Enhance Web Development

Angular 2

Le routeur

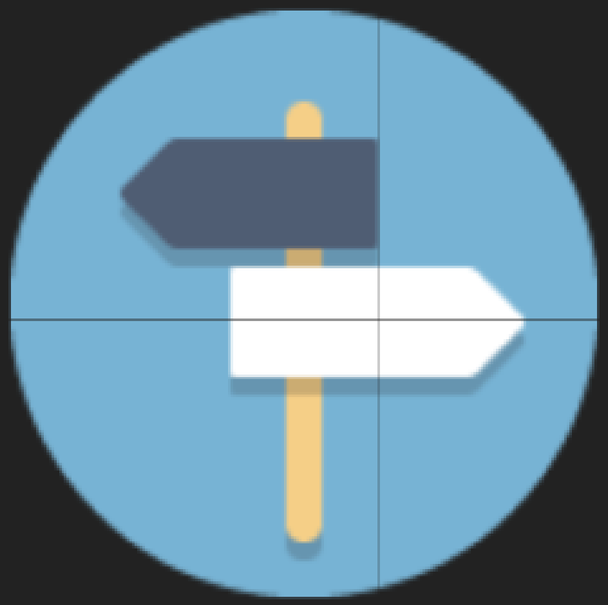
v3.0.0-alpha.7

- Ressources
- Qu'est-ce qu'un routeur ?
- Ok et angular ?

Ressources

- angular.io - Routing & Navigation
  https://angular.io/docs/ts/latest/guide/router.html
- Plunker - Official demo
  https://angular.io/resources/live-examples/router/ts/plnkr.html

Qu'est-ce qu'un routeur ?



SPA & routeurs

- SPA = navigation sans rafraichissement
- Les URLs doivent être lisible par un humain et porter

  l'état de l'application demandé
- ex: pour accéder à la page météo de Montpellier

  http://site.com/meteo/montpellier

  ou

  http://site.com/meteo/43.6100788,3.8391422,13z

Une URL

Décrit l'état courant de l'application

Protocol

Path

Hash / Fragment

https://www.monsite.com/weather/34000?type=awesome#traffic=disabled

Domain

Query string

Paramètres additionnels

Server

Client

Configuration requise

- HTML 5 History API

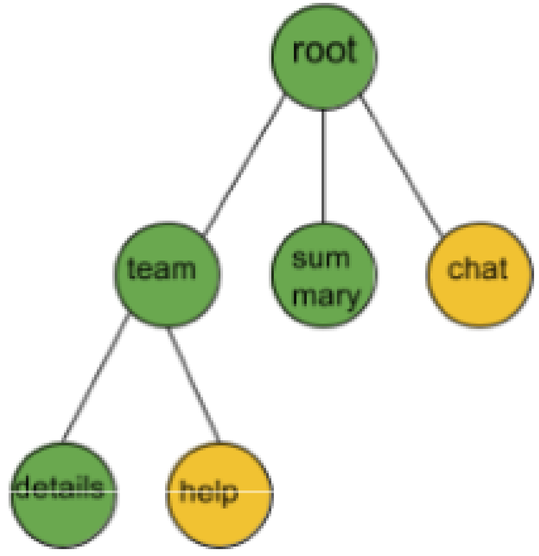
  https://css-tricks.com/using-the-html5-history-api/

