

# Angular2 Router

---

# Angular에서 주소 관리 기법

---

## 1. HashLocationStrategy(해시 기반 내비게이션 정책)

URL에 해시 기호(#)를 사용해서 해시 기호 이후의 부분은 웹 페이지의 특정 부분을 가리키도록 라우팅하는 정책이다. 이 방식은 오래된 브라우저에서도 정상적으로 동작한다.

## 2. PathLocationStrategy(방문 기록 API 기반 내비게이션 정책)

브라우저 방문 기록 API를 사용하는 정책이며 HTML5를 지원하는 브라우저에서만 동작한다. Angular 기본 정책

# Angular의 정책 설정 - Hash

---

```
import { LocationStrategy, HashLocationStrategy }    from '@angular/common'

@NgModule({
  imports:      [ BrowserModule, FormsModule ],
  declarations: [ ],
  bootstrap:    [ ],
  providers:    [
                    {provide: LocationStrategy, useClass: HashLocationStrategy}
                ]
})
export class AppModule{}
```

# Angular의 정책 설정 - Path

---

```
import { APP_BASE_HREF } from '@angular/common';
@NgModule({
  imports:      [ BrowserModule, FormsModule ],
  declarations: [ ],
  bootstrap:    [ ],
  providers:    [
                  {provide: APP_BASE_HREF, useValue: './'}
                ]
})
export class AppModule {} // <base href="/"> 필수
```

# Route 구성 요소

---

## 1. Route

라우터를 구현하는 객체. 원하는 경로로 이동하려면 이 객체의 `navigate()` 함수와 `navigateByUrl()` 함수를 사용하거나, 라우터 객체의 `path` 프로퍼티 값을 변경하면 된다. 브라우저의 URL을 직접 수정해도 원하는 경로로 이동할 수 있다.

## 2. RouteOutlet

라우터가 컴퍼넌트를 랜더링하는 영역인 `<route-outlet>`을 구현하는 디렉티브이다.

# Route 구성 요소

---

## 3. Routes

특정 URL에 연결되는 컴퍼넌트를 지정하는 배열이다.

## 4. RouteLink

HTML 앵커 태그 <a>의 원래 용도는 브라우저의 URL 주소를 변경하는 것이지만, RouteLink 디렉티브를 사용하면 앵커 태그가 Angular 라우터를 통하도록 기능을 변경할 수 있다. RouteLink를 사용하면 라우터가 렌더링할 컴퍼넌트에 인자를 전달할 수도 있다.

## 5. ActivatedRoute

현재 동작하는 라우터 인스턴스를 나타내는 객체이다.

# RouterModule 연결 설정

---

```
import { RouterModule } from '@angular/router';

@NgModule({
  imports: [ BrowserModule, FormsModule,
            RouterModule.forRoot([
              {path: '/',      component: HomeComponent},
              {path: '/product', component: ProductComponent},
              {pate: '/content', component: ContentComponent}
            ])
  ],
  ...
```

