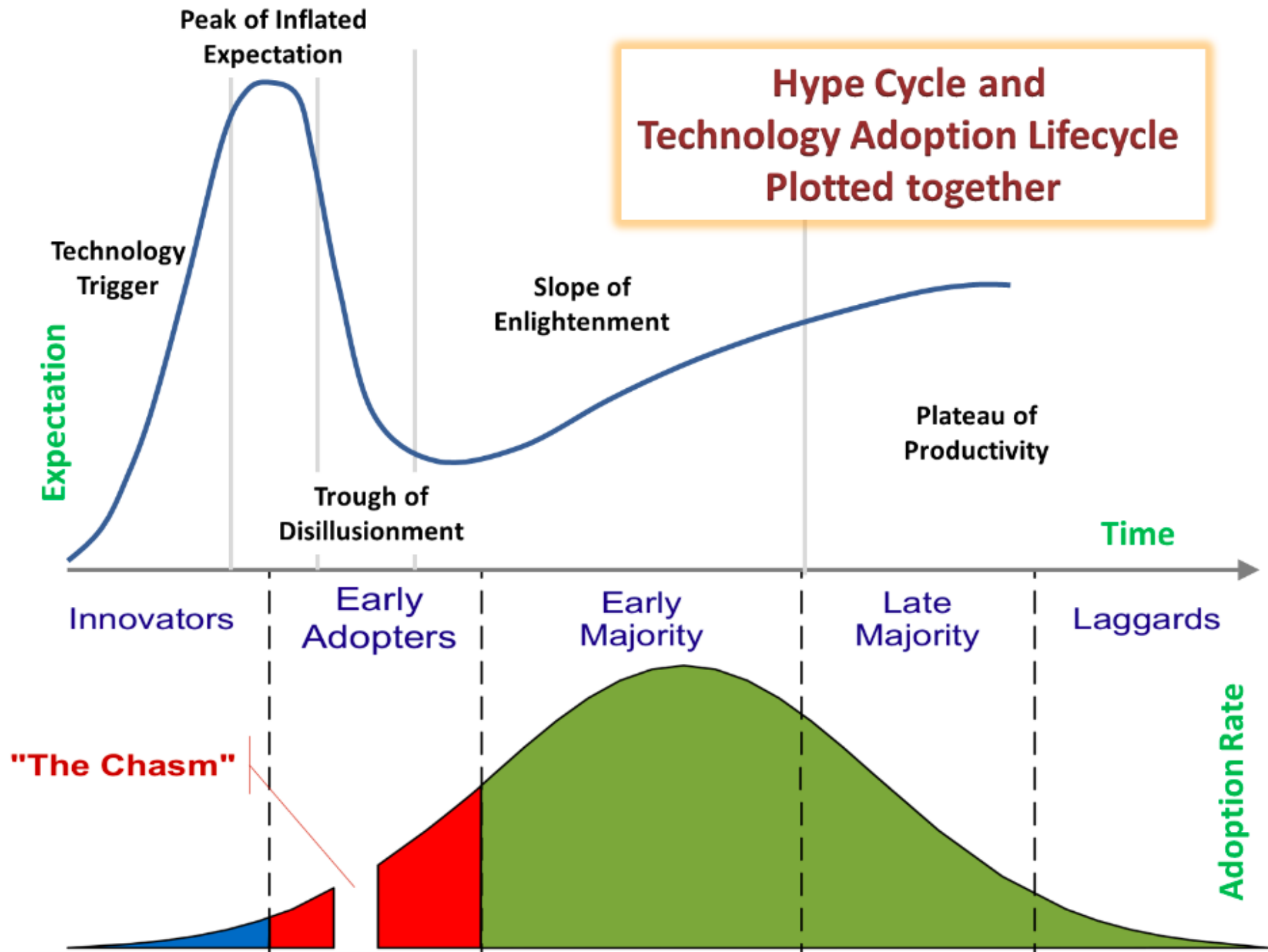
since 2009

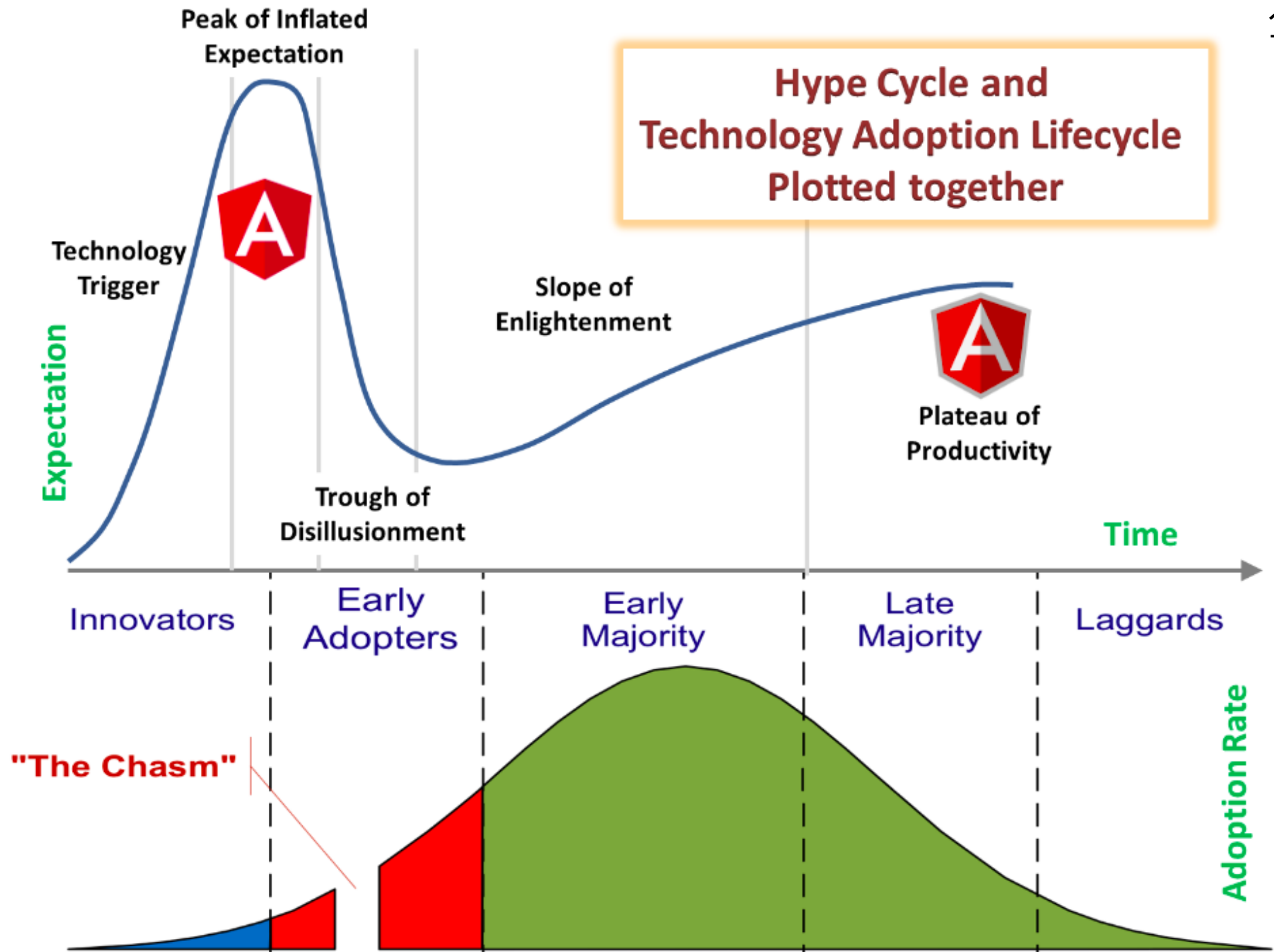


Angular 2

since 2016

Hype Cycle and Technology Adoption Lifecycle Plotted together





JavaScript vs TypeScript

1-4

```
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

1-5

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

```
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 typu pole
let arrayOfNumbers: number[];
```

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

TypeScript - Class

1-7

```
class Greeter {  
    private message: string;  
  
    constructor(message: string) {  
        this.message = message;  
    }  
  
    greet() {  
        return "Hello, " + this.message;  
    }  
}  
  
let greeter = new Greeter("world");
```

TypeScript - Class

1-8

```
class Greeter {  
    constructor(private message: string) {  
    }  
  
    greet() {  
        return "Hello, " + this.message;  
    }  
}  
  
let greeter = new Greeter("world");
```

TypeScript - Interface

1-9

```
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

1-10

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

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

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

můžete dekorovat i vlastnosti a metody

Angular CLI

1-11

Oficiální devstack od autorů Angularu

Instalace

```
npm install -g @angular/cli
```

Vytvoření nové aplikace

```
ng new my-new-application  
  
...  
  
cd my-new-application  
ng serve
```


- build

```
ng build  
ng build --prod
```

- scaffolding

```
ng generate ...
```

Test, Lint, Format

Struktura aplikace

1-13

- Angular staví aplikaci jako strom komponent

[Seznam piv](#) / [Bakalář \(Rakovník\)](#) / [hodnocení](#)

Bakalář

Uváděné hodnoty

Alkohol: 0.02%

Stupeň: Nealko°

světlé pivo

[O pivovaru](#)[Hodnocení](#)

Pavel Pivorád ★★★★★

10. 11. 2016 15:18:49

Mě chutná každé české pívko.

