Angular HTTP Communication

CONFIGURING AN APPLICATION TO MAKE HTTP REQUESTS

Brice Wilson

@brice_wilson www.BriceWilson.net

Course Overview

Consuming REST Services

Course Overview

Consuming REST Services

Advanced HTTP Requests and Error Handling

Course Overview

Consuming REST Services

Advanced HTTP Requests and Error Handling

Creating Interceptors

Course Overview

Consuming REST Services

Advanced HTTP Requests and Error Handling

Creating Interceptors

Caching HTTP Requests with Interceptors

Course Overview

Consuming REST Services

Advanced HTTP Requests and Error Handling

Creating Interceptors

Caching HTTP Requests with Interceptors

Testing HTTP Requests

Demo

Book Tracker project overview

The Role of RxJS

Angular apps are dependent on RxJS

Installed with apps created by the CLI

Most HttpClient methods return Observables

Lots of RxJS operators available to work with Observables

Demo

Preparing to use HttpClient in a project

Consuming REST Services

Brice Wilson

@brice_wilson www.BriceWilson.net

What is a REST Service?

Representational State Transfer

Web API or HTTP API

Uses HTTP verbs to specify CRUD operations

URL conventions address individual as well as collections of resources

HTTP response codes indicate success/failure of server action

RESTful CRUD

C

R

U

D

RESTful CRUD

C

R

U

D

Create

- POST – http://localhost/api/books
- If successful, returns HTTP 201 Created

RESTful CRUD

C
R
U
D

Create

- POST – http://localhost/api/books
- If successful, returns HTTP 201 Created

Read

- GET – http://localhost/api/books OR http://localhost/api/books/5
- If successful, returns HTTP 200 OK

RESTful CRUD

C
R
U
D

Create
- POST – http://localhost/api/books
- If successful, returns HTTP 201 Created

Read
- GET – http://localhost/api/books OR http://localhost/api/books/5
- If successful, returns HTTP 200 OK

Update
- PUT – http://localhost/api/books/5
- If successful, returns HTTP 204 No Content

RESTful CRUD

C
R
U
D

Create
- POST – http://localhost/api/books
- If successful, returns HTTP 201 Created

Read
- GET – http://localhost/api/books OR http://localhost/api/books/5
- If successful, returns HTTP 200 OK

Update
- PUT – http://localhost/api/books/5
- If successful, returns HTTP 204 No Content

Delete
- DELETE – http://localhost/api/books/5
- If successful, returns HTTP 204 No Content

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}
```

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}
```

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}
```

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}
```

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}

this.dataService.getAllBooks()
  .subscribe(



  );
```

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}

this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,


  );
```

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}

this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,
    (err: any) => console.log(err),

  );
```

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}

this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,
    (err: any) => console.log(err),
    () => console.log('All done getting books.')
  );
```

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}

this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,
    (err: any) => console.log(err),
    () => console.log('All done getting books.')
  );
```

Subscribing to Observables

```
getAllBooks(): Observable<Book[]> {
  return this.http.get<Book[]>('/api/books');
}