# RouterModule 연결 설정

---

template: `

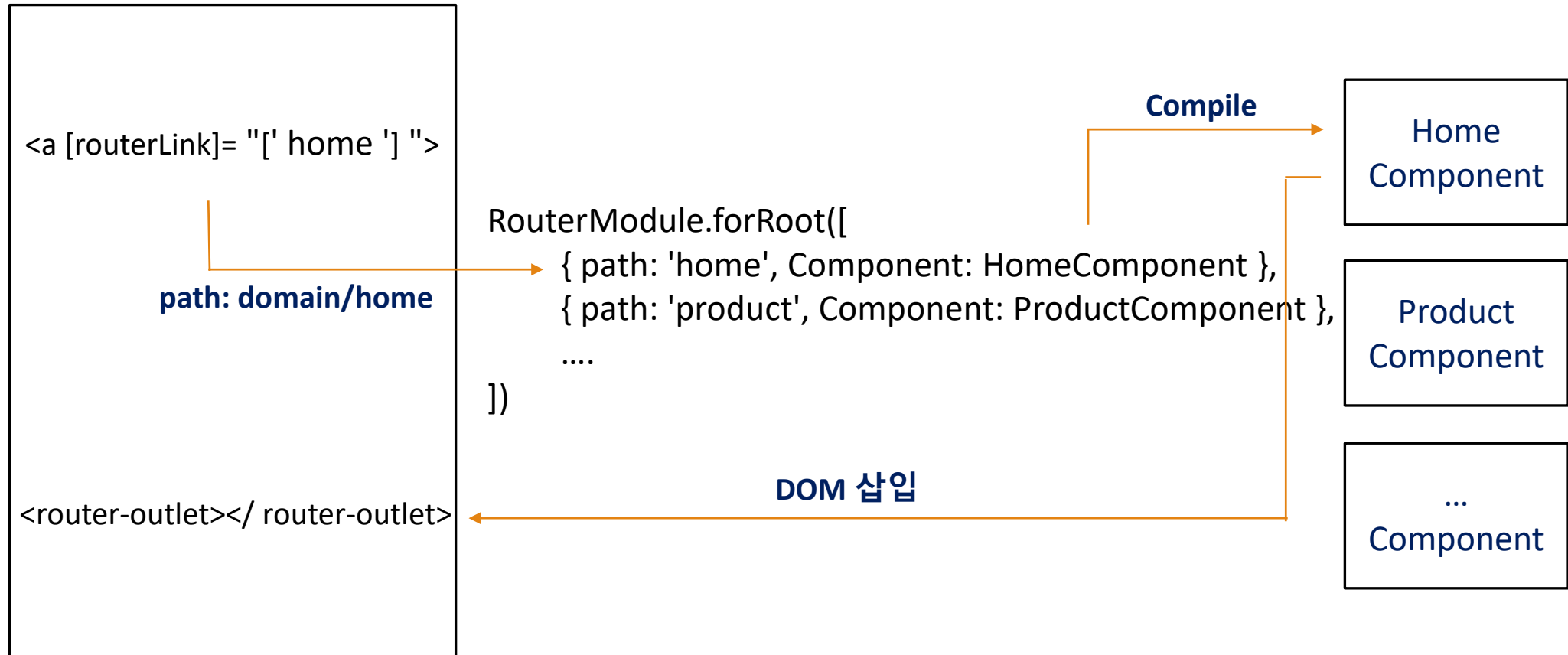
```
<div class="panel-body">  
  <a [routerLink]="[]">HOME</a> |  
  <a [routerLink]="['product']">PRODUCT</a> |  
  <a [routerLink]="['content']">CONTENT</a>  
</div>
```

```
<div>  
  <router-outlet></router-outlet>  
</div>
```