Jméno:

Hodnocení: ★ ★ ★ ★ ★

Popis:

Ohodnotit

Smazat

Bakalář

Uváděné hodnoty

Alkohol: 0.02%

Stupeň: Nealko°

světlé pivo

[O pivovaru](#)[Hodnocení](#)

Pavel Pivorád ★★★★★

10. 11. 2016 15:18:49

Mě chutná každé české pívko.

Jméno:

Hodnocení: ★ ★ ★ ★ ★

Popis:

[Ohodnotit](#)[Smazat](#)

Bakalář

Uváděné hodnoty

Alkohol:	0.02%
Stupeň:	Nealko°

světlé pivo

[O pivovaru](#)[Hodnocení](#)

Pavel Pivorád ★★★★★

10. 11. 2016 15:18:49

Mě chutná každé české pivo.

Jméno:

Hodnocení: ☆ ☆ ☆ ☆ ☆

Popis:

Ohodnotit

Smazat

Bakalář

Uváděné hodnoty

Alkohol: 0.02%

Stupeň: Nealko°

světlé pivo

[O pivovaru](#)[Hodnocení](#)

Pavel Pivorád ★★★★★

10. 11. 2016 15:18:49

Mě chutná každé české pivo.

Jméno:

Hodnocení: ☆ ☆ ☆ ☆ ☆

Popis:

Ohodnotit

Smazat

Bakalář

Uváděné hodnoty

Alkohol: 0.02%

Stupeň: Nealko°

světlé pivo

O pivovaru

Hodnocení

Pavel Pivorác 

10. 11. 2016 15:18:49

Mě chutná každé české pívko.

Jméno:

Hodnocení 

Popis:

Ohodnotit

Smazat

[Seznam piv](#) / [Bakalář \(Rakovník\)](#) / [hodnocení](#)

Bakalář

Uváděné hodnoty

Alkohol: 0.02%

Stupeň: Nealko°

světlé pivo

[O pivovaru](#)[Hodnocení](#)

Pavel Pivorád ★★★★★

10. 11. 2016 15:18:49

Mě chutná každé české pívko.

Jméno:

Hodnocení: ★ ★ ★ ★ ★

Popis:

Ohodnotit

Smazat

Struktura aplikace

1-20

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

```
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

1-22

```
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 ...
  }

  ...
}
```

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 / property binding

1-24

```
<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>
```

```
@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>
```

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
```

Vyzkoušejte si to

1-27

PART 2

2-0

- Moduly vystavují komponenty a udávají závislost na jiné moduly

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

  imports: [
    BrowserModule,
    HttpClientModule,
  ],

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

Routing

2-2

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

Router Module

2-3

```
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

2-4

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

Moduly a routing

2-5

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

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

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

Přechod mezi routami

2-6

```
<a routerLink="/tasks">Seznam</a> <!-- www.app.cz/tasks -->  
<a [routerLink]="['task', task.id]">Detail</a> <!-- www.app.cz/task/42 -->
```

Injectables

2-7

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

@Injectable()
export class TaskDataService {

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


Začlenení do modulu

2-8

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

  imports: [
    BrowserModule,
    HttpClientModule,
    AppRoutingModule
  ],

  providers: [TaskDataService],    // injektovatelná závislost

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

Injektáž do komponenty

2-9

