



Angular state management



Store & Effect

Structure



Modules

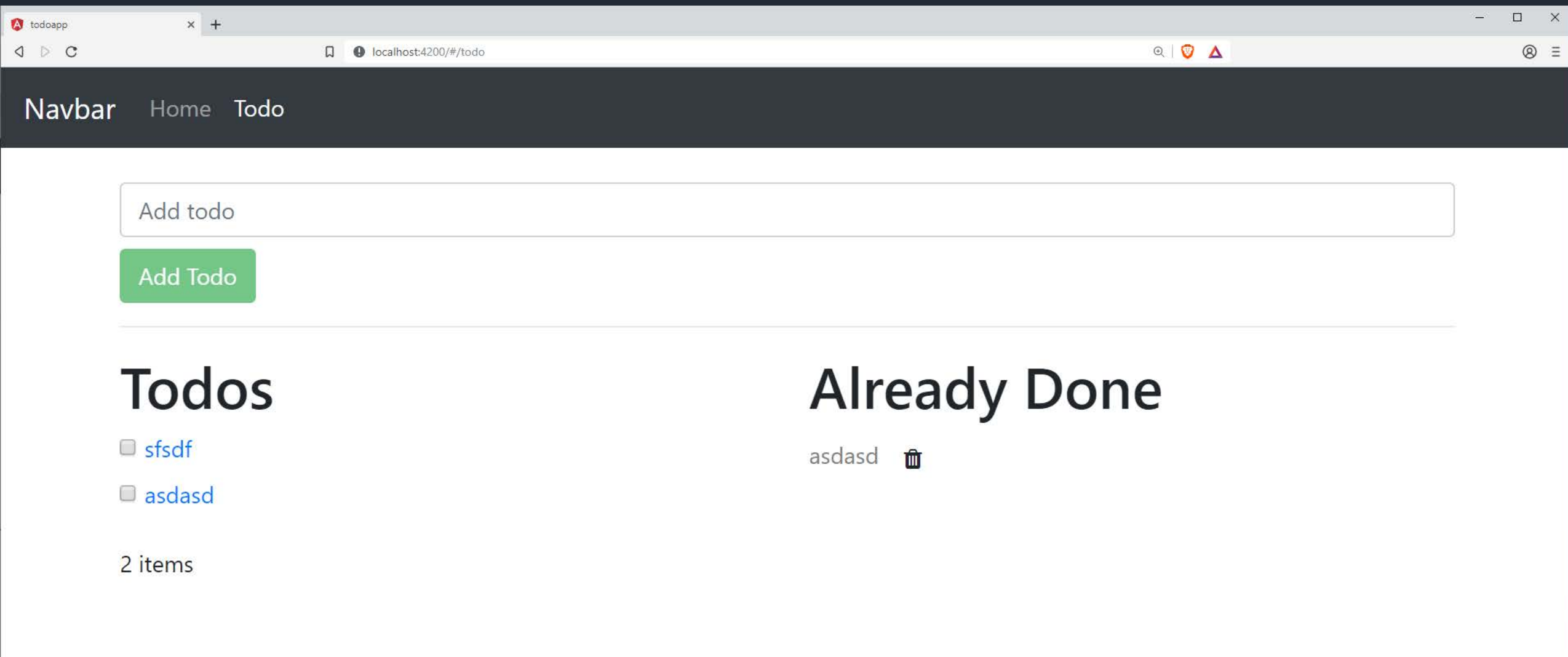


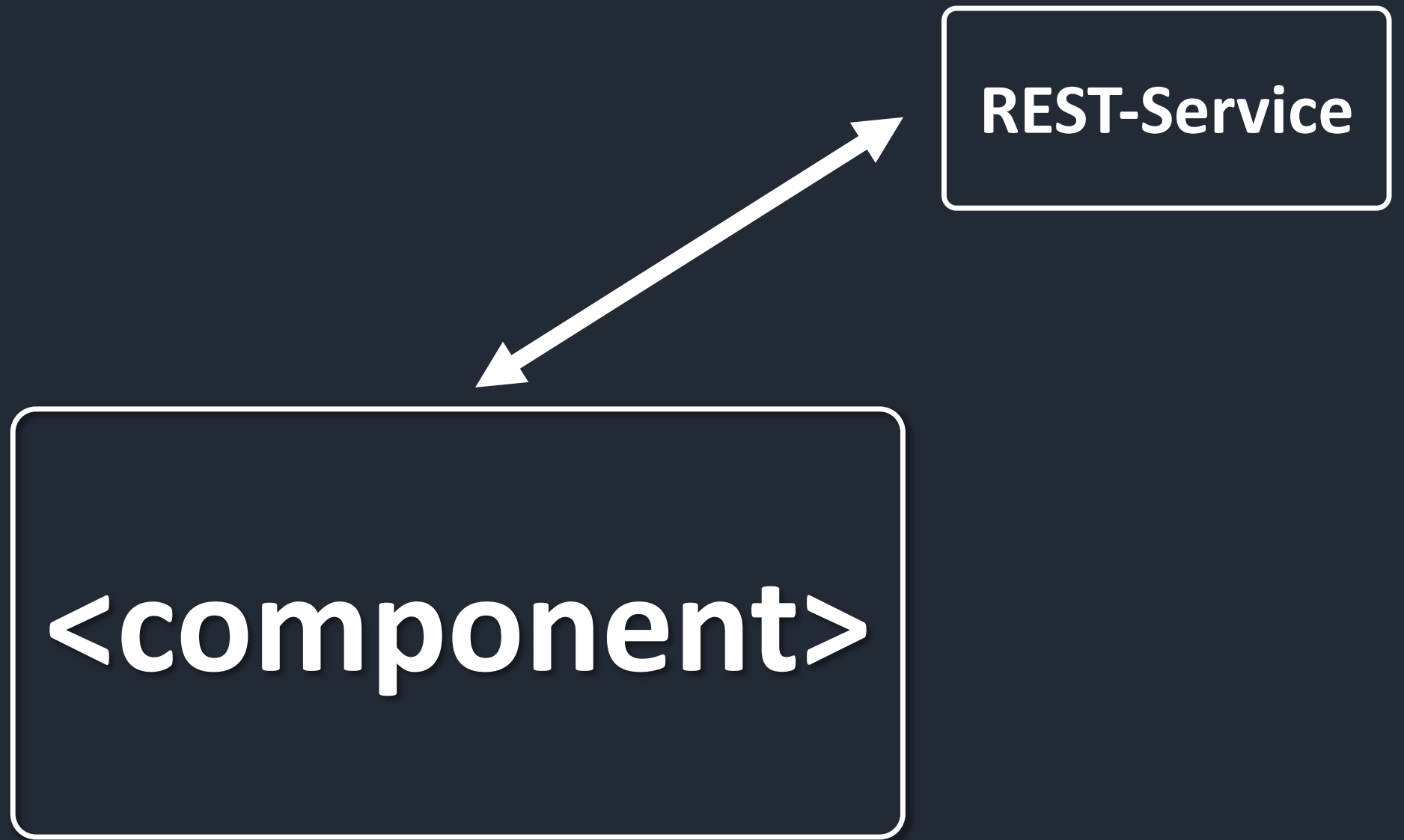


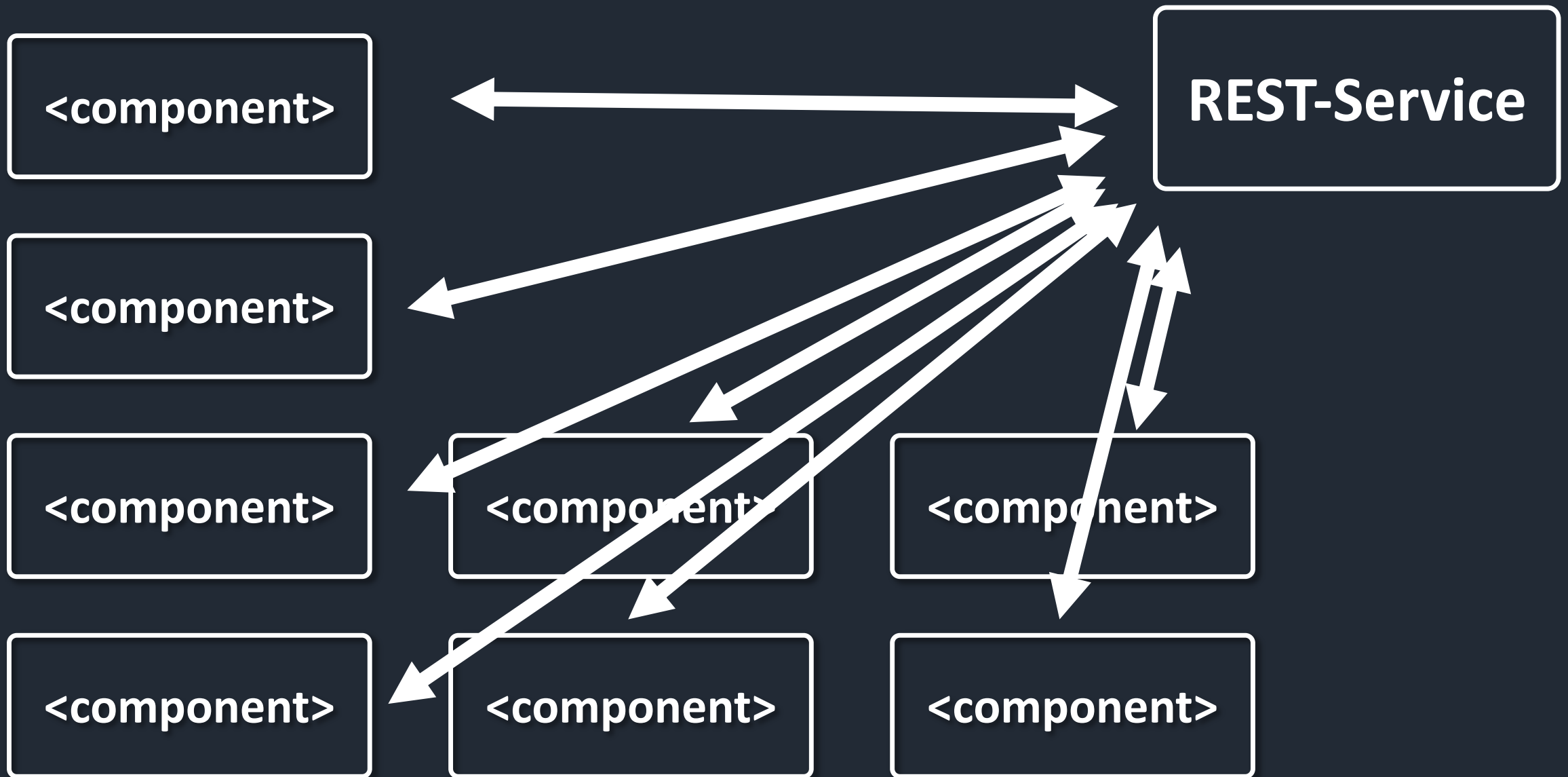
```
@NgModule({  
  imports: [  
    // Modules  
  ],  
  declarations: [  
    // Components & directives  
  ],  
  providers: [  
    // Services  
  ],  
  exports: [ /* ... */ ]  
})
```