  http://html5demos.com/history

- Configuration serveur

  Rediriger le traffic vers la page d'accueil

Example de routing

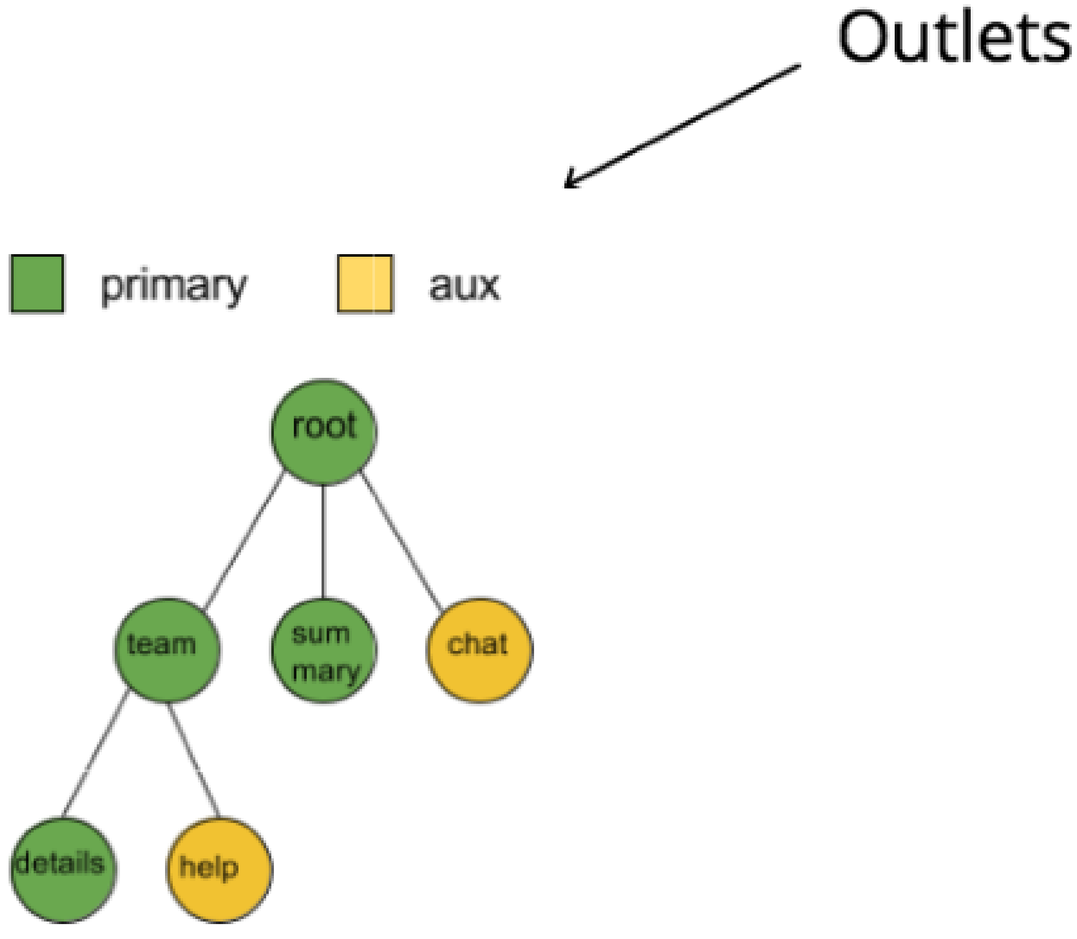


Ok et angular ?

Configuration

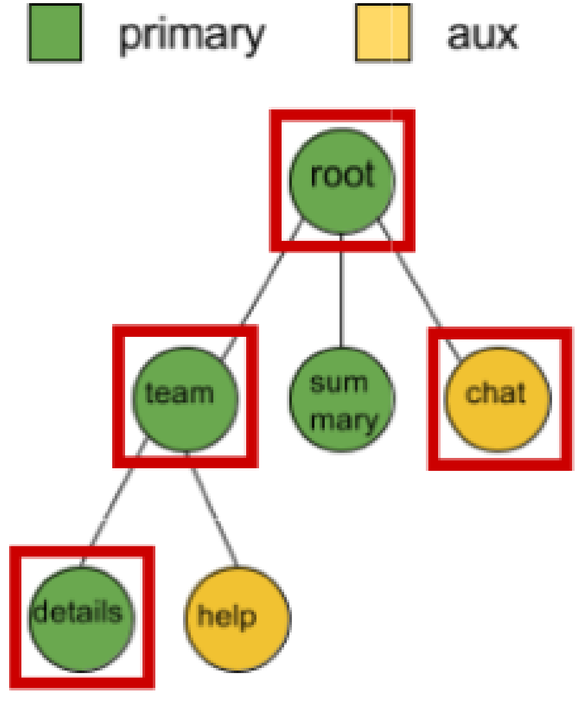
```
[
  {path: 'team/:id', component: TeamCmp, children: [
    {path: 'details', component: DetailsCmp},
    {path: 'help', component: HelpCmp, outlet: 'aux'}
  ]},
  {path: 'summary', component: SummaryCmp},
  {path: 'chat', component: ChatCmp, outlet: 'aux'}
]
```