this.dataService.getAllBooks()
  .subscribe(
    (data: Book[]) => this.allBooks = data,
    (err: any) => console.log(err),
    () => console.log('All done getting books.')
  );
```

Demo

Retrieving a collection from a RESTful service

Demo

Retrieving a single item from a RESTful service

Using RxJS Operators

Operate on an Observable and return an Observable

May be chained together to perform complex transformations

Flexibility to transform data into exactly the shape you need

Demo

Transforming data with RxJS

Demo

Creating, updating, and deleting data in a RESTful service

Advanced HTTP Requests and Error Handling

Brice Wilson

@brice_wilson www.BriceWilson.net

Handling HTTP Errors

Encapsulate HTTP errors in a service

Don’t expose implementation details to the component

Use the RxJS “catchError” operator

Return custom errors to components

Demo

Handling HTTP errors

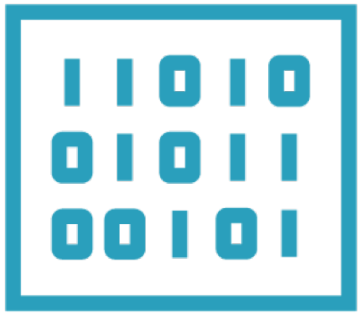
What Are Resolvers?

- Fetch data before navigating
- Prevent presentation of an empty component
- Prevent routing to component with errors