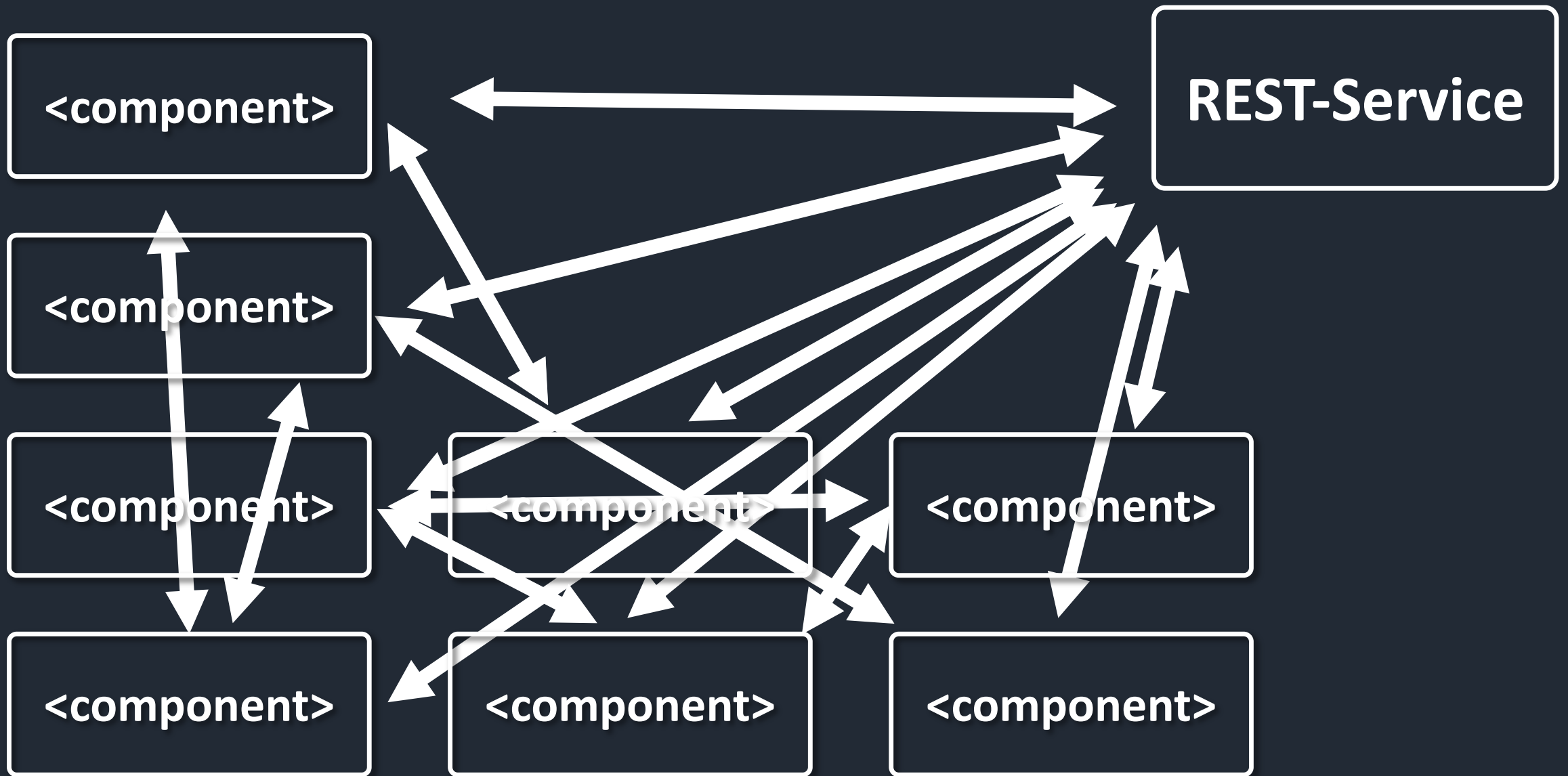
Componen

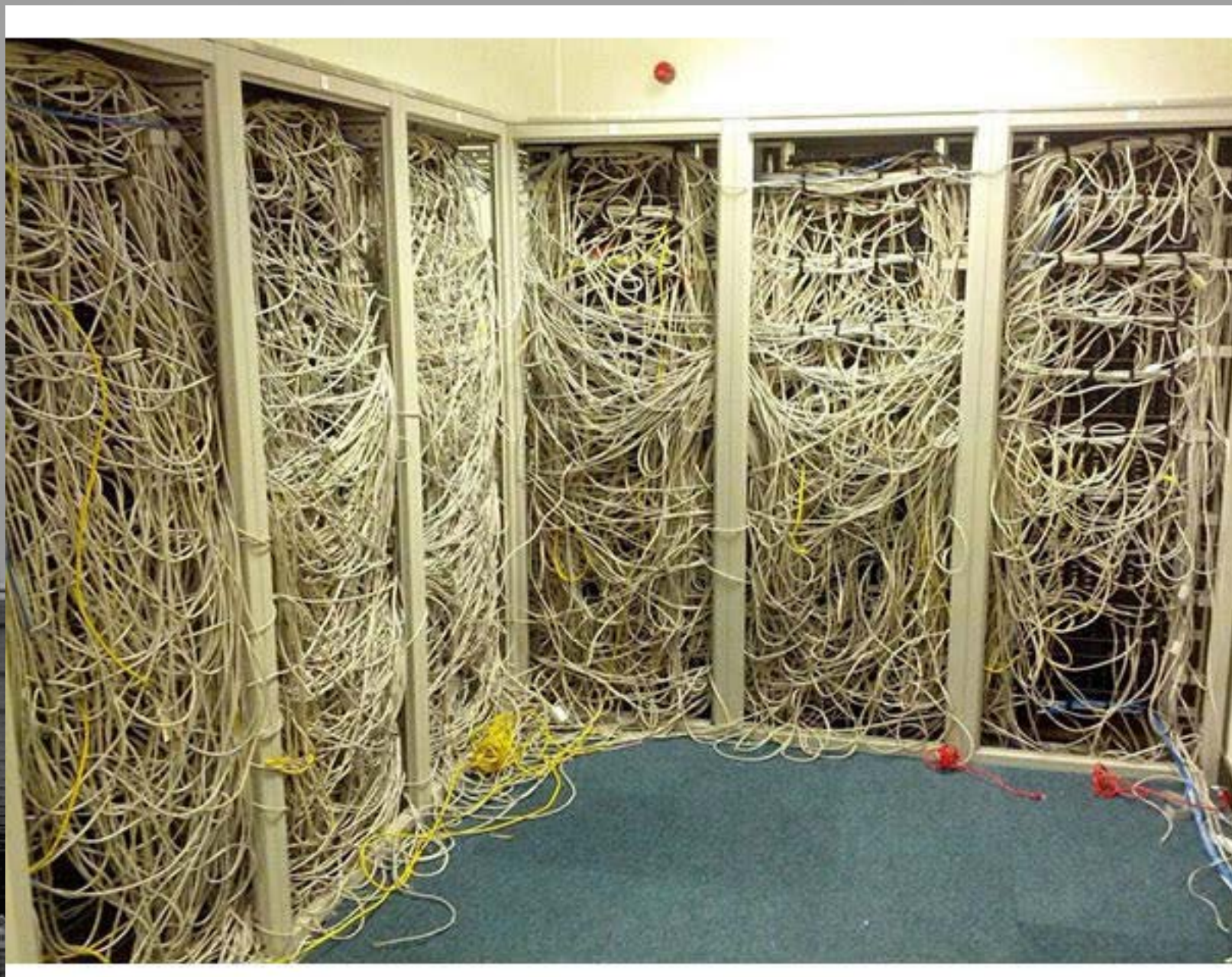


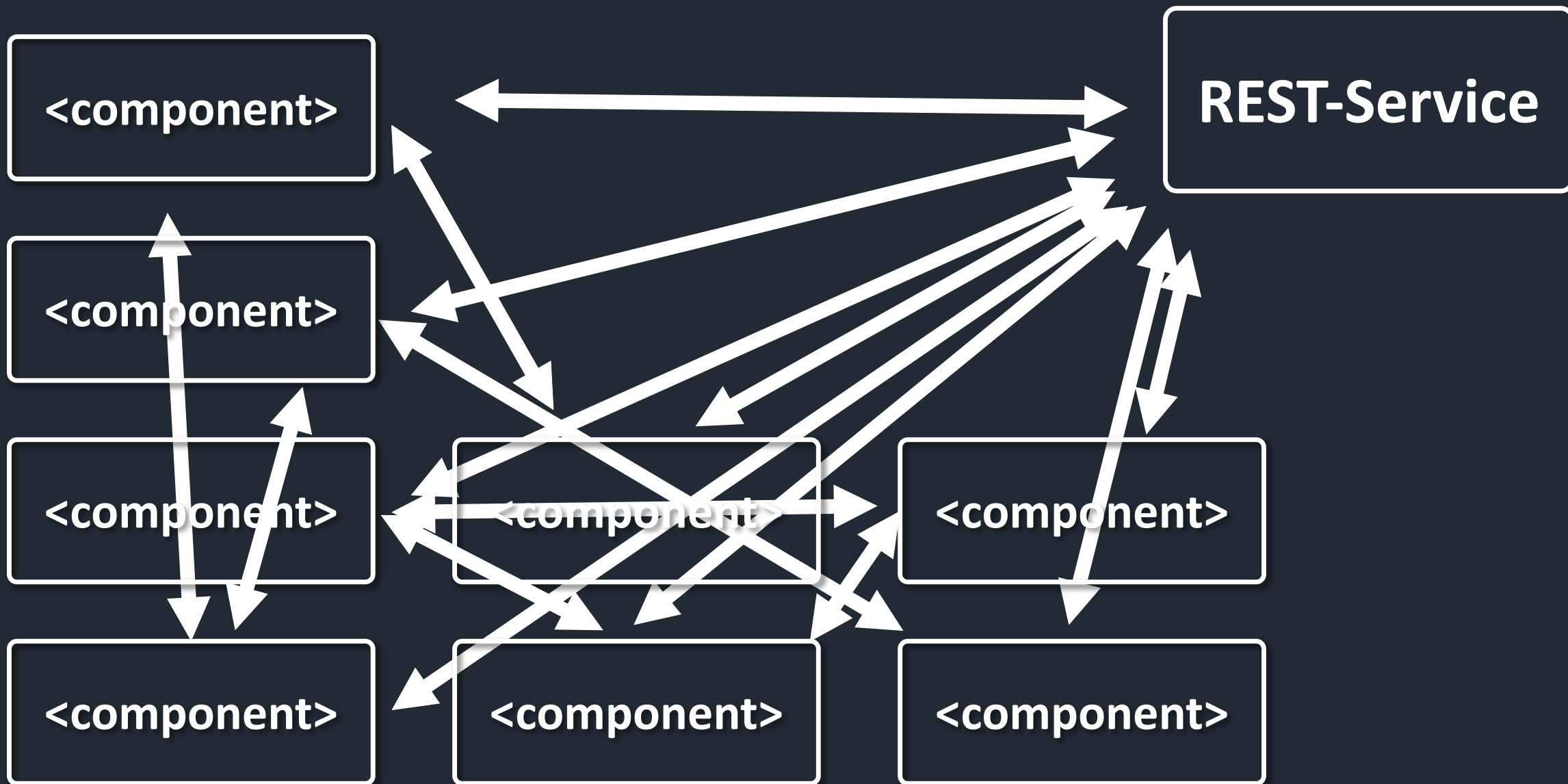














One **Way** Dataflow



Add Todo

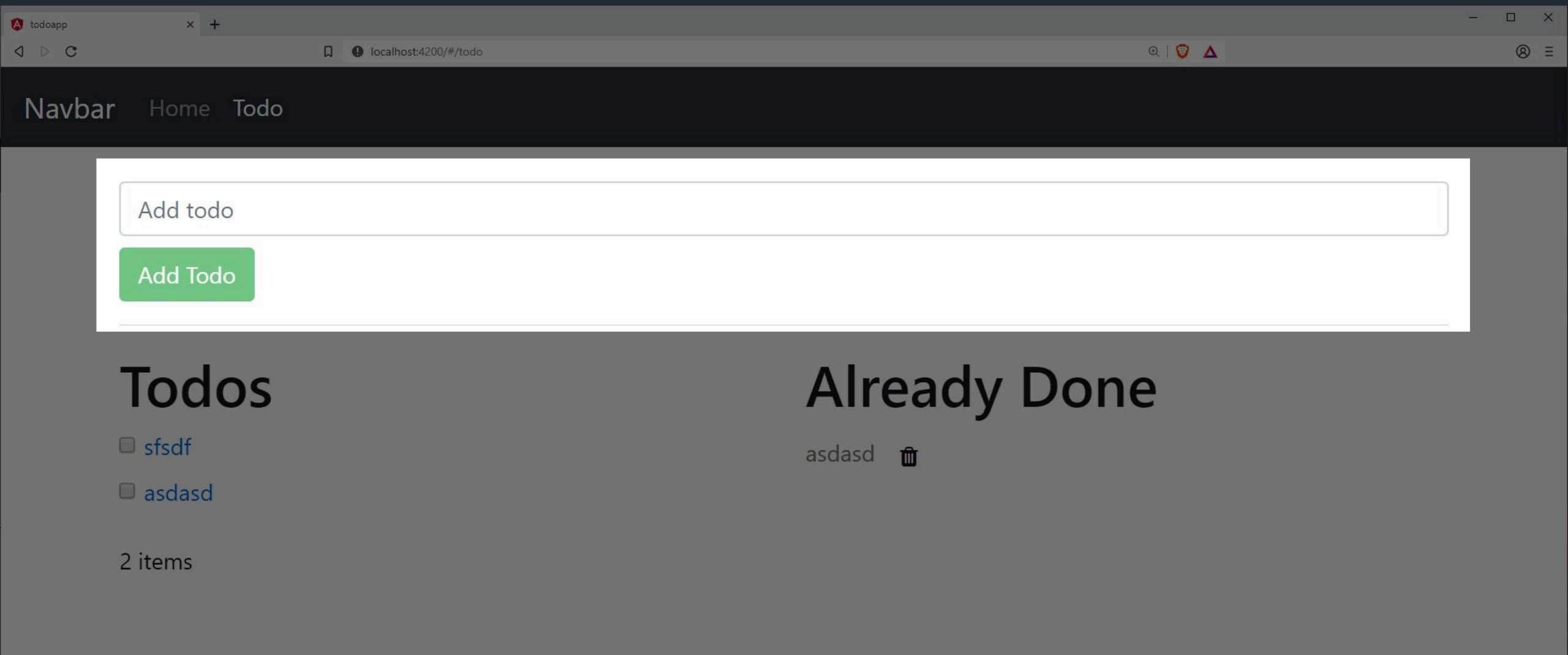