Configuration

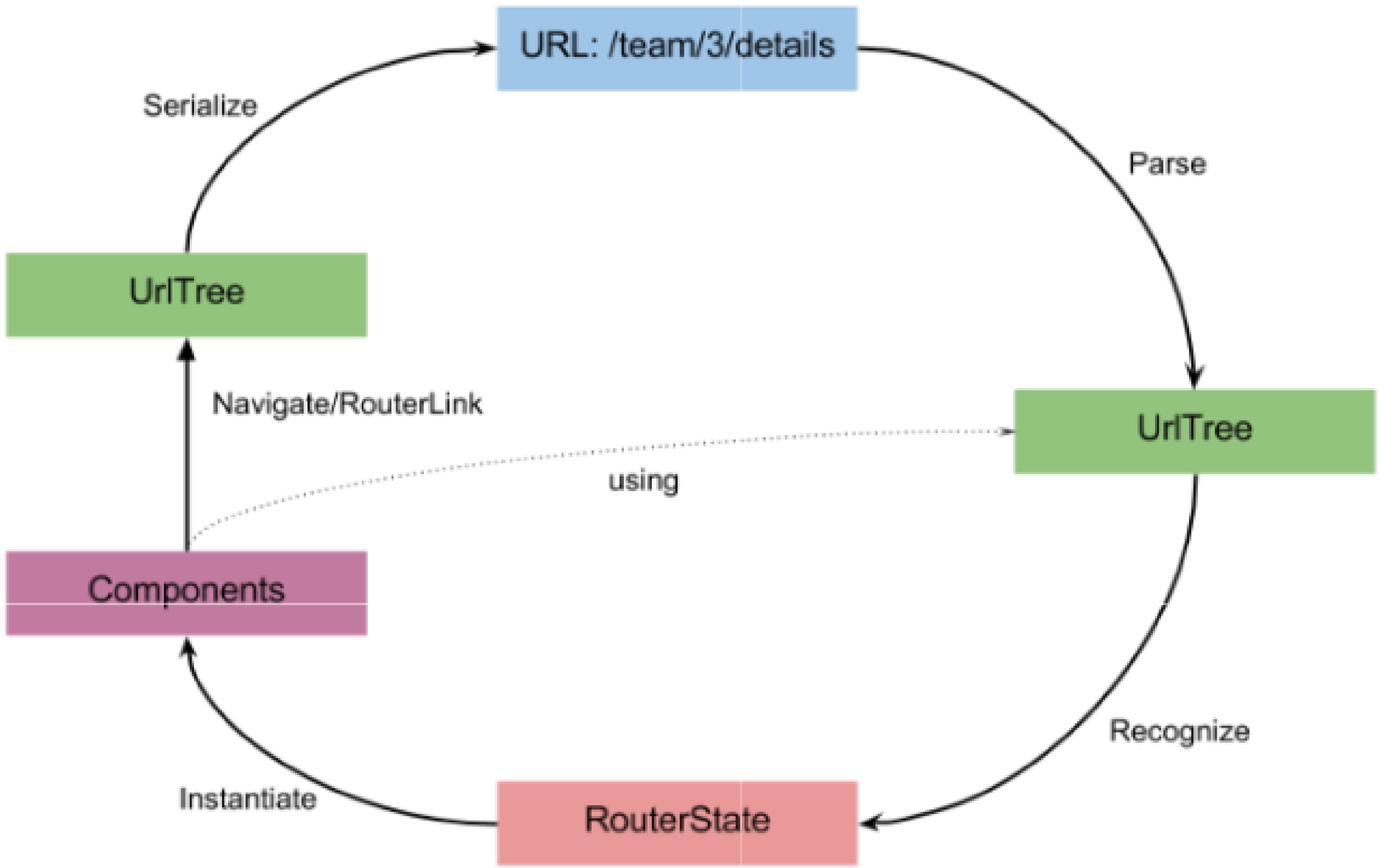


Etat activé

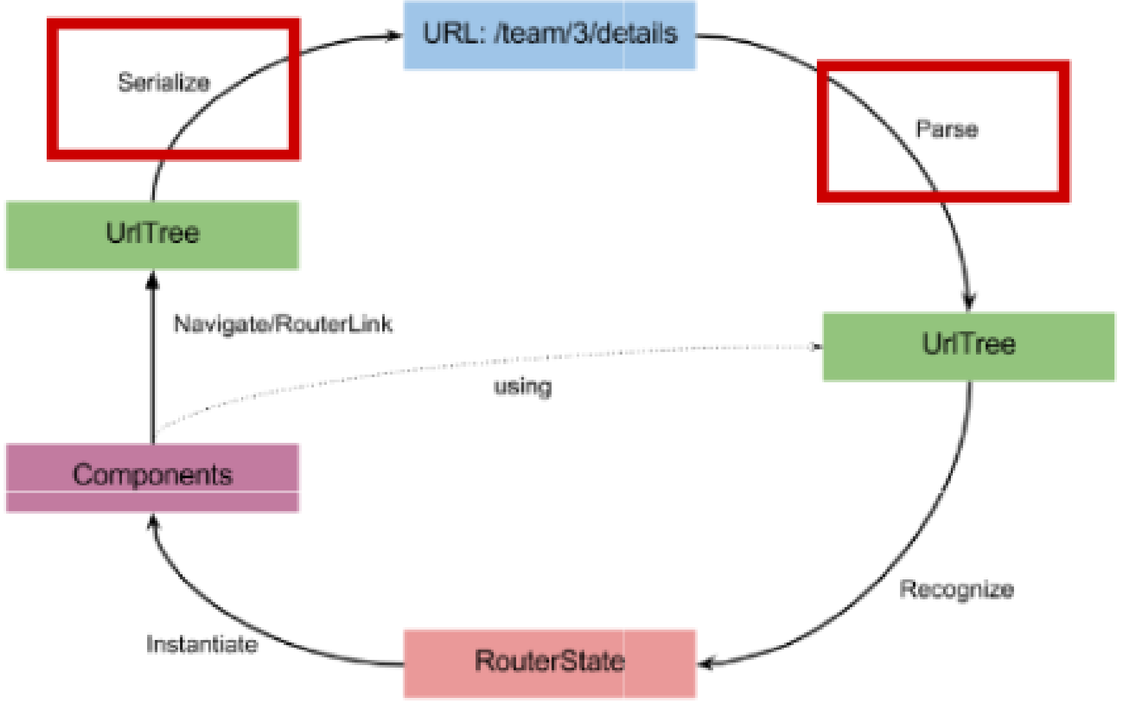
Summary component



Le processus Angular



Parsing & Serialization



Parsing & serialization

- Atout des web application vs native application pour le partage d'état d'une application
- Transforme une URL en un abre URL et vice-versa
- Cette étape ne dépend pas de votre application

```
// URL
/team/3/details

// URL Tree
new UrlSegment(paths: [], children: {
  primary:
    new UrlSegment(paths: [
      new UrlPathWithParams(path: 'team', parameters: {}),
      new UrlPathWithParams(path: '3', parameters: {}),
      new UrlPathWithParams(path: 'details', parameters: {})
    ], children: {}
  )
```

Parsing

```
// URL
/team/3(aux:/chat;open=true)

// URL Tree
new UrlSegment(paths: [], children: {
  primary:
    new UrlSegment(paths: [
      new UrlPathWithParams(path: 'team', parameters: {})
      new UrlPathWithParams(path: '3', parameters: {})
    ], children: {}
  ),
  aux:
    new UrlSegment(paths: [
      new UrlPathWithParams(path: 'chat', parameters: {open: 'true'})
    ], children: {}
  )
}
```

Parsing

Conclusion

- () => enfant multiple
- : => pour spécifier l'outlet
- ; => Paramètre spécifique à une route

Exemples :

/team/3(aux:/chat;open=true)

/team/3(aux:/help;lang=fr)