```
export class TaskListComponent implements OnInit {  
  
  constructor(private taskDataService: TaskDataService) {  
  }  
  
  ngOnInit(): void {  
    this.tasks = this.taskDataService.getTasks();  
  }  
  
  ...  
}
```

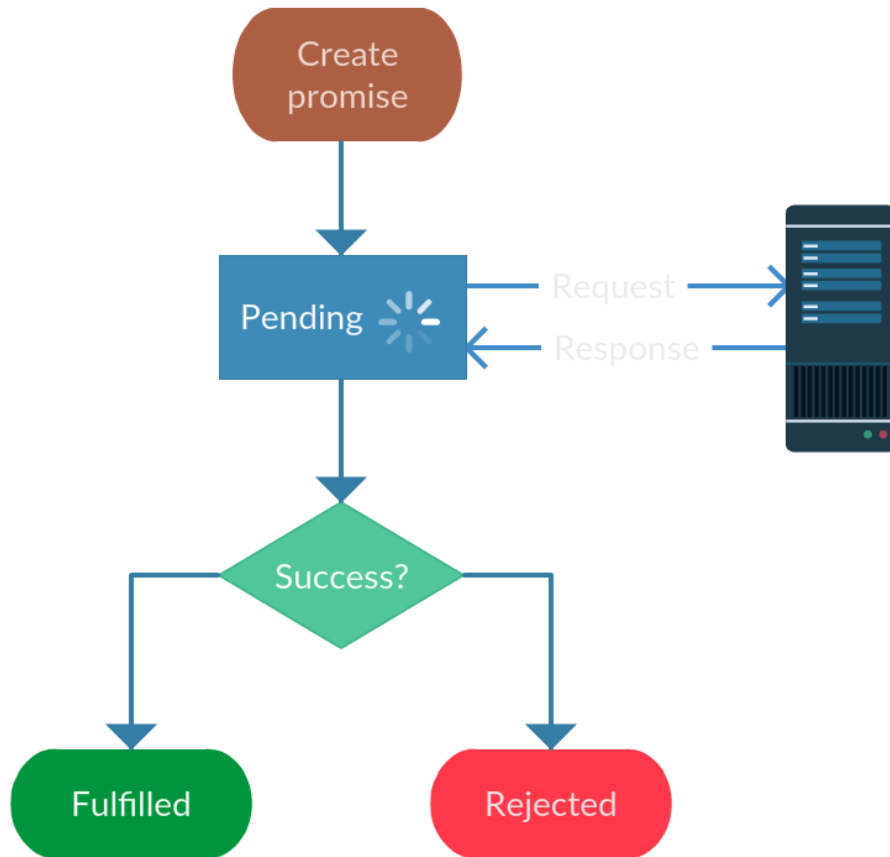
Získání parametrů routy

2-10

```
export class TaskDetailComponent implements OnInit {  
  
  constructor(private route: ActivatedRoute) {  
  }  
  
  ngOnInit(): void {  
    let idParam = this.route.snapshot.params['id'];  
    let id = parseInt(idParam);  
  
    ...  
  }  
  
  ...  
}
```

Observable vs Promise

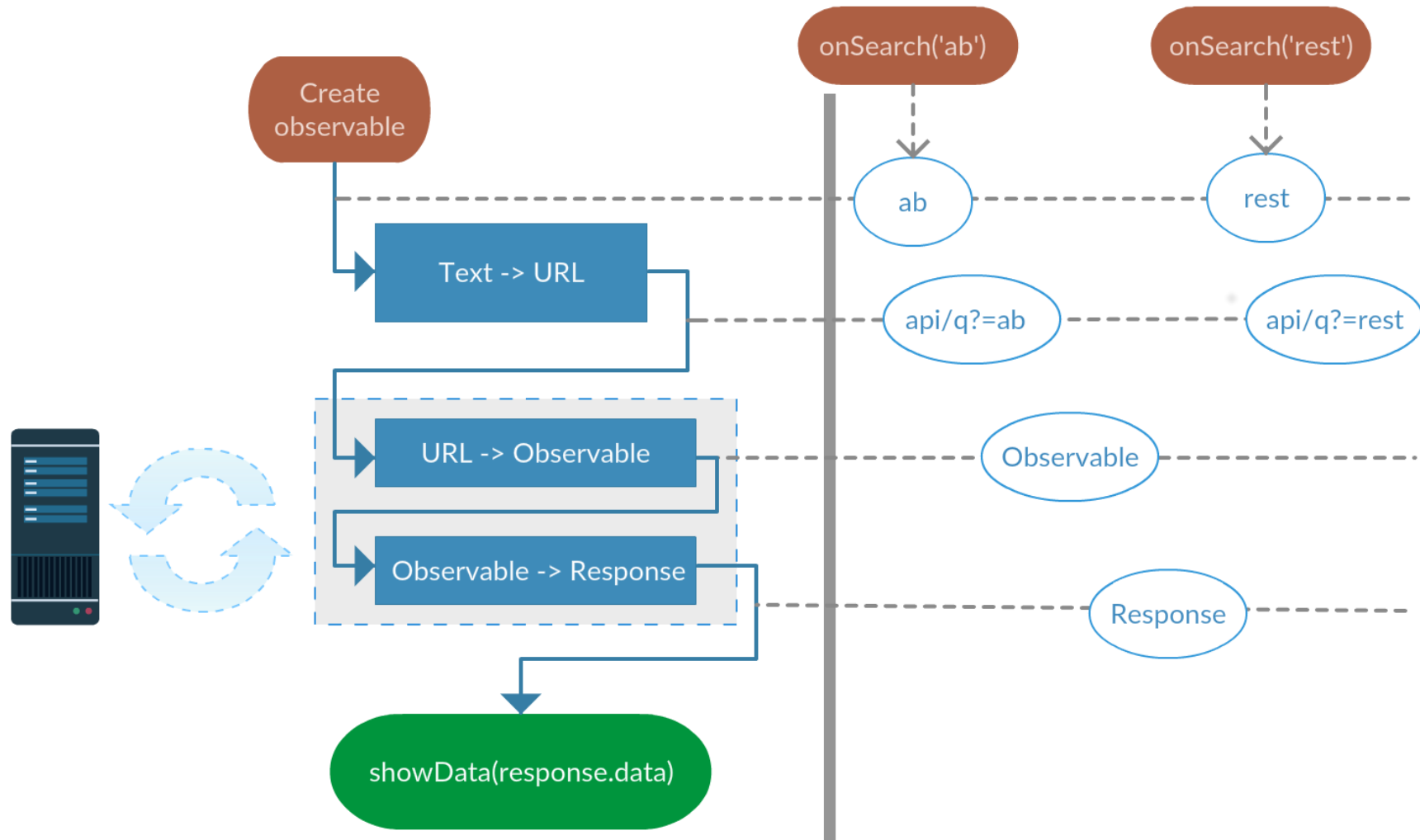
2-11