Todos

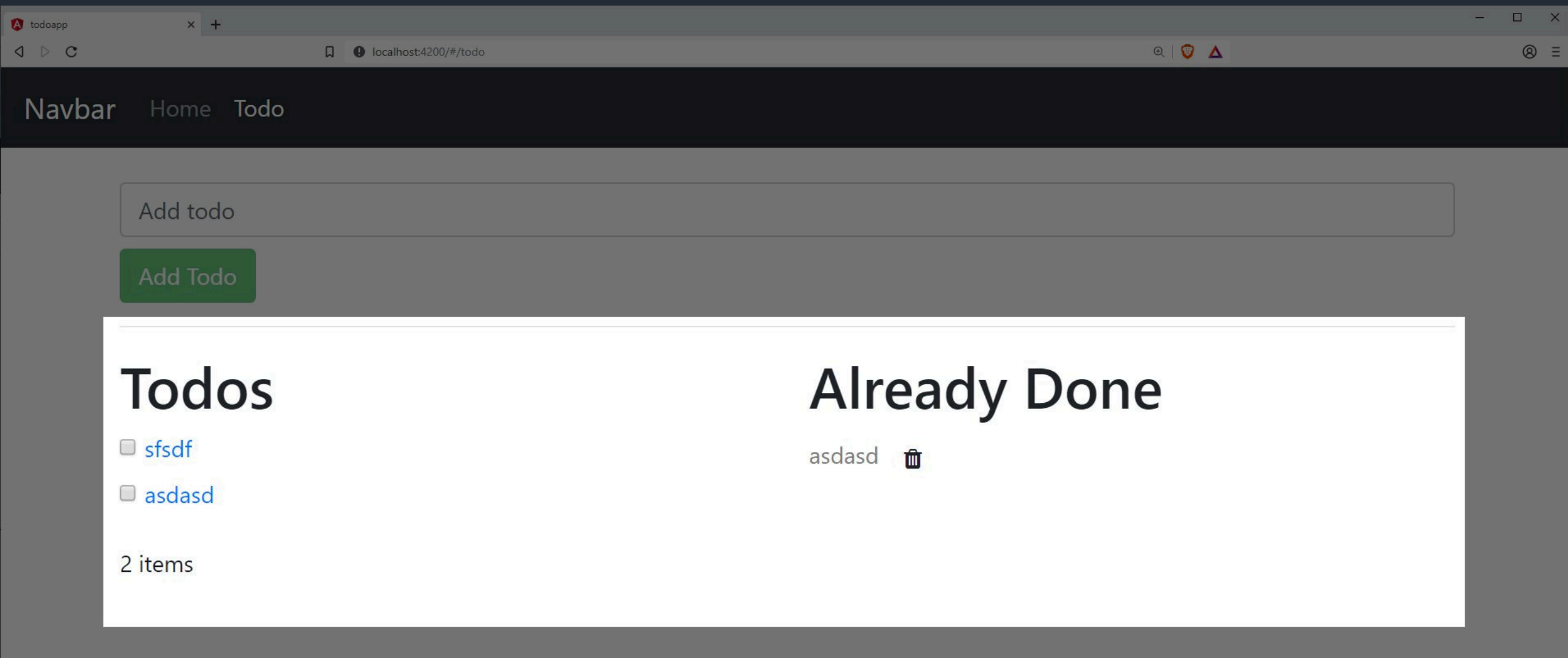
- ☐ sfsdf
- ☐ asdasd

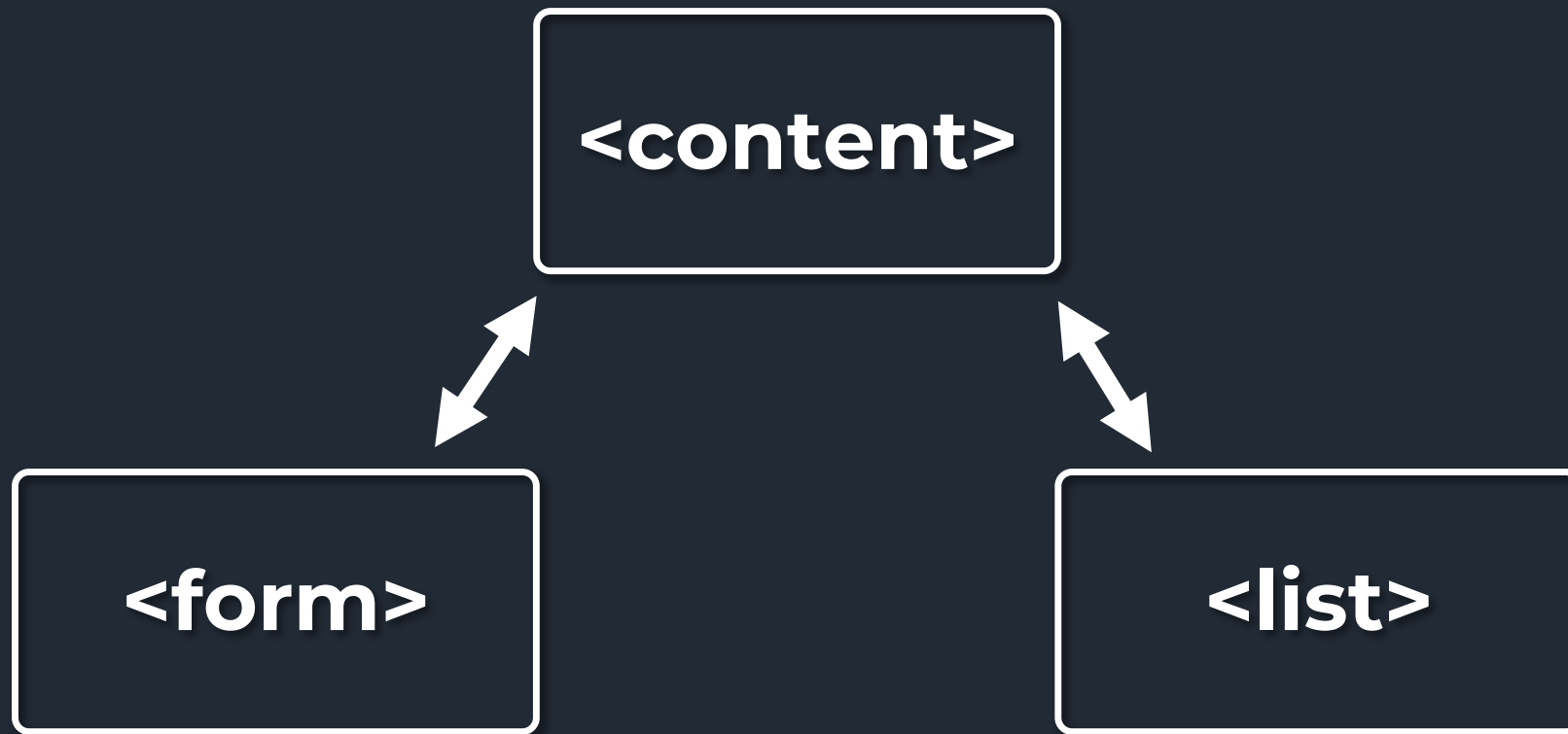
2 items

Already Done

asdasd 

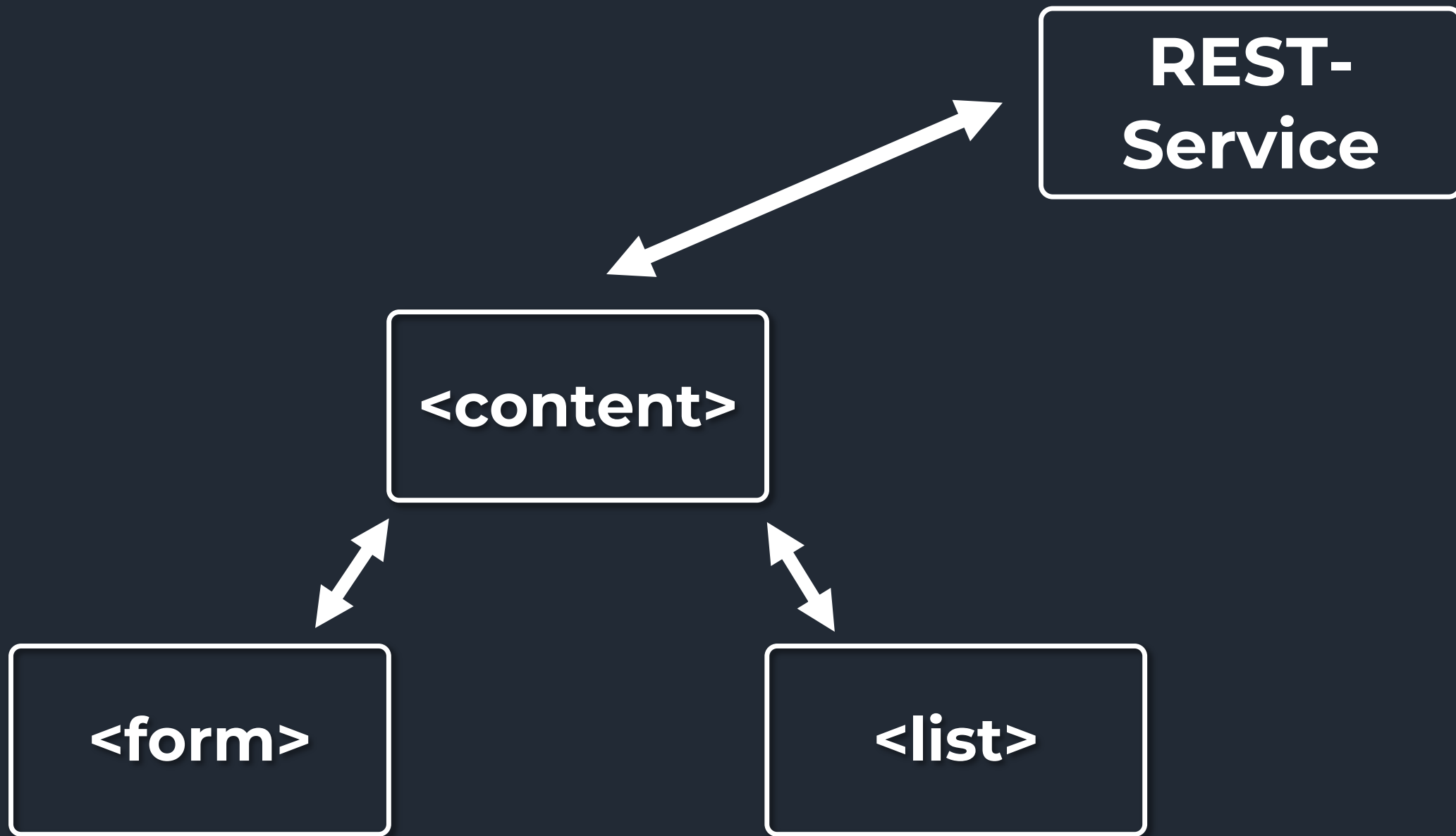