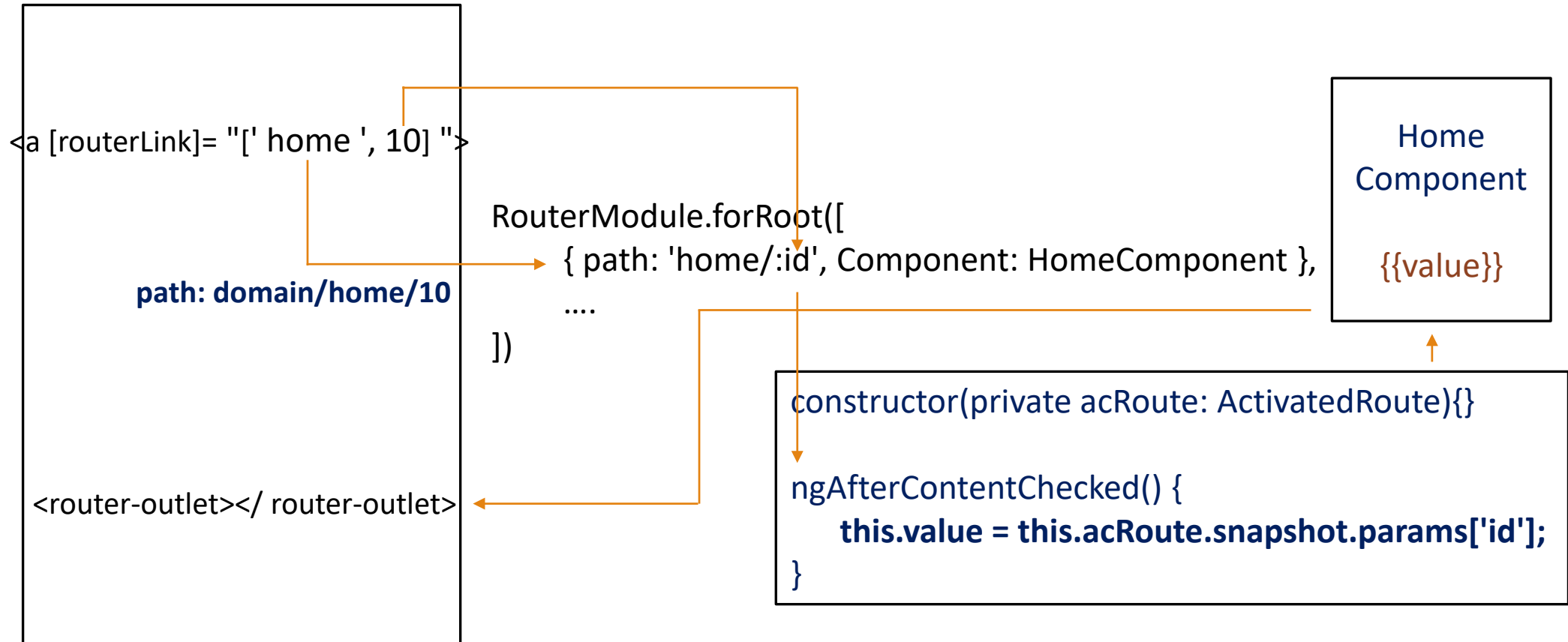
`  
,



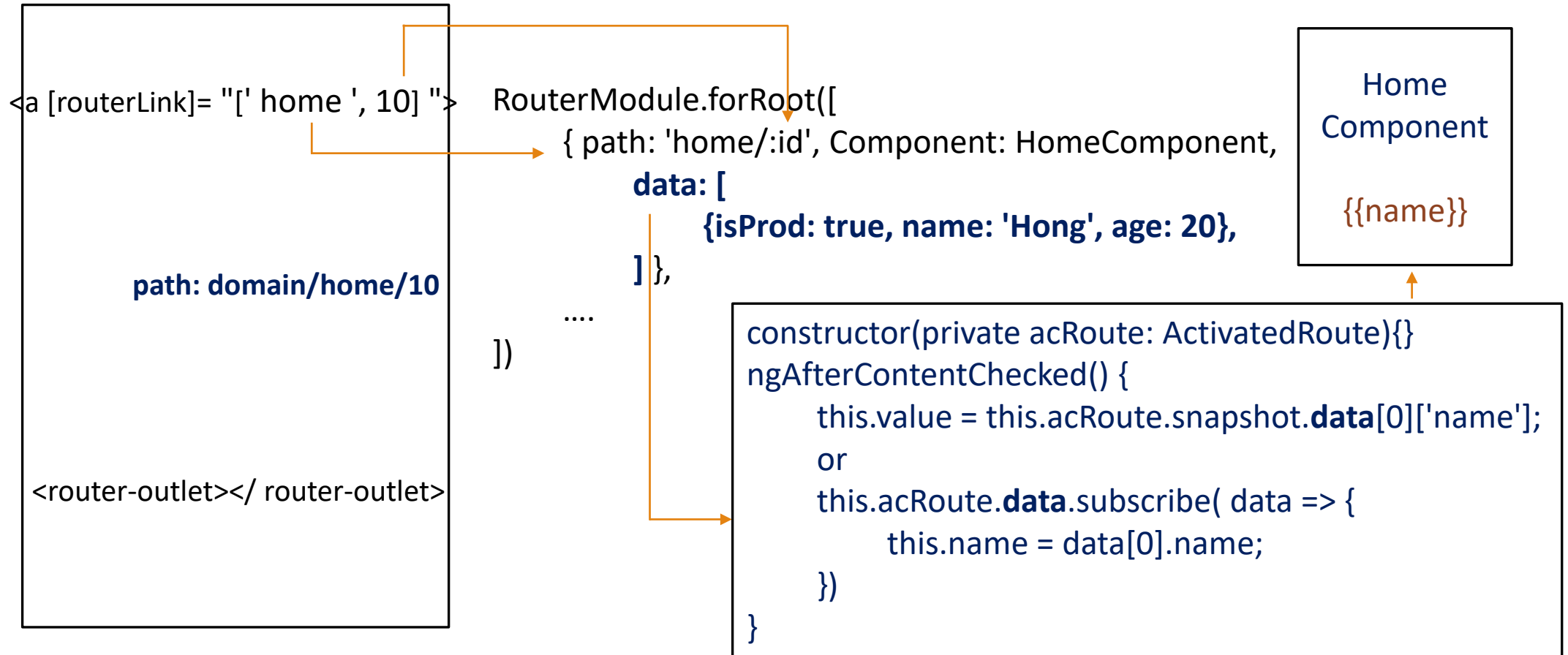
# RouterModule 연결 설정



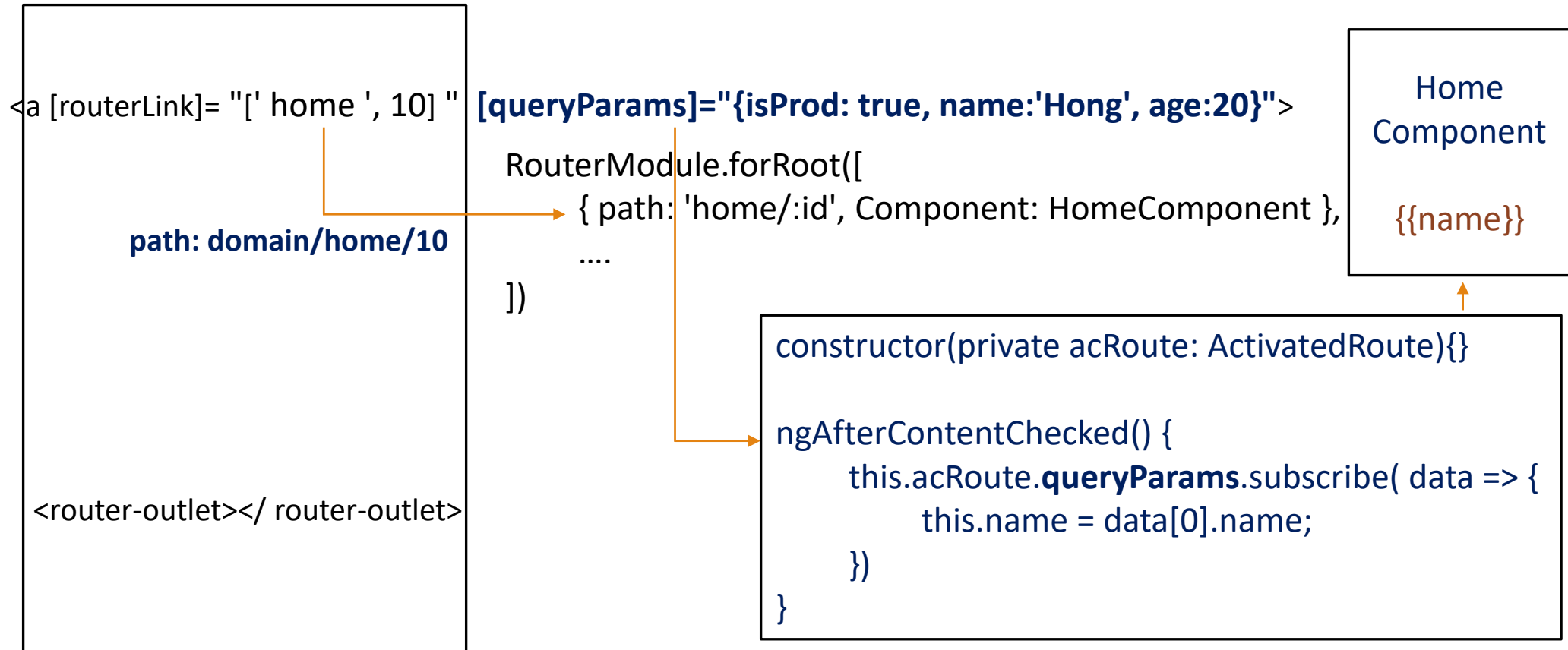
# RouterModule 연결 설정 - Params



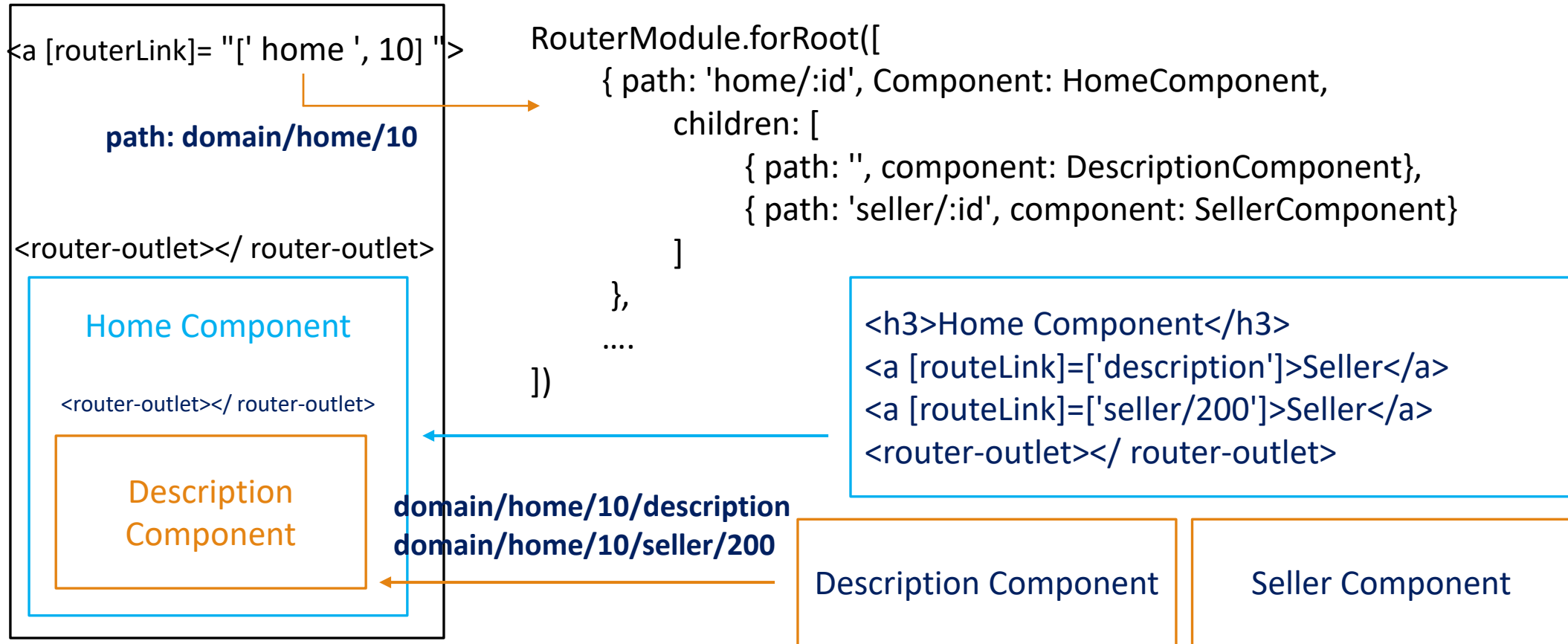