- Provide better user experience (maybe)

Route Transitions Without Resolvers

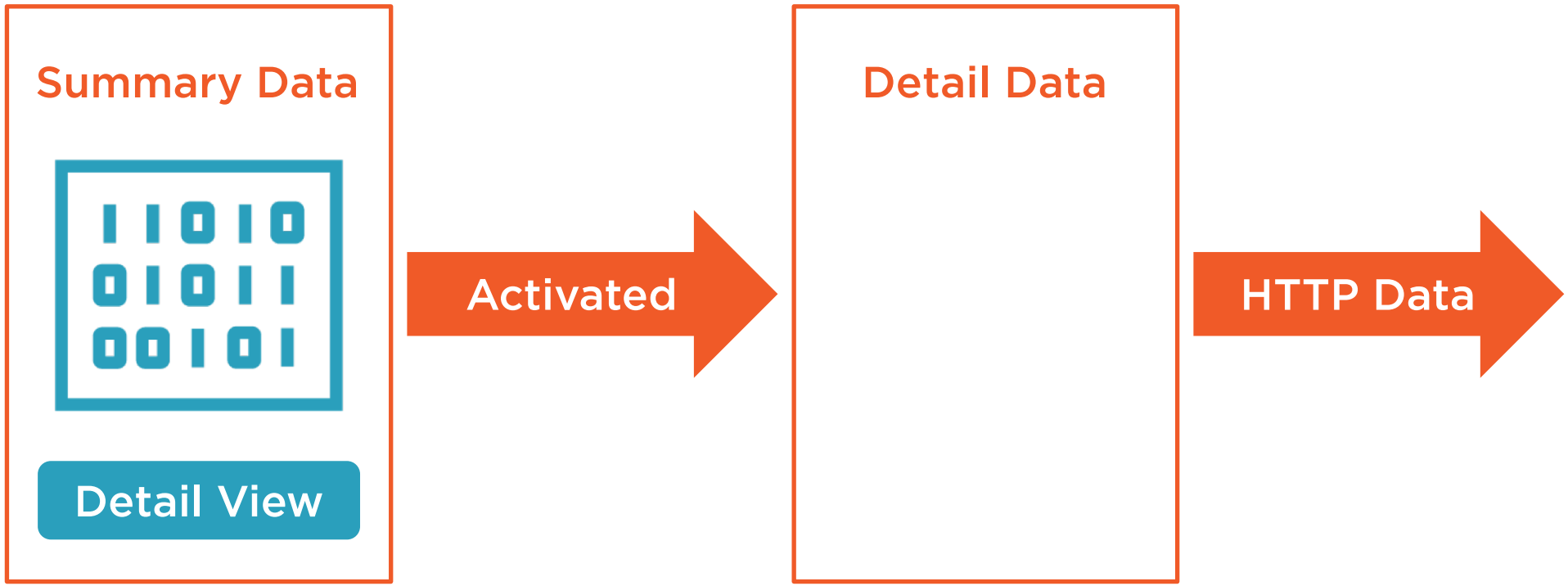
Route Transitions Without Resolvers

Summary Data

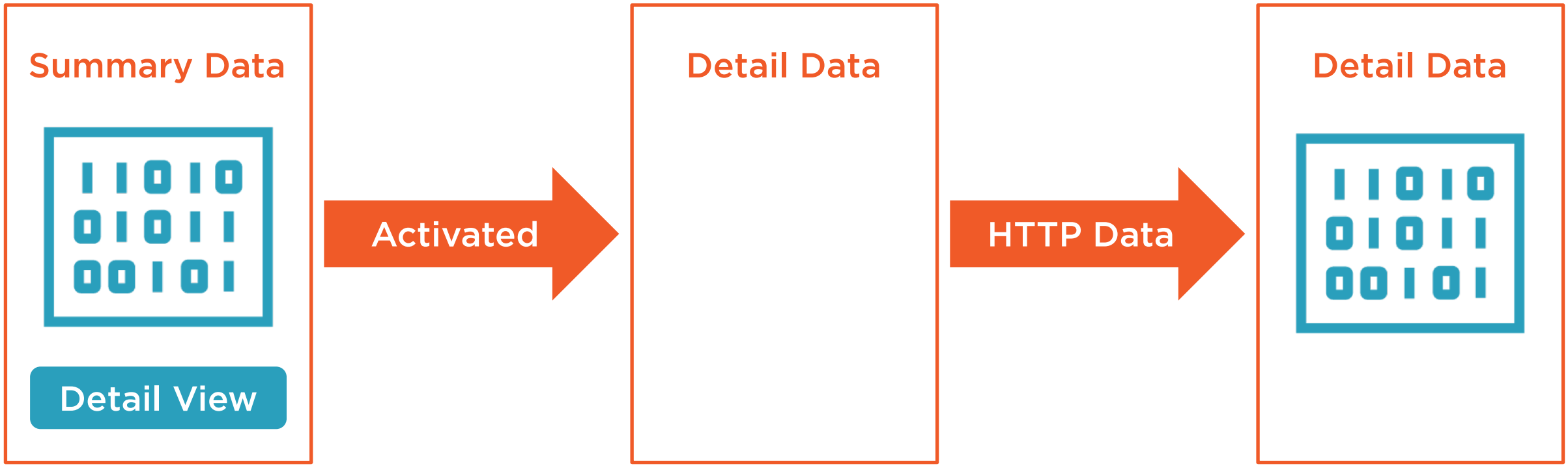


Detail View

Route Transitions Without Resolvers



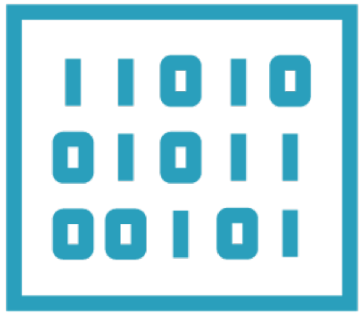
Route Transitions Without Resolvers



Route Transitions With Resolvers

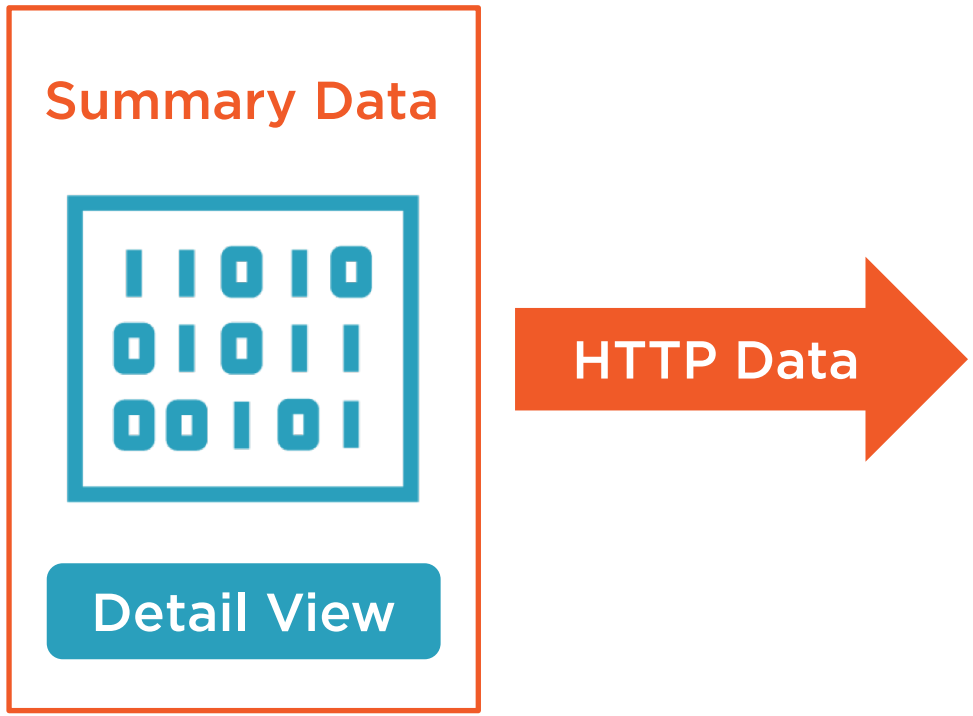
Route Transitions With Resolvers

Summary Data

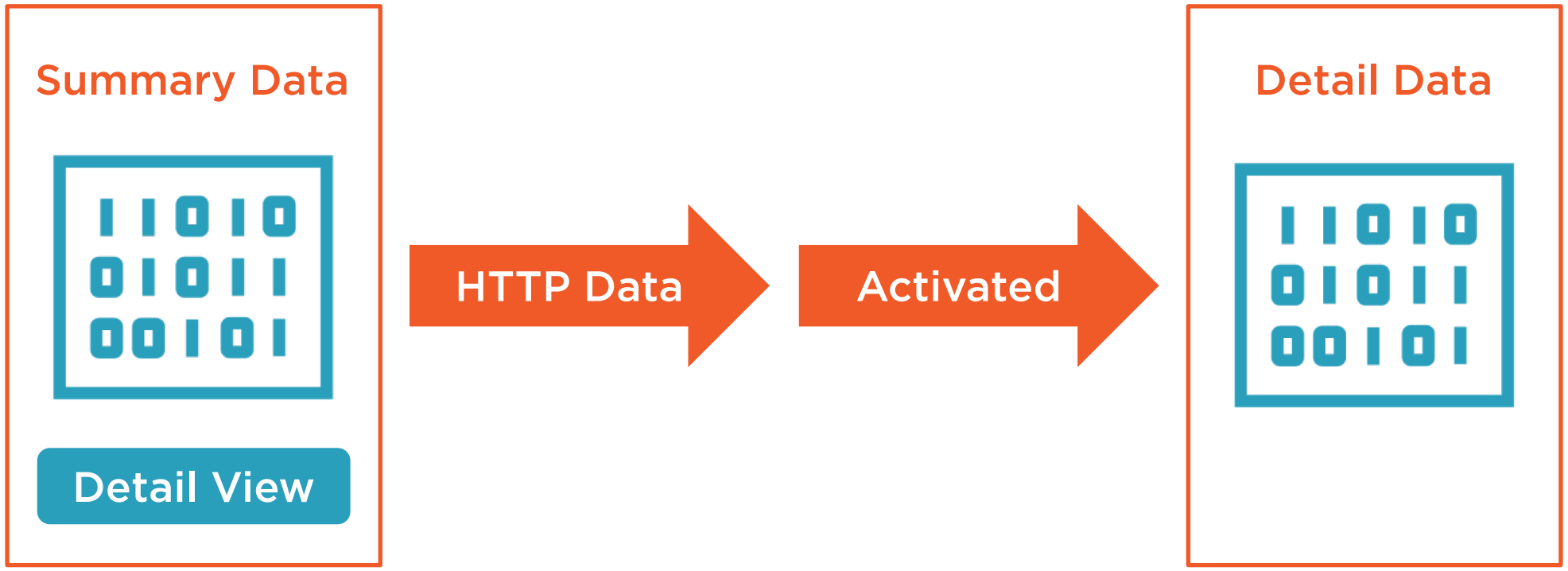


Detail View

Route Transitions With Resolvers