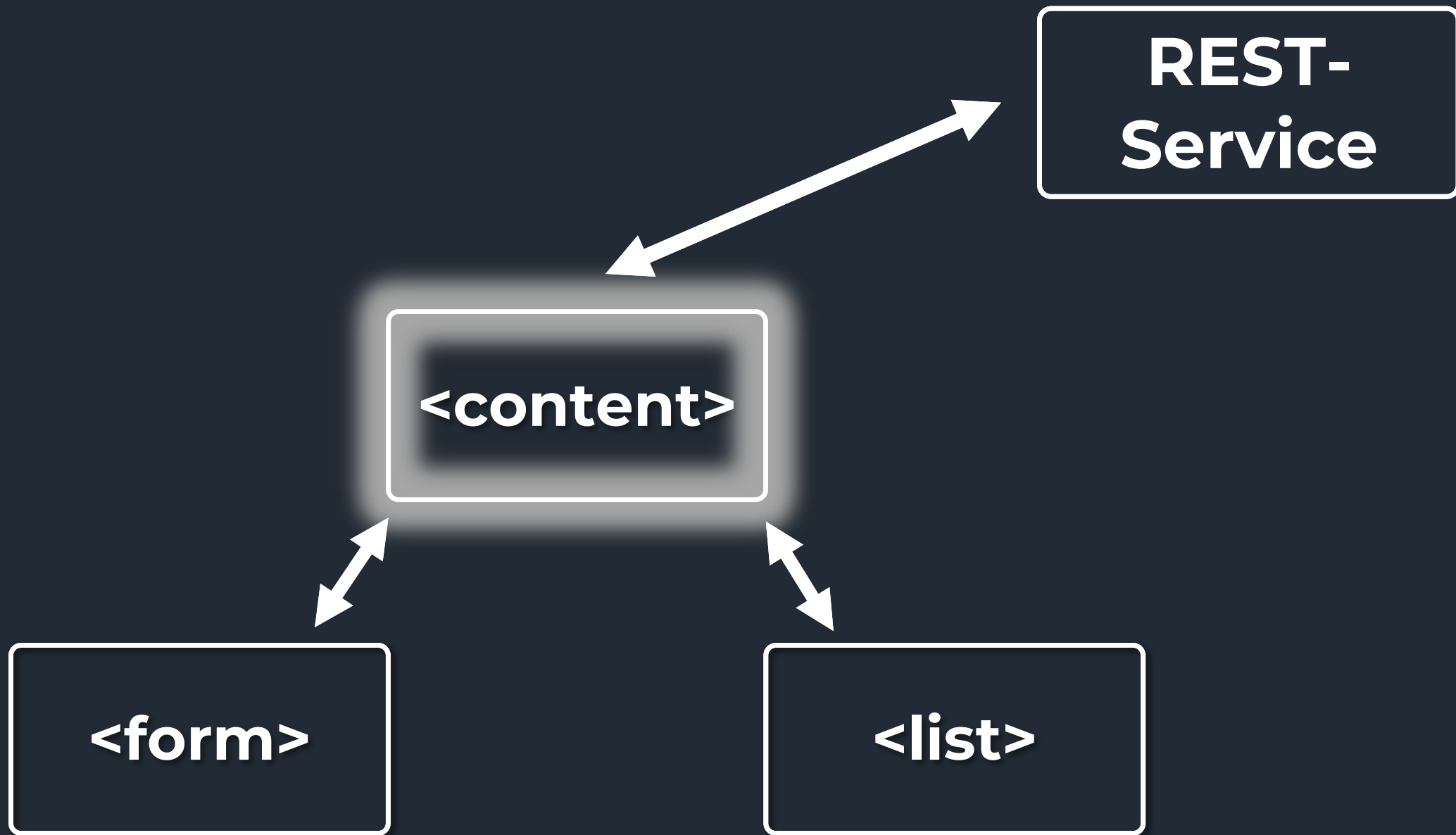


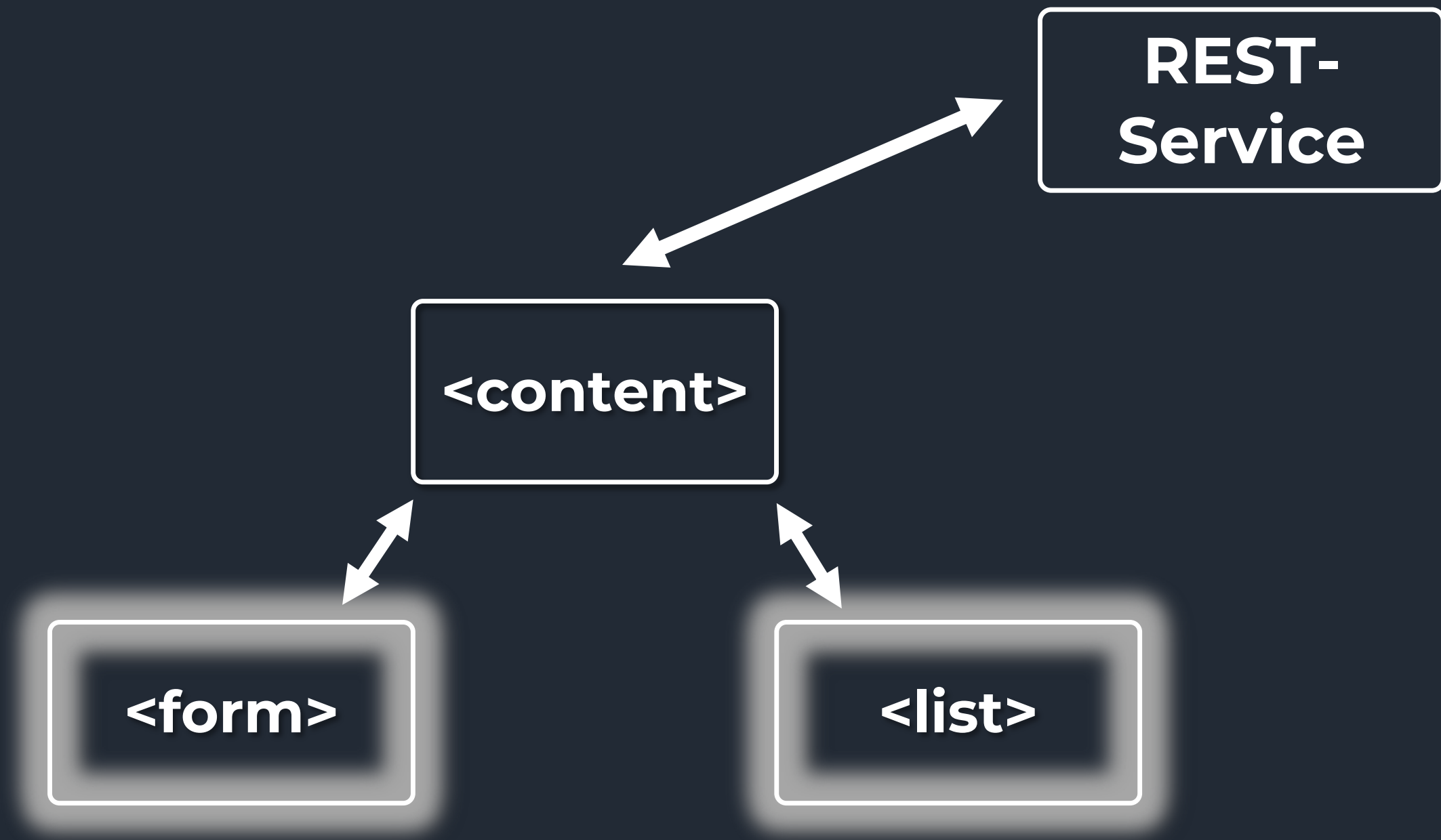


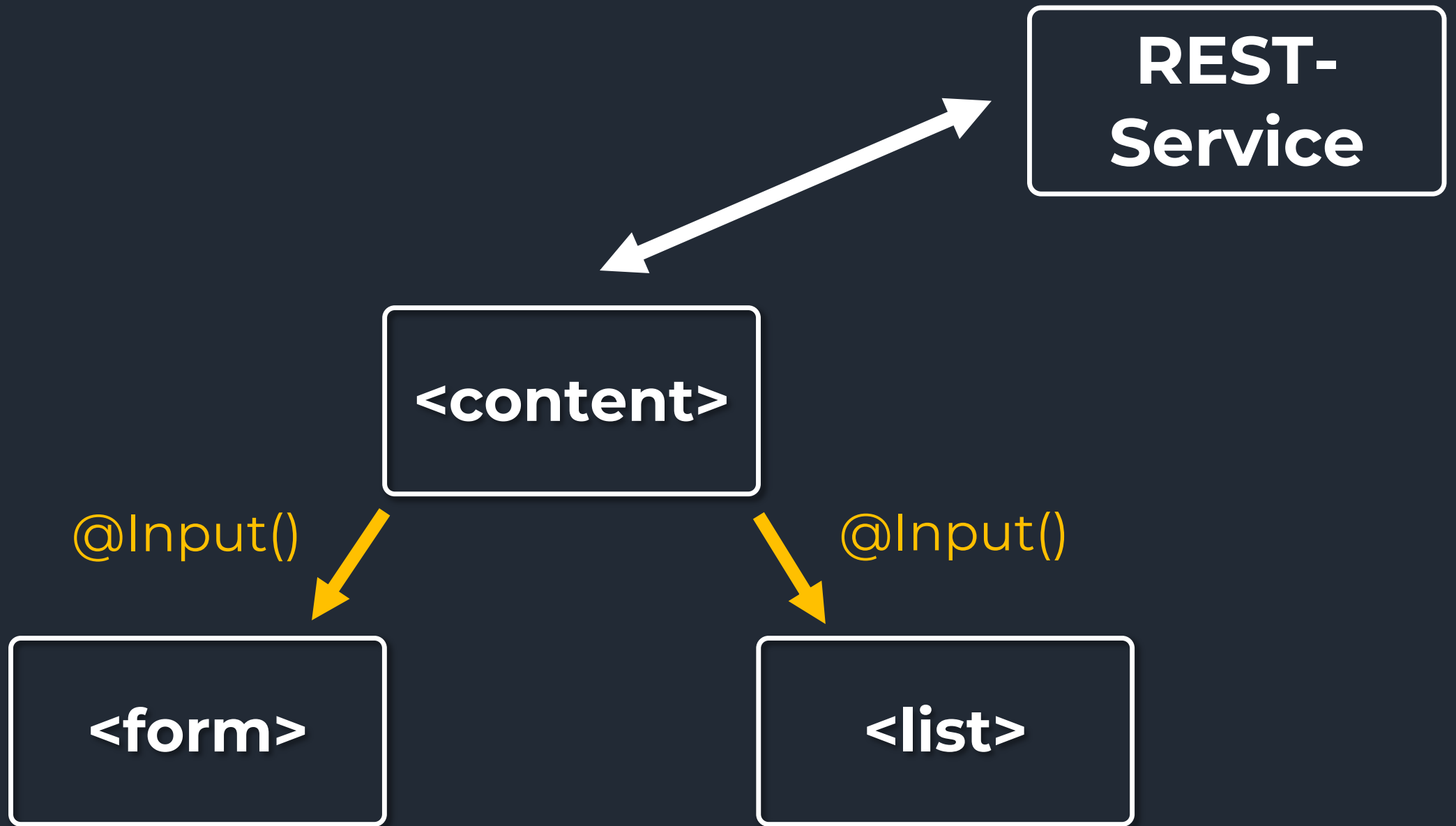


```
<div>  
  <app-todo-form></app-todo-form>  
  
  <app-todo-list></app-todo-list>  
</div>
```