```
function onSearch(text) {  
  
    let url = '/api/?q=' + text;  
  
    getDataFrom(url)  
        .then(function(response) {  
            showData(response.data);  
        })  
        .catch(function (error) {  
            // Handle error  
        })  
}
```

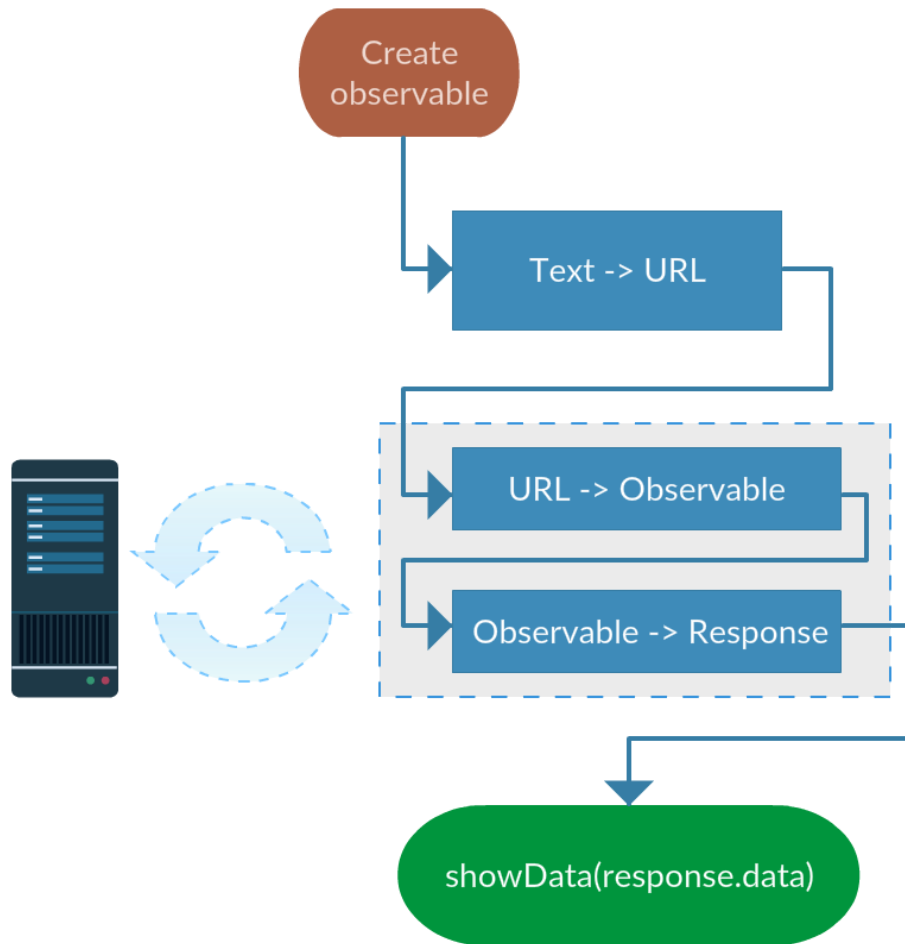
Observable vs Promise

2-12



Observable vs Promise

2-13



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

function onSearch(text) {
  return search.next(text);
}
```