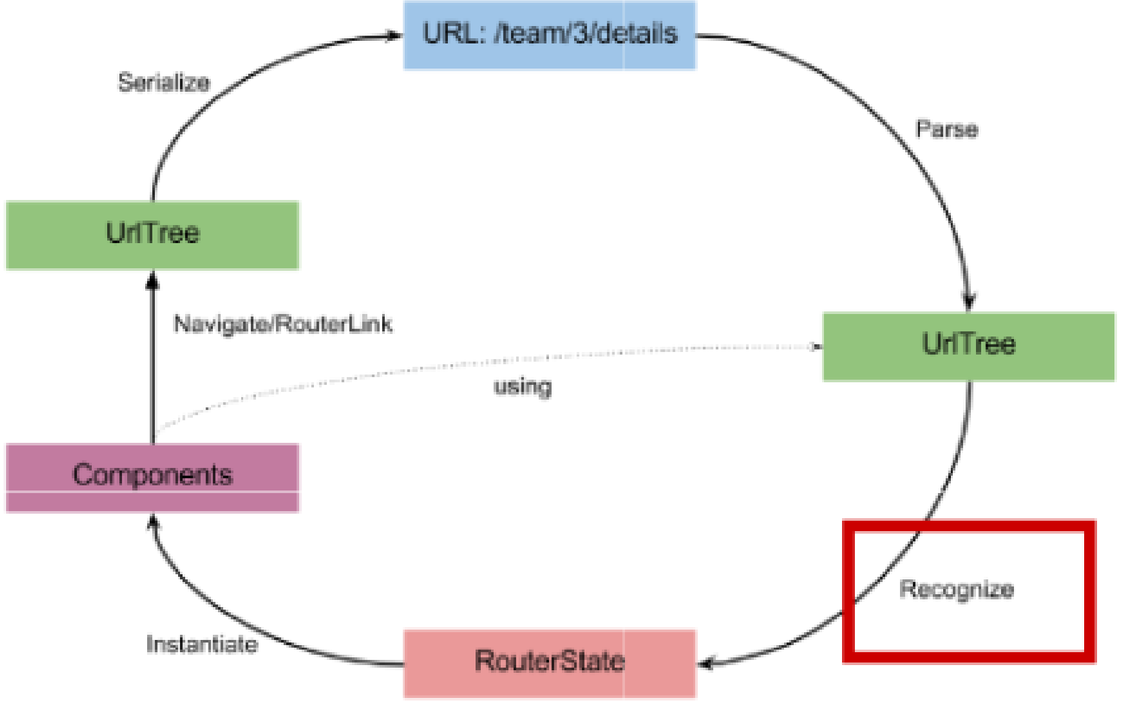
/team/3/detail

Aller plus loin

- Injection d'une stratégie de serialization personnalisée

```
bootstrap(MyComponent, [{
  provider: RouterURLSerializer,
  useClass: MyCustomSerializer
}]);
```

Route state recognition



Route state recognition

- Procède à créer une table de correspondance entre l'arbre d'URL issue du parsing et les composants de l'application
- impossible = erreur fatale
- Le router va créer une liste d'*ActivatedRoute*
- *ActivatedRoute* = un unique composant

```
// URL
/team/3/details

// Activated routes
new ActivatedRoute(component: TeamCmp, url: [
  new UrlPathWithParams(path: 'team', parameters: {}),
  new UrlPathWithParams(path: '3', parameters: {})
])
new ActivatedRoute(component: DetailsCmp, url: [
  new UrlPathWithParams(path: 'details', parameters: {})
])

// RouterState
// ActivatedRoute(component: RootCmp)
//   -> ActivatedRoute(component: TeamCmp)
//     -> ActivatedRoute(component: DetailsCmp)
```

Recognition

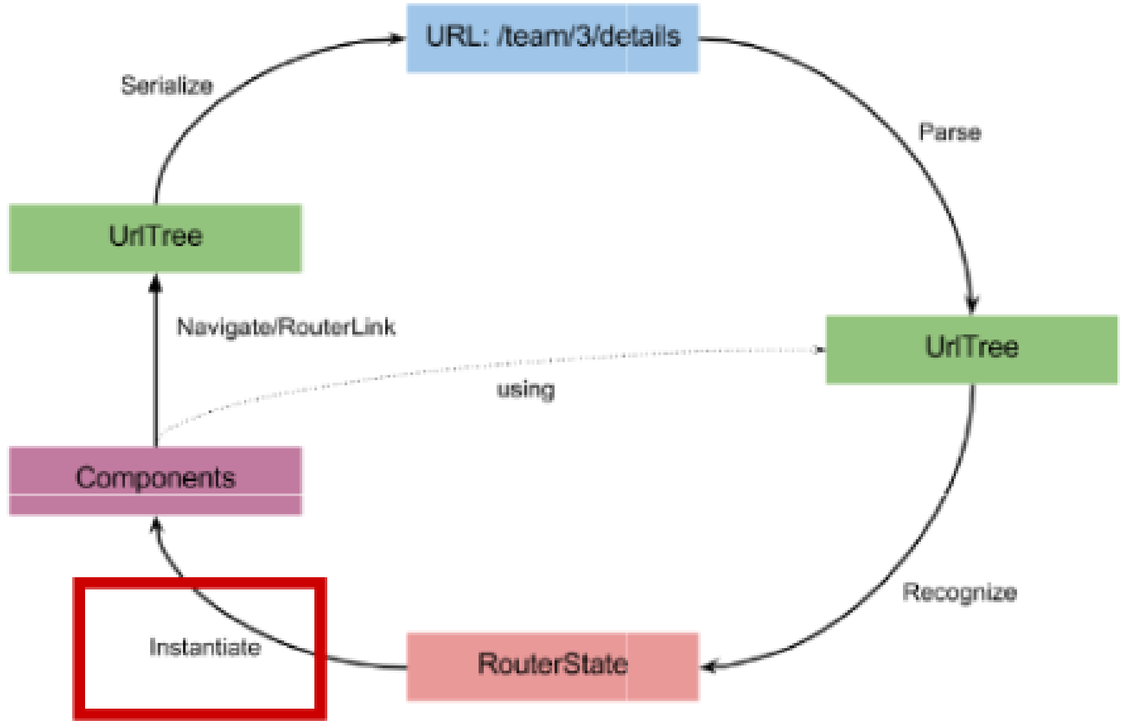
```
// URL
/team/3(aux:/chat;open=true)

// Activated routes
new ActivatedRoute(component: TeamCmp, outlet: primary, url: [
  new UrlPathWithParams(path: 'team', parameters: {}),
  new UrlPathWithParams(path: '3', parameters: {})
])
new ActivatedRoute(component: ChatCmp, outlet: 'aux', url: [
  new UrlPathWithParams(path: 'chat', parameters: {open: 'true'})
])

// RouterState
// ActivatedRoute(component: RootCmp, outlet: primary)
//   -> ActivatedRoute(component: TeamCmp, outlet: primary)
//   -> ActivatedRoute(component: ChatCmp, outlet: aux)
```