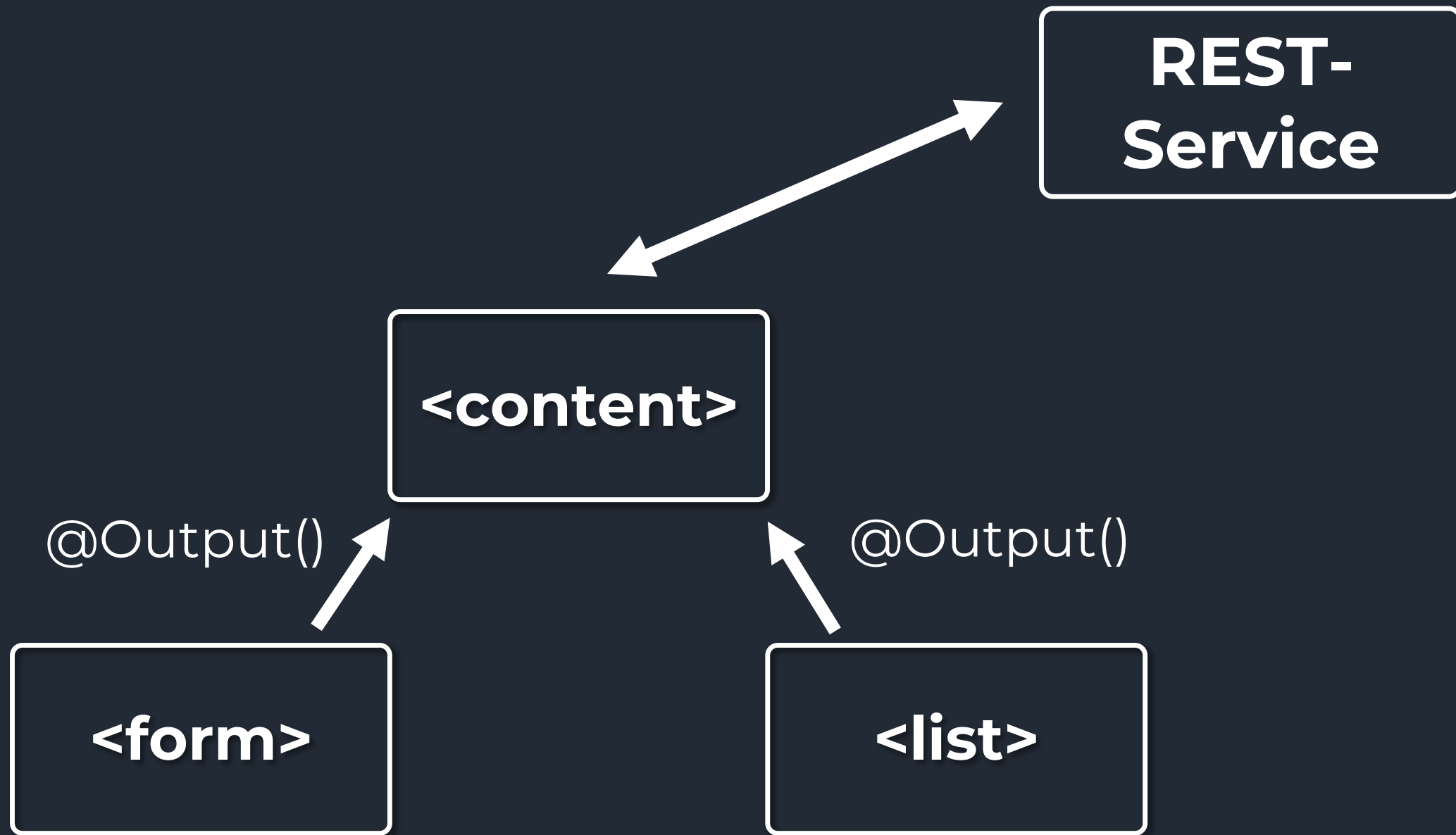




```
@Component({  
  selector: 'app-todo-list'  
})  
export class TodoListComponent {  
  @Input() items: Todo[] = [];  
  @Input() doneItems: Todo[] = [];  
}
```



```
<div>  
  <app-todo-form></app-todo-form>  
  
  <app-todo-list  
    [items]="items"  
    [doneItems]="doneItems"  
  ></app-todo-list>  
</div>
```





```
@Component({  
  selector: 'app-todo-list'  
})  
export class TodoListComponent {  
  @Input() items: Todo[] = [];  
  @Input() doneItems: Todo[] = [];  
  
  @Output() markAsDone = new EventEmitter();  
  @Output() delete = new EventEmitter();  
}
```




```
<div>
  <app-todo-form (todoAdded)="addTodo($event)"></app-todo-form>

  <app-todo-list
    [items]="items"
    [doneItems]="doneItems"
    (markAsDone)="markAsDone($event)"
    (delete)="deleteItem($event)"
  ></app-todo-list>
</div>
```

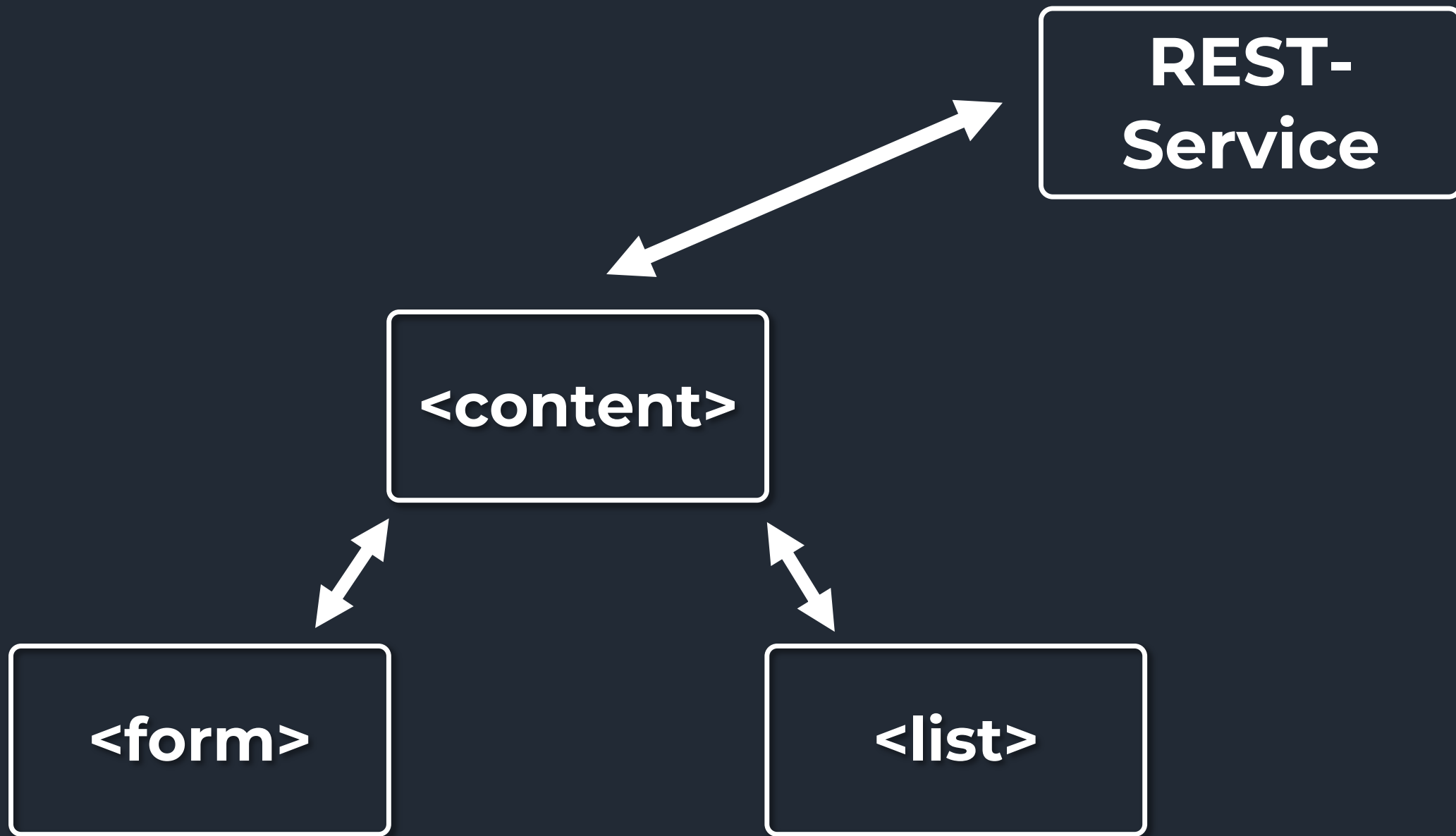


```
@Component({
  selector: 'app-content'
})
export class ContentComponent implements OnInit {
  items: Todo[];

  constructor(private todoService: TodoService) {}

  ngOnInit() {
    this.todoService.getItems().subscribe(result => {
      this.items = result;
    });
  }

  addTodo(item: string) {
    this.todoService.addItem(item).subscribe(result => {
      this.items = [...this.items, result];
    });
  }
}
```



A city skyline at sunset with a warm orange glow. The sky is a deep orange, and the city buildings are silhouetted against it. The water in the foreground is also a warm orange color.

Data FlowSDown

@FabianGosebrink

A city skyline, likely San Francisco, is silhouetted against a bright orange sunset sky. The sun is low on the left, creating a strong glow. The city's skyscrapers are dark against the light sky. In the background, a body of water and distant hills are visible. The text 'Data Flows Up' is overlaid in the lower half of the image.

Data Flows Up

@FabianGosebrink



One Way



State

Where

An aerial photograph of a city skyline, featuring numerous skyscrapers and buildings. The image is overlaid with a semi-transparent teal color. The word "Where" is written in a light gray, sans-serif font in the upper left, and the word "Change" is written in a large, white, sans-serif font in the center, spanning across the middle of the image.

Where Change

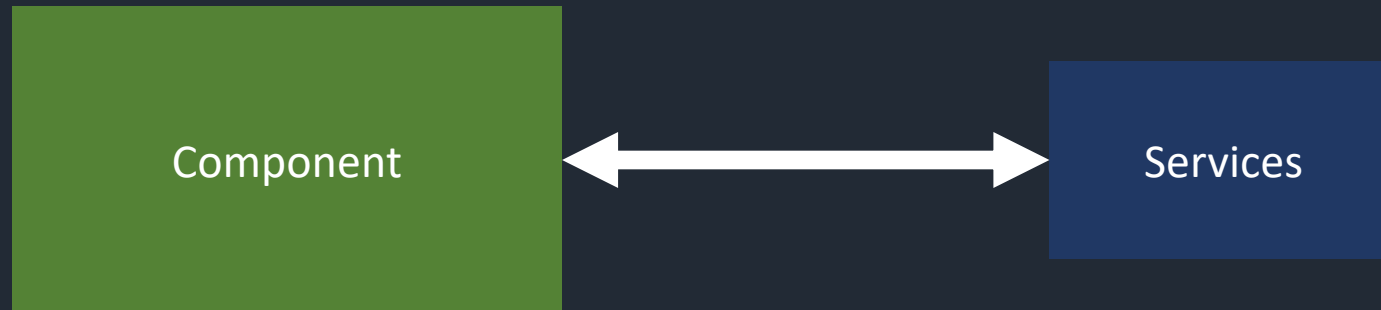
The background of the image is a city skyline, likely New York City, with the Empire State Building prominently visible. A semi-transparent teal overlay covers the entire image. The text 'Where Change Effects' is written in a large, sans-serif font. 'Where' and 'Change' are in a light grey color, while 'Effects' is in white. The text is arranged in three lines, centered horizontally.