Observable + subscribe

2-14

```
@Component({  
  ...  
})  
export class SomeComponent implements OnInit {  
  
  task: Task;  
  ngOnInit() {  
  
    this.taskDataService.getMainTask()  
      .subscribe(task => this.task = task);  
  }  
  
}
```

```
{{ title }}
```

Pipe

2-15

```
@Component({  
  ...  
})  
export class SomeComponent {  
  
  title: string = "Angular"  
}
```

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

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

Observable + async pipe

2-16

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

  tasks: Observable<Tasks[]>;

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

```
<ul>
  <li *ngfor="let task of tasks | async">
    {{task.title}}
  </li><li>
</li></ul>
```

Vyzkoušejte si to

2-17

PART 3 - BONUS

3-0

Angular 2 Forms

3-1

- template based - FormsModule
podobné jako v Angular 1
- reactive - modul ReactiveFormsModule
změny data jako Observable

Angular 2 Reactive Forms

3-2

```
@Component()  
export class SearchComponent implements OnInit {  
  
  searchControl = new FormControl();  
  
  ngOnInit() {  
    this.searchControl.valueChanges.subscribe(value => {  
      // do something with value here  
    });  
  }  
}
```

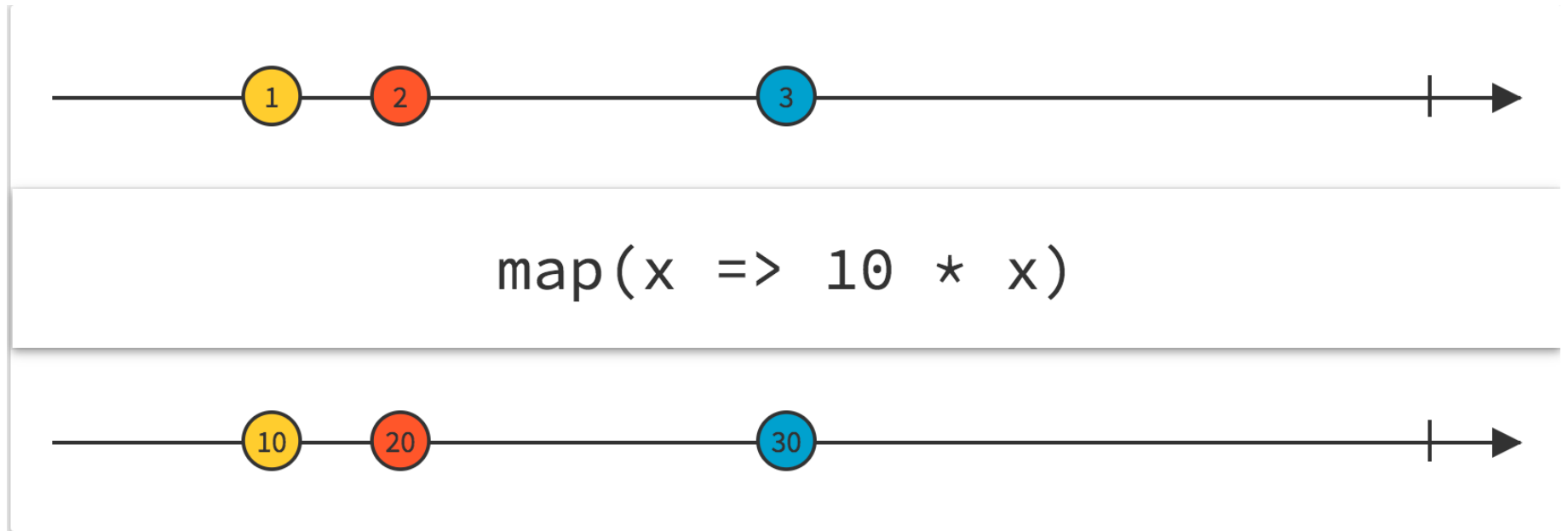
```
<input type="search" [formControl]="searchControl">
```