# RouterModule 연결 설정 – Data



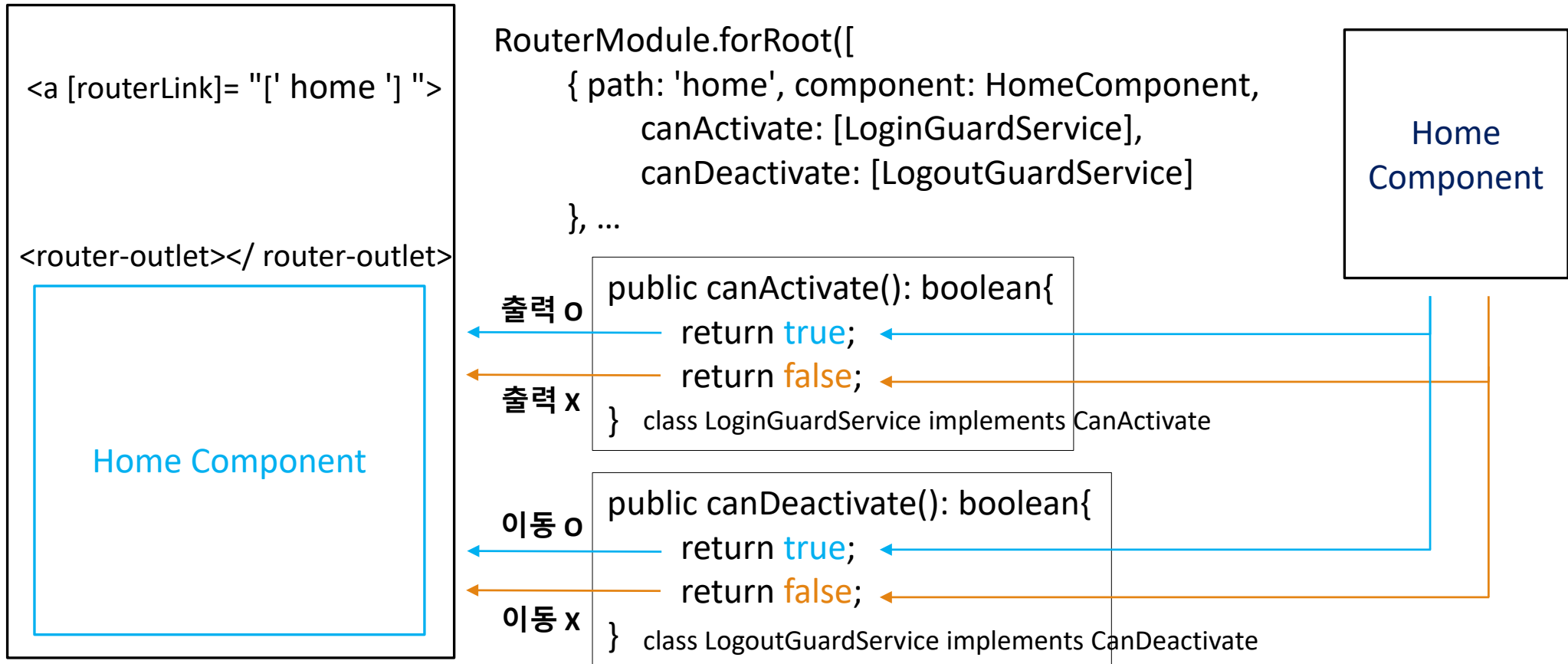
# RouterModule 연결 설정 – Argument



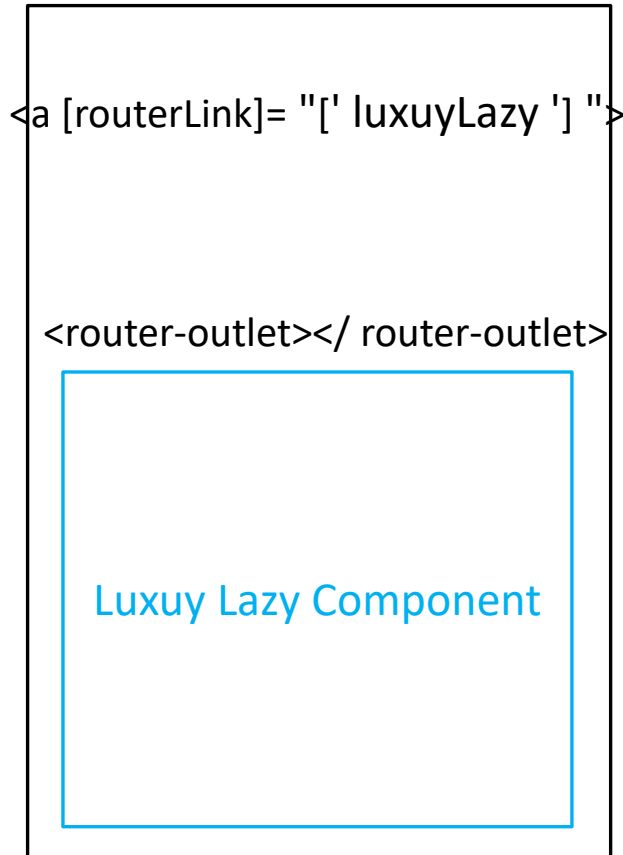
# RouterModule 연결 설정 – Children



# RouterModule 연결 설정 – CanActivate



# RouterModule 연결 설정 – Lazy Load



```
<a [routerLink]="['luxuyLazy/abc']">  
  Luxury Lazy  
</a>
```



Child Module명을 문자열 형태로 호출

```
RouterModule.forRoot([  
  { path: 'luxuyLazy',  
    loadChildren: 'app/components/luxury/luxuryLazy.module'},
```



Module 구성 요소를 Load

```
const route: Route = [  
  { path: 'abc', component: LuxuryLazyComponent}  
]  
export const luxuryLazyRouter = RouterModule.forChild(route);
```