Where

Change

Effects





State

A dark, atmospheric photograph of a city street intersection, likely in New York City. The scene is viewed from a low angle, looking down the street. On the left, a Starbucks Coffee shop is visible with its logo and name. Several yellow taxis are parked or moving in the street. Pedestrians are walking across the crosswalk. On the right, a red brick building with many windows is visible. The overall tone is moody and urban. The word "State" is overlaid in large, white, sans-serif font in the center of the image.



```
export interface ReducerTodoState {  
  items: Todo[];  
  selectedItem: Todo;  
  loading: boolean;  
}
```



```
export interface ReducerTodoState {  
  items: Todo[];  
  selectedItem: Todo;  
  loading: boolean;  
}
```

```
export const initialState: ReducerTodoState = {  
  items: [],  
  selectedItem: null,  
  loading: false  
};
```

Component

Actions

A dark, atmospheric photograph of a city street intersection, likely in New York City. The scene is viewed from a low angle, looking down the street. On the left, a yellow taxi is stopped at a crosswalk. In the center, a pedestrian is crossing the street. On the right, another yellow taxi is visible. The background shows tall brick buildings and traffic lights. The word "Actions" is overlaid in large, white, sans-serif font across the center of the image.



```
{
```

```
  type: '[Todo] Load Todos'
```

```
}
```



```
{  
  type: '[Todo] Load Todos',  
  payload?: ...  
}
```




```
import { Action } from '@ngrx/store';  
  
export class LoadAllTodosAction implements Action {  
  readonly type = '[Todo] Load Todos';  
}
```

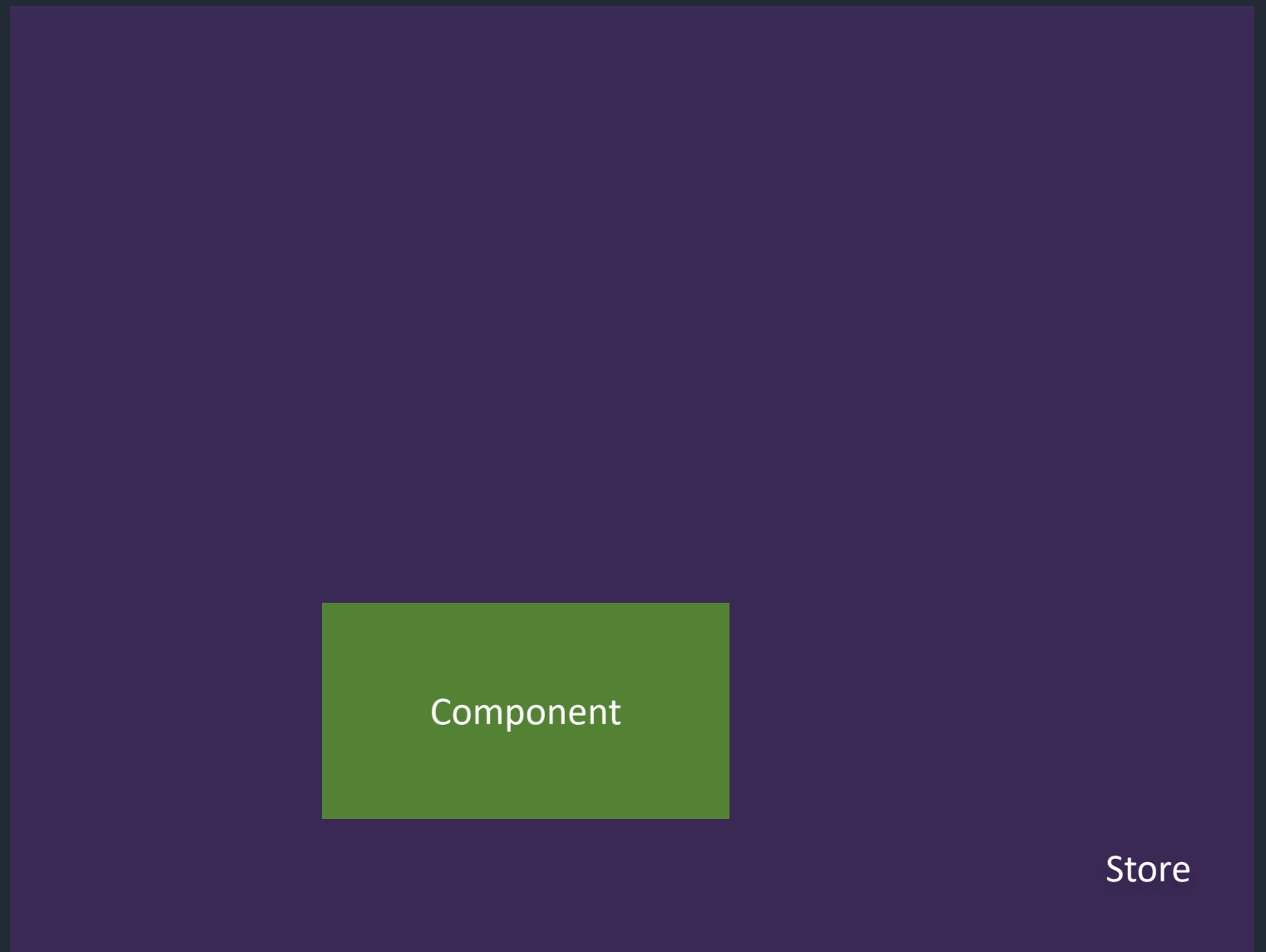


```
import { Action } from '@ngrx/store';

export class LoadAllTodosAction implements Action {
  readonly type = '[Todo] Load Todos';
}

export class LoadAllTodosFinishedAction implements Action {
  readonly type = '[Todo] Load Todos Finished';
  constructor(public payload: Todo[]) {}
}
```

Component

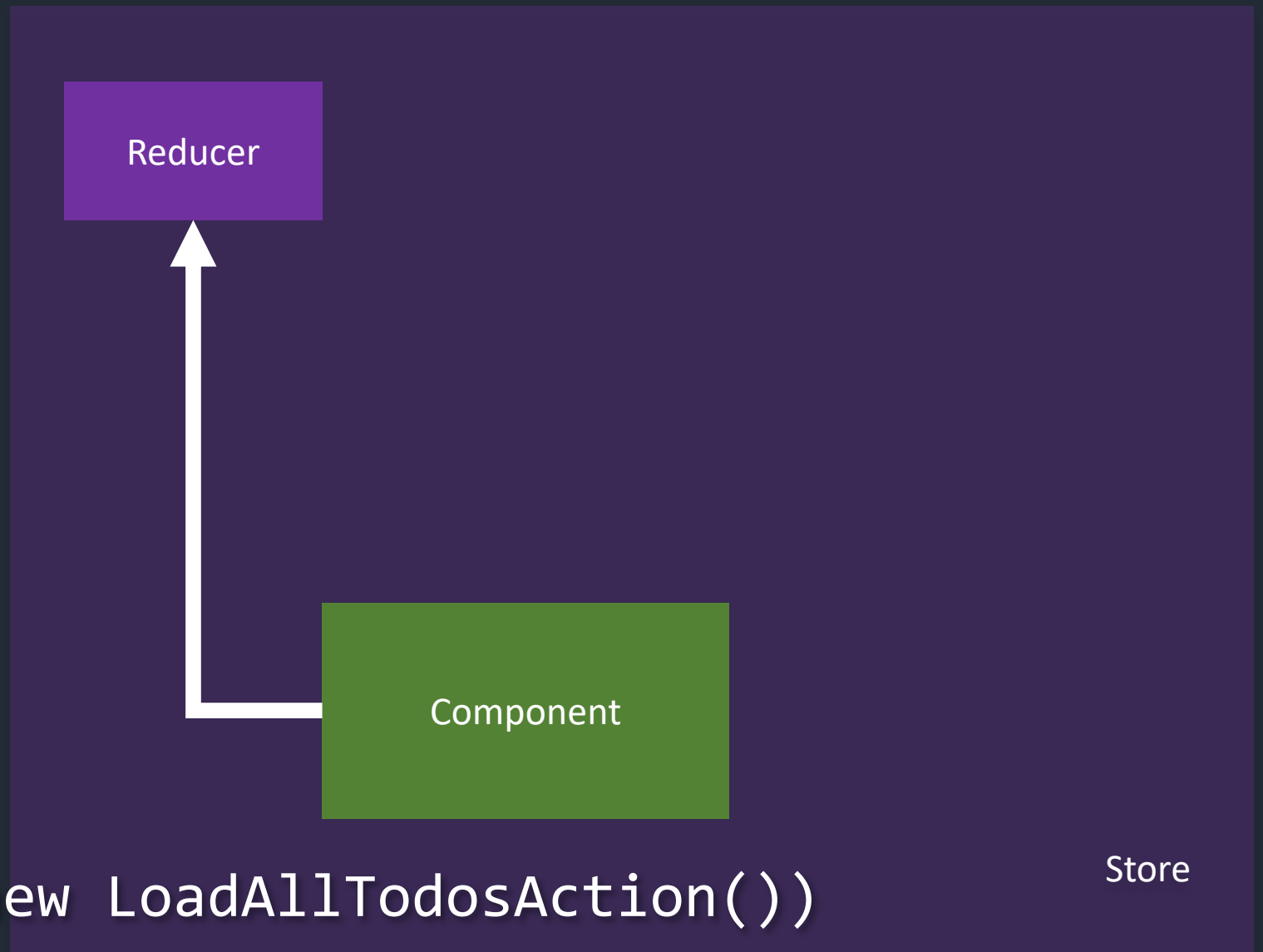


```
store.dispatch(new LoadAllTodosAction())
```

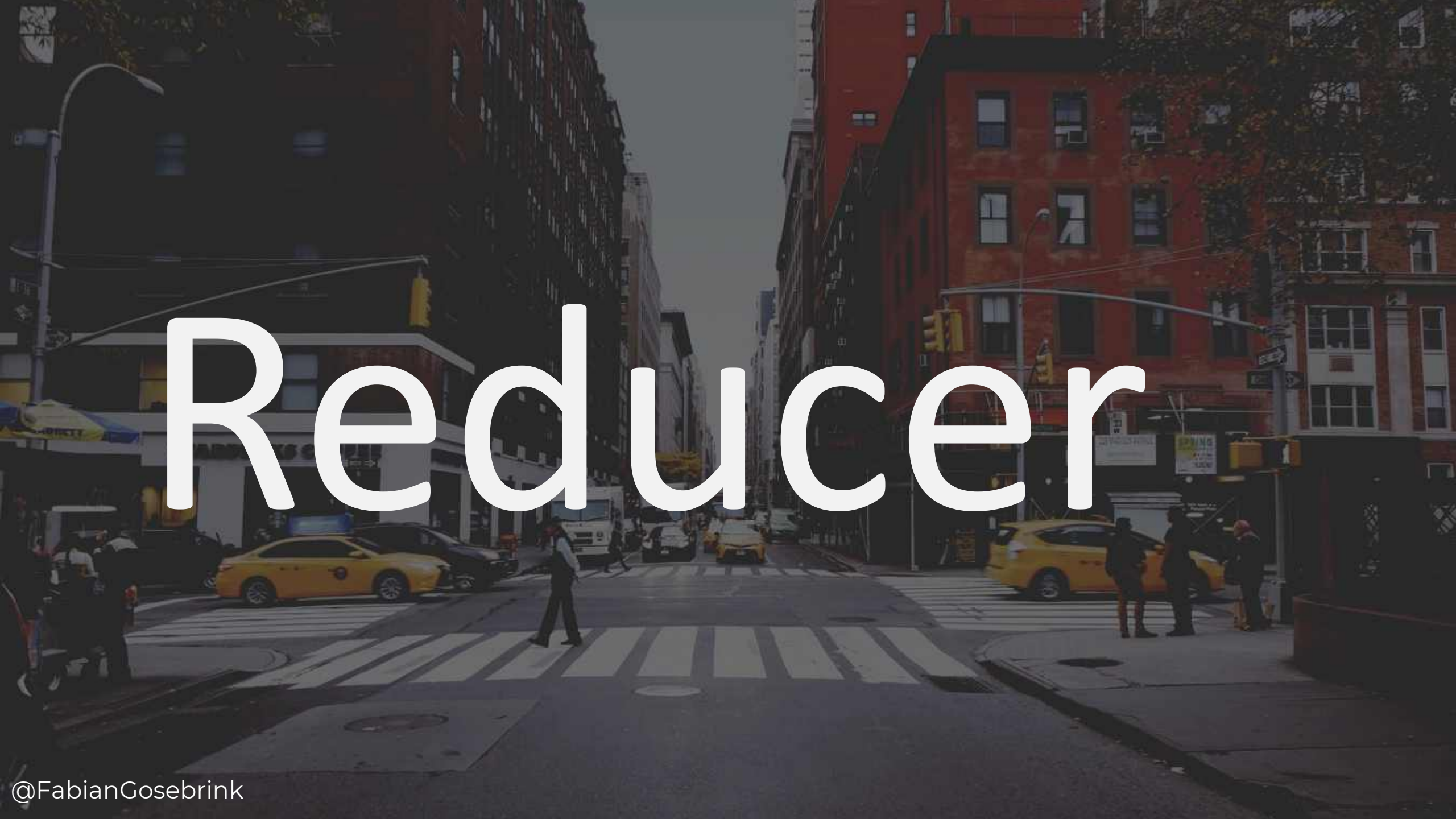
Component

Store

```
store.dispatch(new LoadAllTodosAction())
```



Reducer



$(oldState, action) \Rightarrow newState$



```
export interface ReducerTodoState { ... }

export function todoReducer(state, action): ReducerTodoState {
  switch (action.type) {

    default:
      return state;
  }
}
```