Observables - operátory

3-3

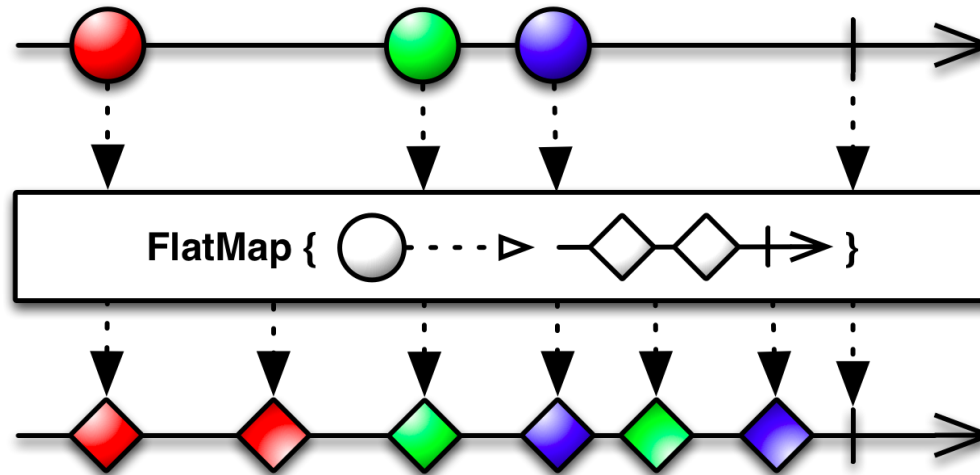
RxJS .map

3-4



RxJS .flatMap

3-5

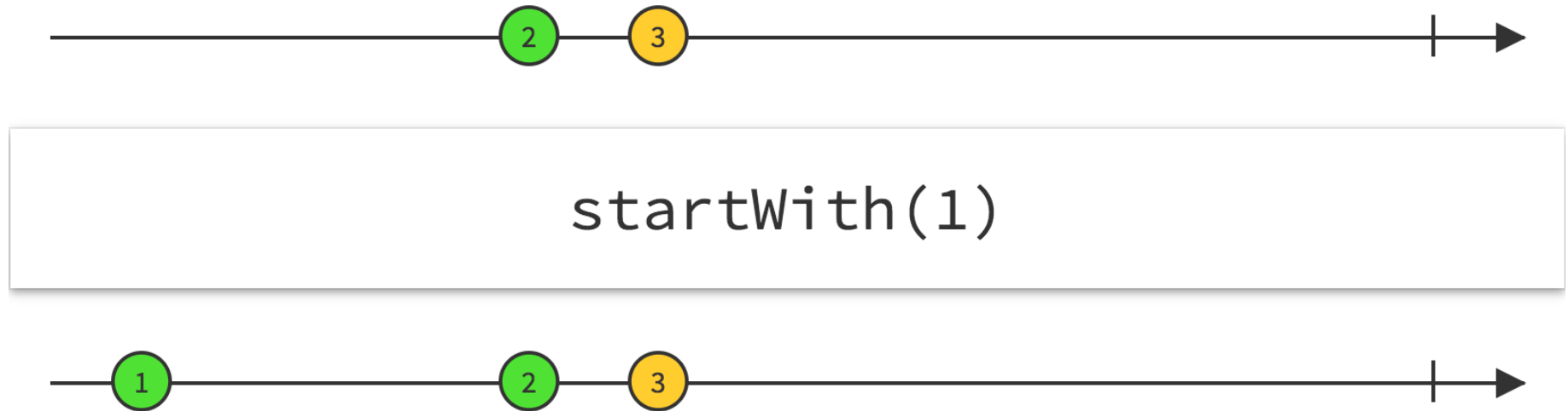


Podobné jako map, ale místo Observable/Promise vrátí rovnou hodnoty, nemusíte tedy zanořovat subscribe/then v map.