Recognition

Component Instantiation



Component instanciation

- Création ordonnée des instances de composant
- Possibilité de récupérer les paramètres de l'URL dans le composant
- Query parameters & Fragment ne sont pas spécifique à une route mais au routeur de façon globale
- Le routeur utilise des observables

```
// Components:
@Component({
  selector: chat,
  template: `
    Chat
  `
})
class ChatComponent {}

@Component({
  selector: 'team',
  template: `
    Team
    primary: { <router-outlet></router-outlet> }
  `
})
class TeamComponent {}

@Component({
  selector: 'root',
  template: `
    Root
    primary: { <router-outlet></router-outlet> }
    aux:   { <router-outlet name='aux'></router-outlet> }
  `
})
class RootComponent {}
```

Instanciation

```
// Activated Route:
ActivatedRoute(component: RootComponent)
  -> ActivatedRoute(component: TeamComponent, parameters: {id: 3}, outlet: primar
    -> ActivatedRoute(component: DetailsComponent, parameters: {}, outlet: primar
  -> ActivatedRoute(component: ChatComponent, parameters: {}, outlet: aux)
```

Instanciation

```
@Component({
  selector: 'team',
  template: `
    Team Id: {{id | async}}
    primary: { <router-outlet></router-outlet> }
  `
})
class TeamComponent {
  id: Observable<string>;
  constructor(r: ActivatedRoute) {
    //r.params is an observable
    this.id = r.params.map(r => r.id);
  }
}
```

Utiliser les paramètres

```
@Component({
  selector: 'details',
  template: `
    Details for Team Id: {{teamId | async}}
  `
})
class DetailsComponent {
  teamId:Observable<string>;
  constructor(r: ActivatedRoute, router: Router) {
    const teamActivatedRoute = router.routerState.parent(r);
    this.teamId = teamActivatedRoute.params.map(r => r.id);
  }
}
```

Utiliser les paramètres

```
@Component({
  selector: 'team',
  template: ` `
})
class MyComponent {
  constructor(r: Router) {
    const q: Observable<{[k:string]:string}> = r.routerState.queryParams;
    const f: Observable<string> = r.routerState.fragment;
  }
}
```