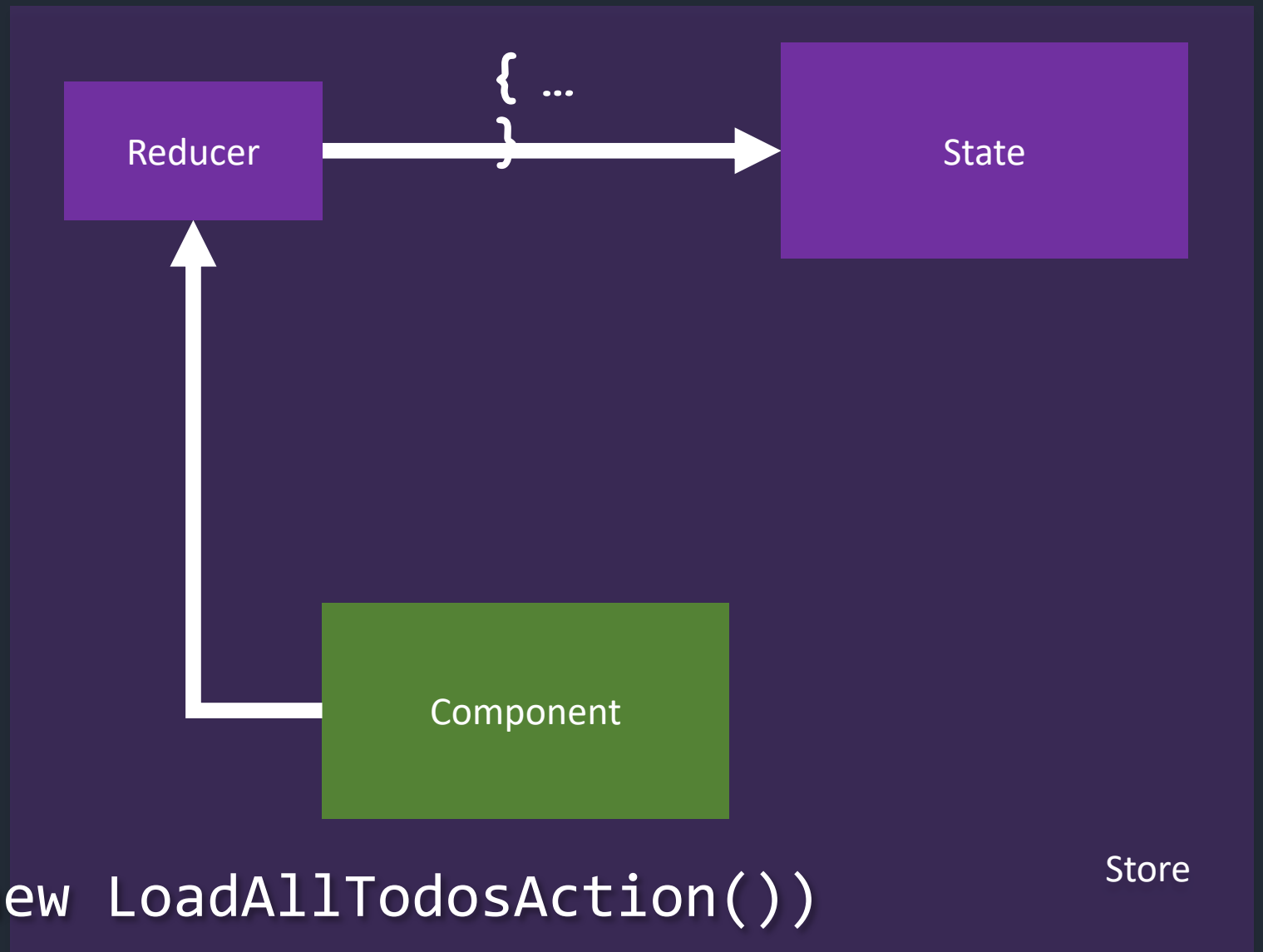
```
export interface ReducerTodoState { ... }

export function todoReducer(state, action): ReducerTodoState {
  switch (action.type) {

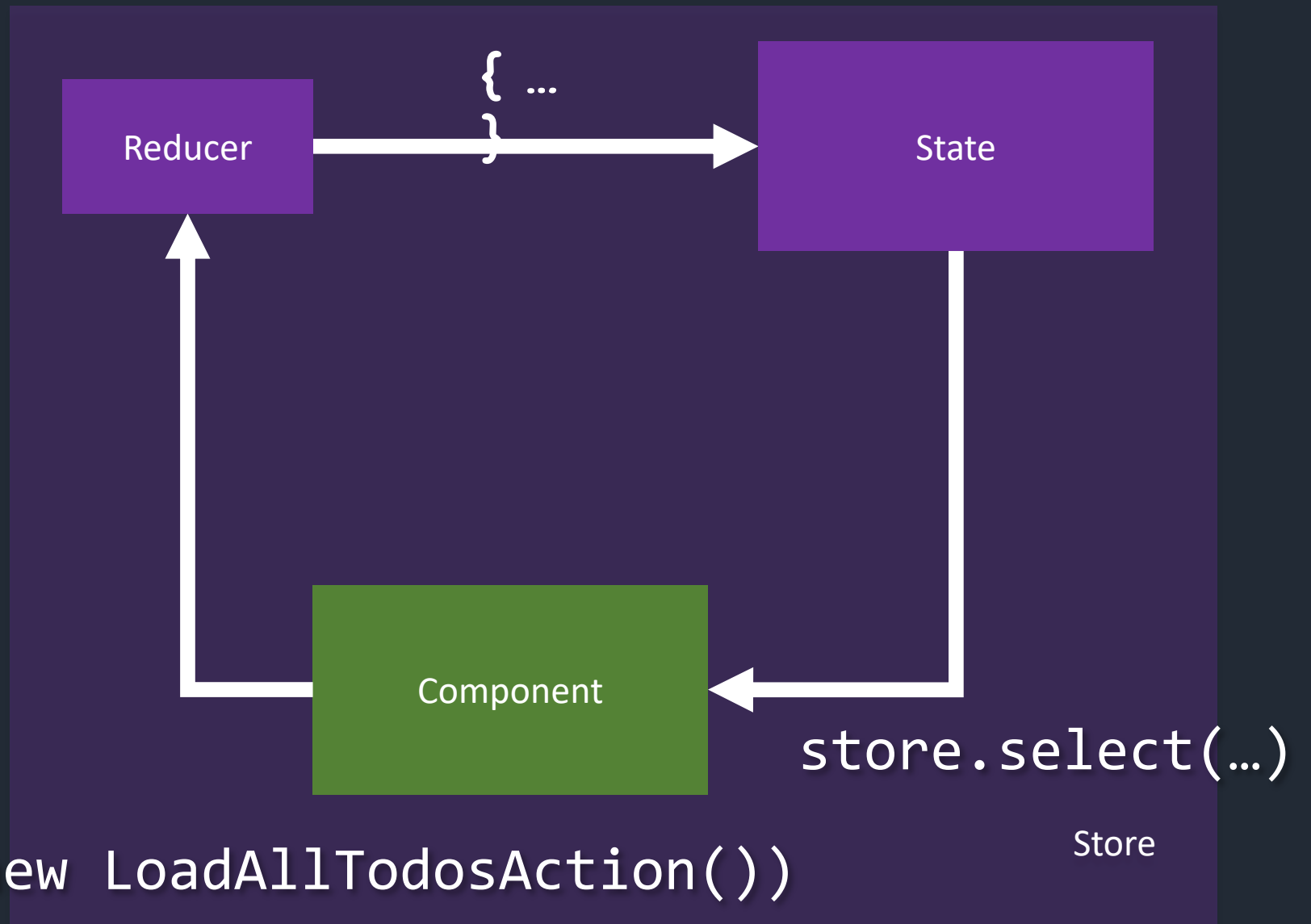
    case '[Todo] Load Todos': {
      return {
        ...state,
        loading: true
      };
    }

    default:
      return state;
  }
}
```

```
store.dispatch(new LoadAllTodosAction())
```



```
store.dispatch(new LoadAllTodosAction())
```

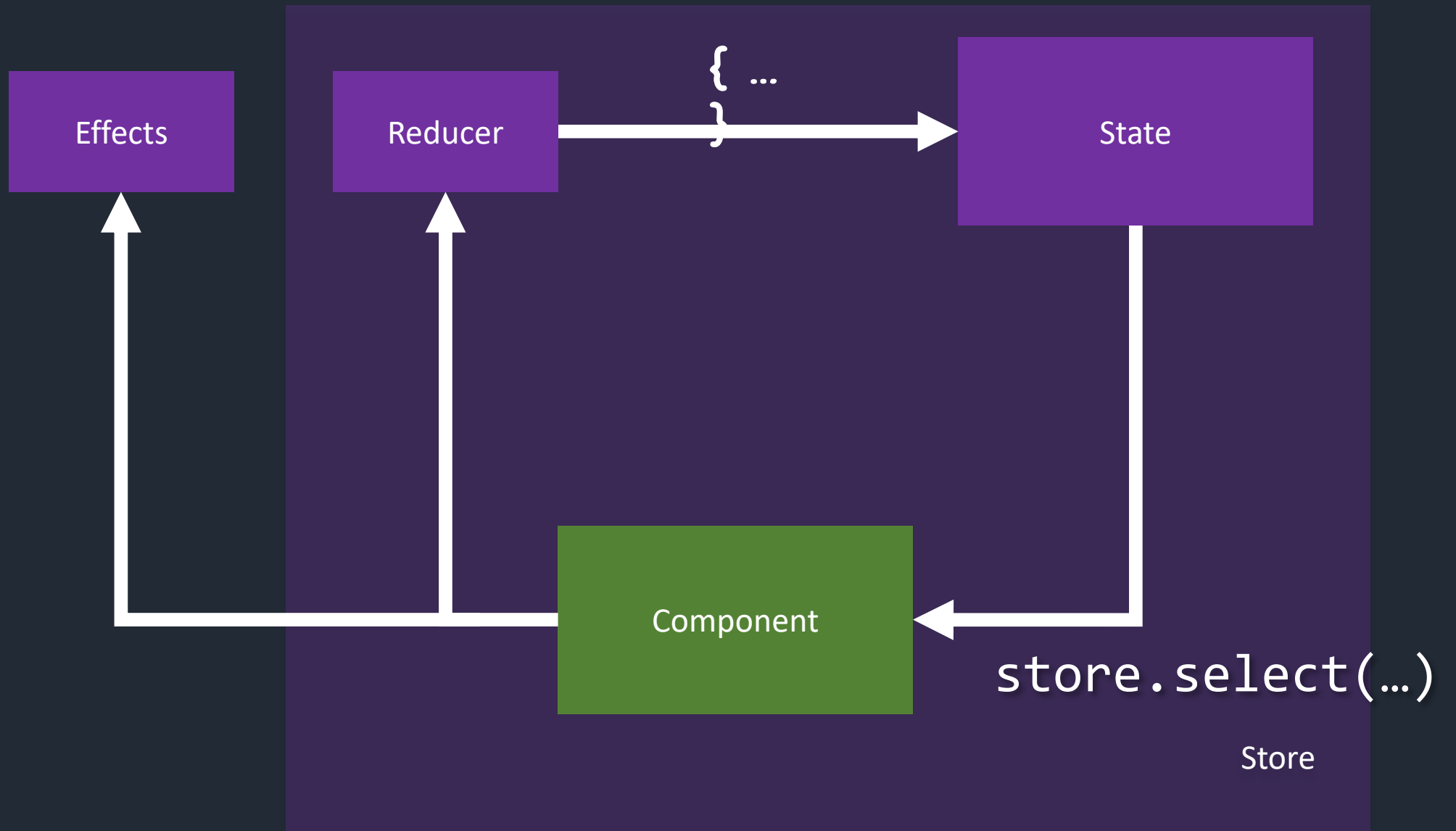


Container component	Presentationa component
<code>store.select(...)</code>	<code>@Input(...)</code>
<code>store.dispatch(...)</code>	<code>@Output(...)</code>