reactivex.io/documentation/operators/flatmap.html

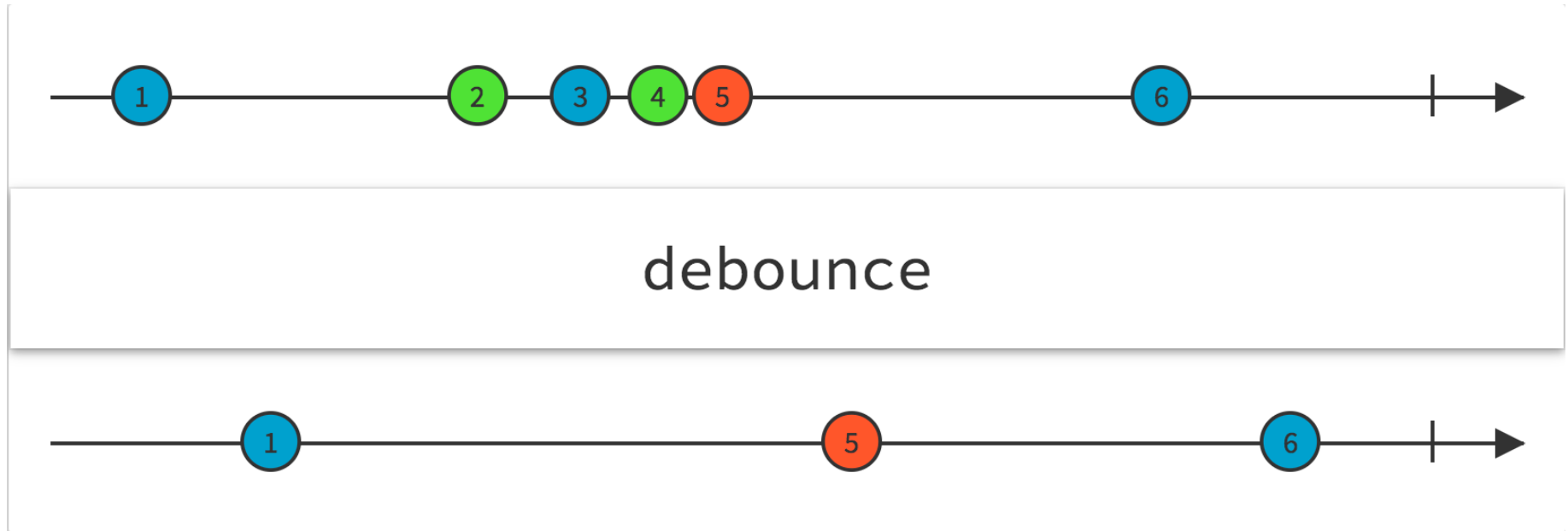
RxJS .startWith

3-6



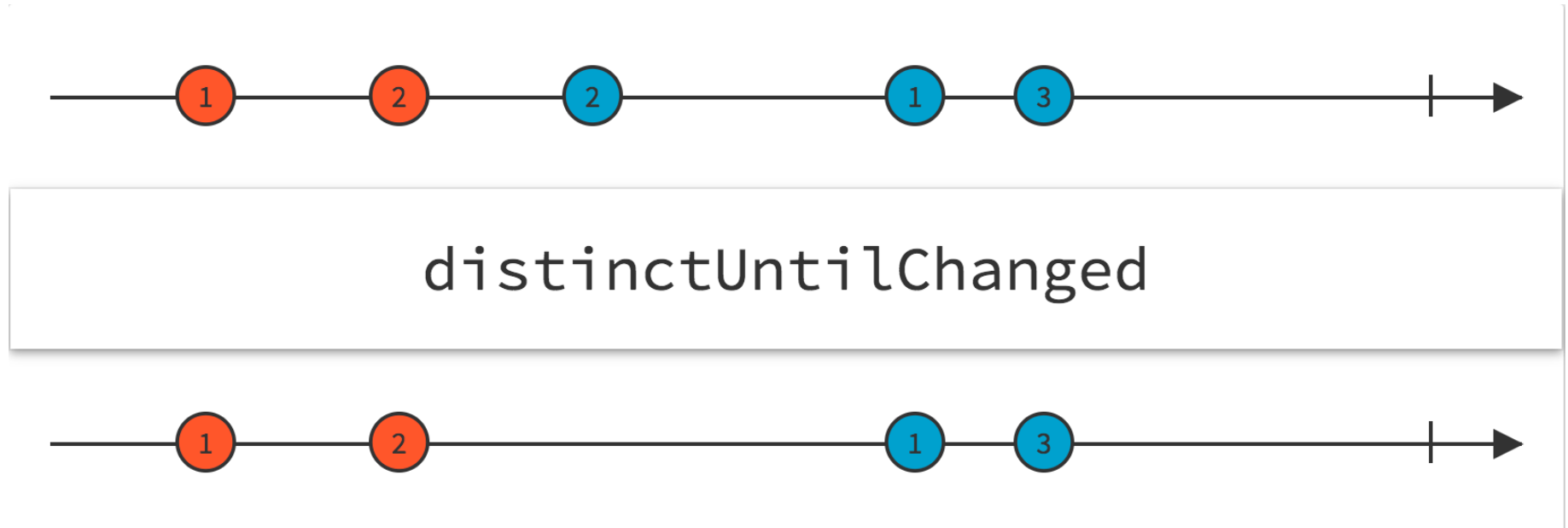
RxJS .debounce

3-7



RxJS .distinctUntilChanged

3-8



rxmarbles.com

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

Vyzkoušejte si to

3-9

angular.cz/gdg-brno-a2

4-0