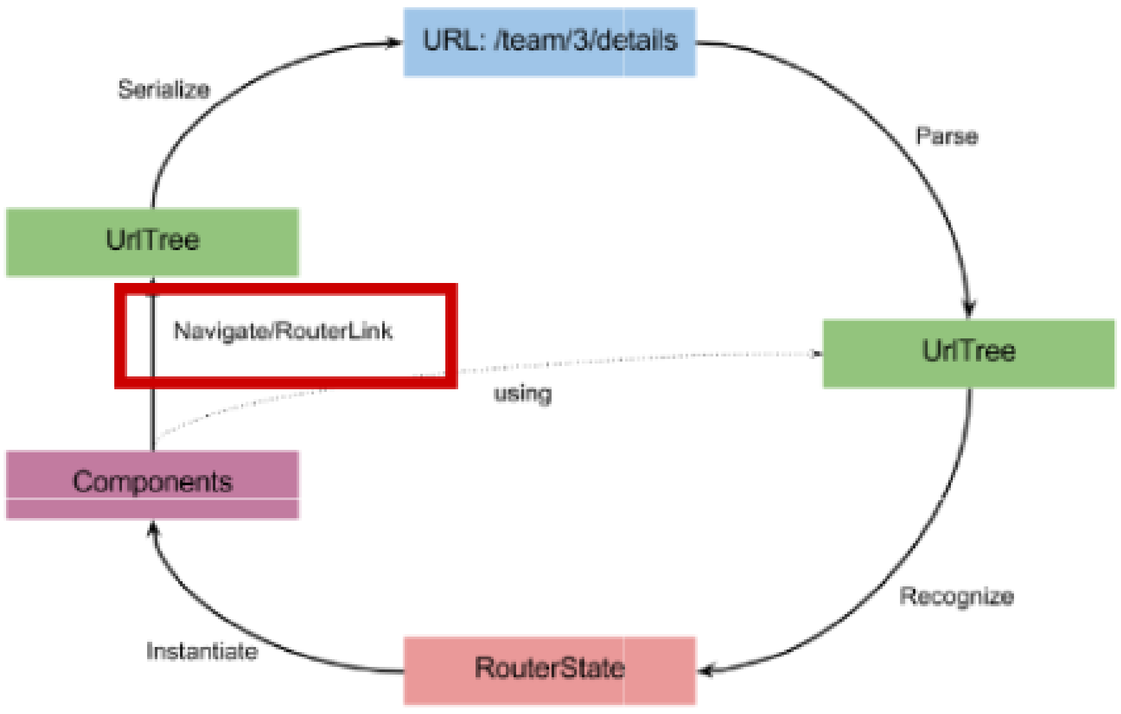
Query params & Fragment

```
@Component({
  selector: 'team',
  template: `
    Team Id: {{id}}
  `
})
class TeamComponent {
  id:string;
  constructor(r: ActivatedRoute, router: Router) {
    const s: ActivatedRouteSnapshot = r.snapshot;
    // matrix params of a particular route
    this.id = s.params.id;

    const ss: RouterStateSnapshot = router.routerState.snapshot;
    // query params are shared
    const q: {[k:string]:string} = ss.queryParams;
  }
```

Capture à un instant t (snapshot)

Navigation


URL: /team/3/details
Serialize
Parse
UrlTree
Navigate/RouterLink
using
UrlTree
Components
Recognize
Instantiate
RouterState

Navigation

- Deux façons :
  - router.navigate
  - RouterLink (directive)

```
class TeamComponent {
  teamId: number;
  userName: string;
  constructor(private router: Router) {}

  onClick(e) {
    this.router.navigate(['/team', this.teamId, 'user', this.userName]).then(_ => {
       //navigation is done
    }); //e.g. /team/3/user/victor
  }
}
```

```
class TeamComponent {
  private teamId;
  private userName;
  constructor(private router: Router, private r: ActivatedRoute) {}

  onClick(e) {
    this.router.navigate(['../', this.teamId, 'user', this.userName], {relativeTo: this.r});
  }
}
```

router.navigate

```
@Component({
  selector: 'team',
  directives: ROUTER_DIRECTIVES,
  template: `
     <a [routerLink]="['../', this.teamId, 'user', this.userName]">Navigate</a>
  `
})
class TeamComponent {
  private teamId;
  private userName;
```

RouterLink

```
// absolute navigation
this.router.navigate(['/team', this.teamId, 'details']);

// you can collapse static parts into a single element
this.router.navigate(['/team/3/details']);

// also set query params and fragment
this.router.navigate(['/team/3/details'], {queryParams: newParams, fragment: 'fra
// e.g., /team/3;extra=true/details
this.router.navigate(['/team/3', {extra: true}, 'details']);

// relative navigation to /team/3/details
this.router.navigate(['./details'], {relativeTo: this.route});

// relative navigation to /team/3
this.router.navigate(['../', this.teamId], {relativeTo: this.route});
```

Plus de syntaxe !