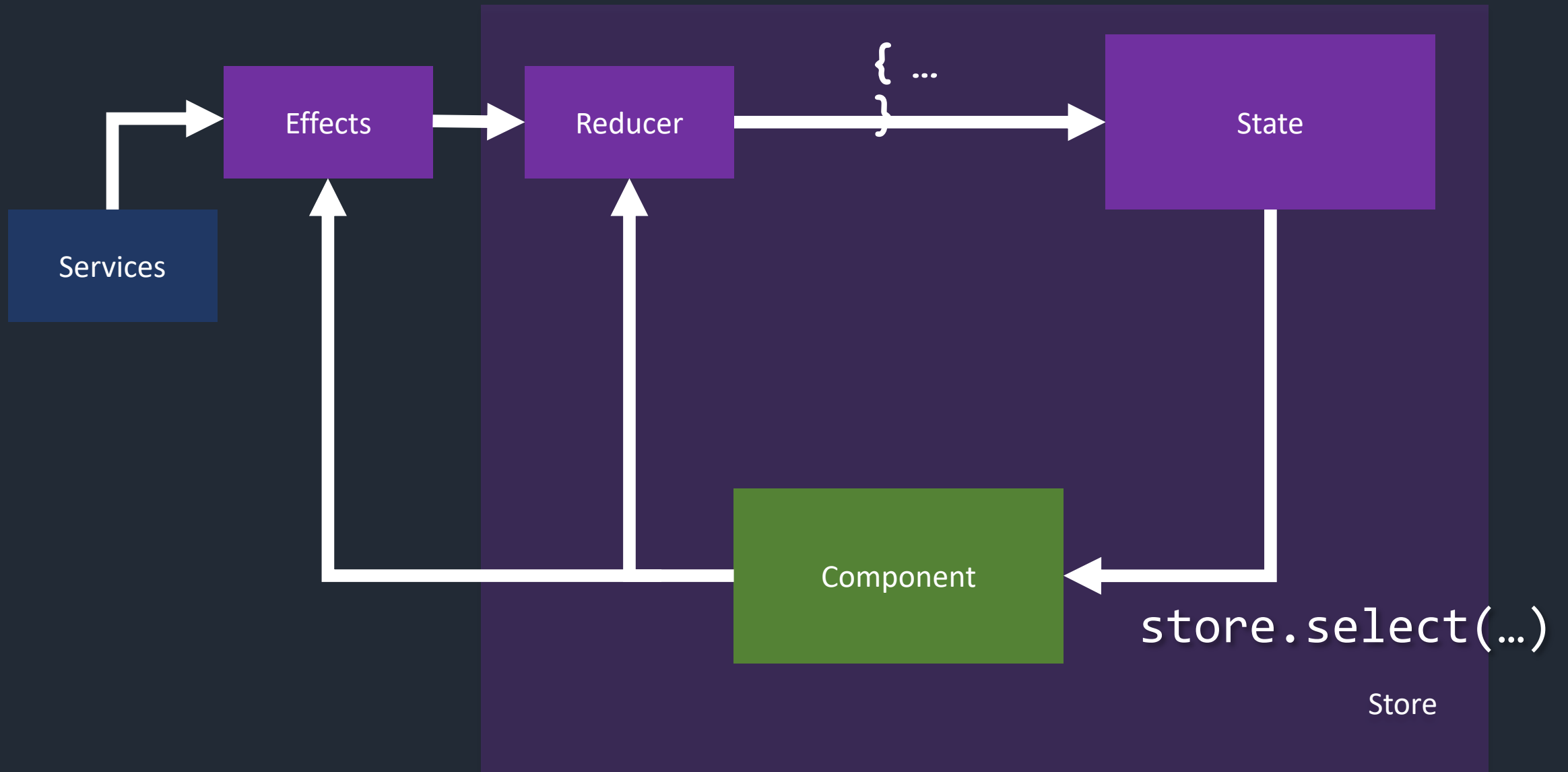


Effects

A dark, atmospheric photograph of a city street intersection, likely in New York City. The scene is viewed from a low angle, looking down the street. On the left, a Starbucks is visible. Several yellow taxis are parked or moving. Pedestrians are crossing the street. The word "Effects" is overlaid in large, white, sans-serif font across the center of the image.



```
store.dispatch(new LoadAllTodosAction())
```



```
store.dispatch(new LoadAllTodosAction())
```



```
@Injectable()
export class TodoEffects {
  constructor(
    private actions$: Actions,
    private todoService: TodoService) {}

  @Effect()
  loadTodos$ = this.actions$.pipe(
    ofType(ActionTypes.LoadAllTodos),
    switchMap(() =>
      this.todoService.getItems()
        .pipe(
          map(
            (todos: Todo[]) => new LoadAllTodosFinishedAction(todos)
          )
        )
    )
  );
}
```




```
export function todoReducer(state, action): ReducerTodoState {  
  switch (action.type) {  
    case ActionTypes.LoadAllTodos: {  
      ...  
    }  
  
    case ActionTypes.LoadAllTodosFinished: {  
      return {  
        ...state,  
        loading: false,  
        items: action.payload  
      };  
    }  
  
    default:  
      return state;  
  }  
}
```

▲ ANGULAR-NGRX-TODO

-  e2e
- ▲  src
 - ▲  app
 -  core
 -  models
 -  shared
 -  start
 - ▲  todo
 -  container
 -  presentational
 - ▲  store
 - TS index.ts
 - TS todo.actions.ts
 - TS todo.effects.ts
 - TS todo.reducer.ts
 -  todo.module.ts
 -  app.component.css



```
import { StoreModule } from '@ngrx/store';

@NgModule({
  imports: [
    StoreModule.forRoot(...)
  ],
})
export class AppModule {}
```

{ ... }



```
import { StoreModule } from '@ngrx/store';

@NgModule({
  imports: [
    StoreModule.forRoot({}),
  ],
})
export class AppModule {}
```





```
import { StoreModule } from '@ngrx/store';

@NgModule({
  imports: [
    StoreModule.forFeature(...),
  ],
})
export class TodoModule {}
```




```
import { StoreModule } from '@ngrx/store';  
import { todoReducer } from '../store/todo.reducer';  
  
@NgModule({  
  imports: [  
    StoreModule.forFeature("todoFeature", todoReducer),  
  ],  
})  
export class TodoModule {}
```



```
import { StoreModule } from '@ngrx/store';
import { todoReducer } from '../store/todo.reducer';

@NgModule({
  imports: [
    StoreModule.forFeature("todoFeature", todoReducer),
  ],
})
export class TodoModule {}
```

```
{
  todoFeature: {
    // ...
  }
}
```



```
import { StoreModule } from '@ngrx/store';
import { todoReducer } from '../store/todo.reducer';

@NgModule({
  imports: [
    StoreModule.forFeature("todoFeature", todoReducer),
  ],
})
export class TodoModule {}
```

```
{
  todoFeature: {
    items: [],
    selectedItem:
null,
    loading: false
  }
}
```



Ben Lesh 🇺🇸👑🔥

@BenLesh

Following



Oh.. and on the anti-redux-because-boilerplate front:

That "boilerplate" is directly the result of separation of concerns: You have to define many separate things. It's more than you're used to, because you weren't separating concerns before when your app was a big pile of code.

3:44 AM - 13 Feb 2018

26 Retweets 108 Likes



6



26



108



Tweet your reply



Ben Lesh 🇺🇸👑🔥 @BenLesh · Feb 13



Tangling state and rendering together into one big, freaky component, or many tiny, cute, little, freaky components doesn't mean you've separated concerns.



2



3



25







```
export class ContentComponent implements OnInit {
  items: Todo[];
  doneItems: Todo[];

  constructor(private todoService: TodoService) {}

  ngOnInit() {
    this.getData();
  }

  addTodo(item: string) {
    this.todoService.addItem(item).subscribe(
      () => {
        this.getData();
      },
      error => console.log(error)
    );
  }

  markAsDone(item: Todo) {
    this.todoService.updateItem(item).subscribe(
      () => {
        this.getData();
      },
      error => console.log(error)
    );
  }

  private getData() {
    this.todoService.getItems().subscribe(
      items => {
        this.items = items.filter(x => !x.done);
        this.items = items.filter(x => x.done);
      },
      error => console.log(error)
    );
  }
}
```




```
export class ContentComponent implements OnInit {
  items$: Observable<Todo[]>;
  doneItems$: Observable<Todo[]>;

  constructor(private store: Store<any>) {}

  ngOnInit() {
    this.items$ = this.store.pipe(select(/*...*/));
    this.doneItems$ = this.store.pipe(select(/*...*/));

    this.store.dispatch(new LoadAllTodosAction());
  }

  addTodo(item: string) {
    this.store.dispatch(new AddTodoAction(item));
  }

  markAsDone(item: Todo) {
    this.store.dispatch(new SetAsDoneAction(item));
  }
}
```

Demo

A dark, atmospheric street scene in New York City. The image is dimly lit, with a large white word 'Demo' centered over the middle. On the left, a Starbucks Coffee shop is visible with its logo. Several yellow taxis are parked or moving on the street. Pedestrians are walking on the sidewalks. The background shows tall brick buildings and a street sign.



```
import { createAction, props } from '@ngrx/store';  
  
export const addTodo = createAction(  
  '[Todo] Add Todo',  
  props<{ value: string }>()  
)
```

A dark, atmospheric photograph of a city street intersection, likely in New York City. The scene is viewed from a low angle, looking down the street. Tall, multi-story buildings line both sides of the street. On the left, a building has a sign that partially reads "STARBUCKS". Several yellow taxis are visible, including one in the foreground on the left and another on the right. Pedestrians are walking across the street, and a few vehicles are stopped at a traffic light. The overall tone is moody and urban.

@ngrx/data



```
import { EntityMetadataMap } from '@ngrx/data';

const entityMetadata: EntityMetadataMap = {
  Todo: {},
  // Add other items here
};

export const entityConfig = {
  entityMetadata
};
```



```
import { NgModule } from '@angular/core';
import { EffectsModule } from '@ngrx/effects';
import { StoreModule } from '@ngrx/store';
import { DefaultDataServiceConfig, EntityDataModule } from '@ngrx/data';
import { entityConfig } from './entity-metadata';

@NgModule({
  imports: [
    StoreModule.forRoot({}),
    EffectsModule.forRoot([]),
    EntityDataModule.forRoot(entityConfig)
  ]
})
export class AppModule {}
```




```
import { Injectable } from '@angular/core';
import {
  EntityCollectionServiceBase,
  EntityCollectionServiceElementsFactory
} from '@ngrx/data';
import { Todo } from '../todo.model';

@Injectable({ providedIn: 'root' })
export class TodoService extends EntityCollectionServiceBase<Todo> {
  constructor(serviceElementsFactory: EntityCollectionServiceElementsFactory) {
    super('Todo', serviceElementsFactory);
  }
}
```



```
export class TodoComponent implements OnInit {  
  loading$: Observable<boolean>;  
  todos$: Observable<Todo[]>;  
  
  constructor(private todoService: TodoService) {  
    this.todos$ = todoService.entities$;  
    this.loading$ = todoService.loading$;  
  }  
  
  ngOnInit() {  
    this.todoService.getAll();  
  }  
  
  add(todo: Todo) {  
    this.todoService.add(todo);  
  }  
  
  delete(todo: Todo) {  
    this.todoService.delete(todo.id);  
  }  
}
```



Fabian Gosebrink

<http://offering.solutions>

<http://github.com/FabianGosebrink>

<http://fabian-gosebrink.com>

@FabianGosebrink