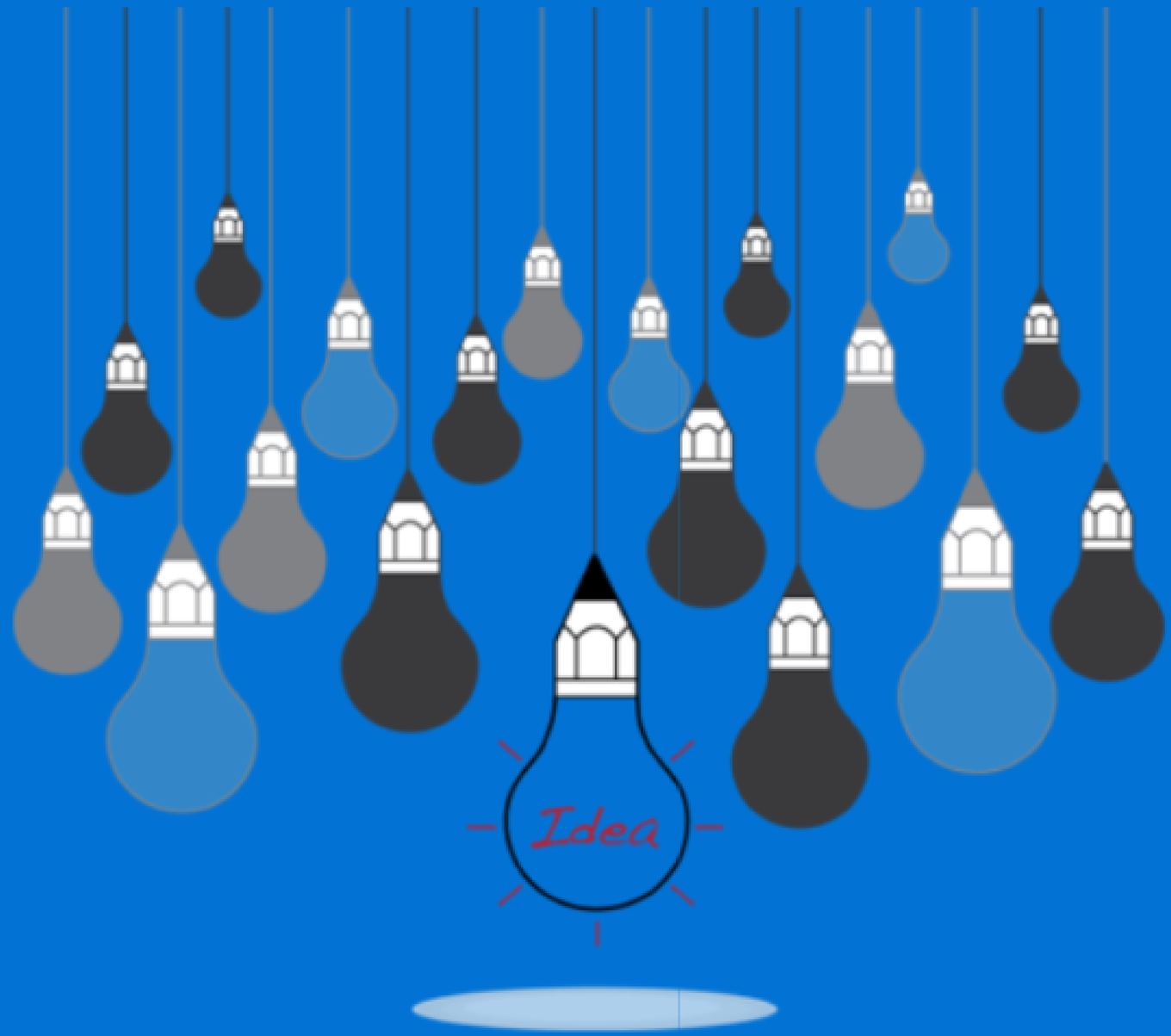
URL ou state ?

- La navigation se base sur les URLs et non sur les états

  !!!

- Raison : le lazy-loading qui pose cette contrainte

Demo time !



Défi

- Créer une route pour le composant **widget**
- Utiliser la directive **RouteLink** pour naviguer au composant **widget**
- Créer une méthode qui navigue directement à cette route
- Ajouter des paramètres de route et des query params/fragment à la route **widget**
- BONUS : Créer un composant **widget-item** avec la définition de routes enfant