Route Transitions With Resolvers



Demo

Retrieving data over HTTP with Resolvers

Creating Interceptors

Brice Wilson

@brice_wilson www.BriceWilson.net

What Are Interceptors?

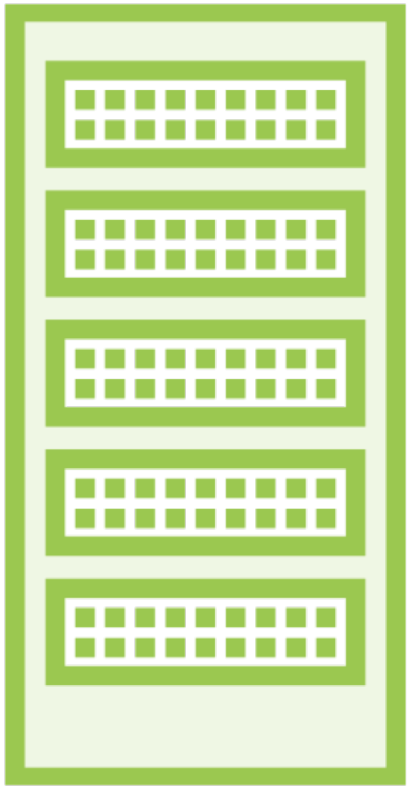
Services

Implement the HttpInterceptor interface

Manipulate HTTP requests before they're sent to the server

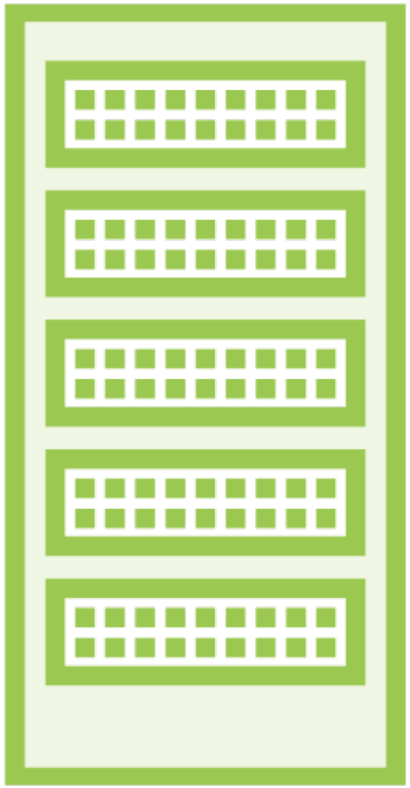
Manipulate HTTP responses before they're returned to your app

Intercepting Requests and Responses

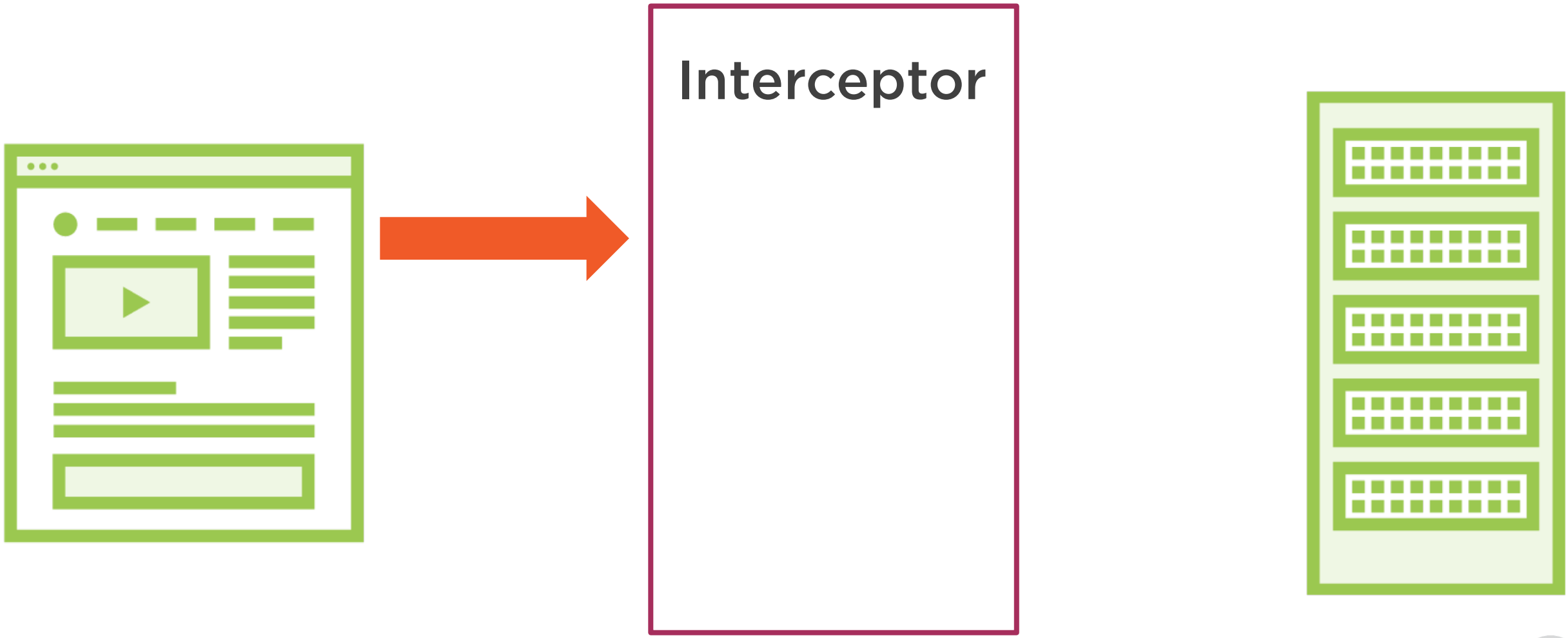


Intercepting Requests and Responses

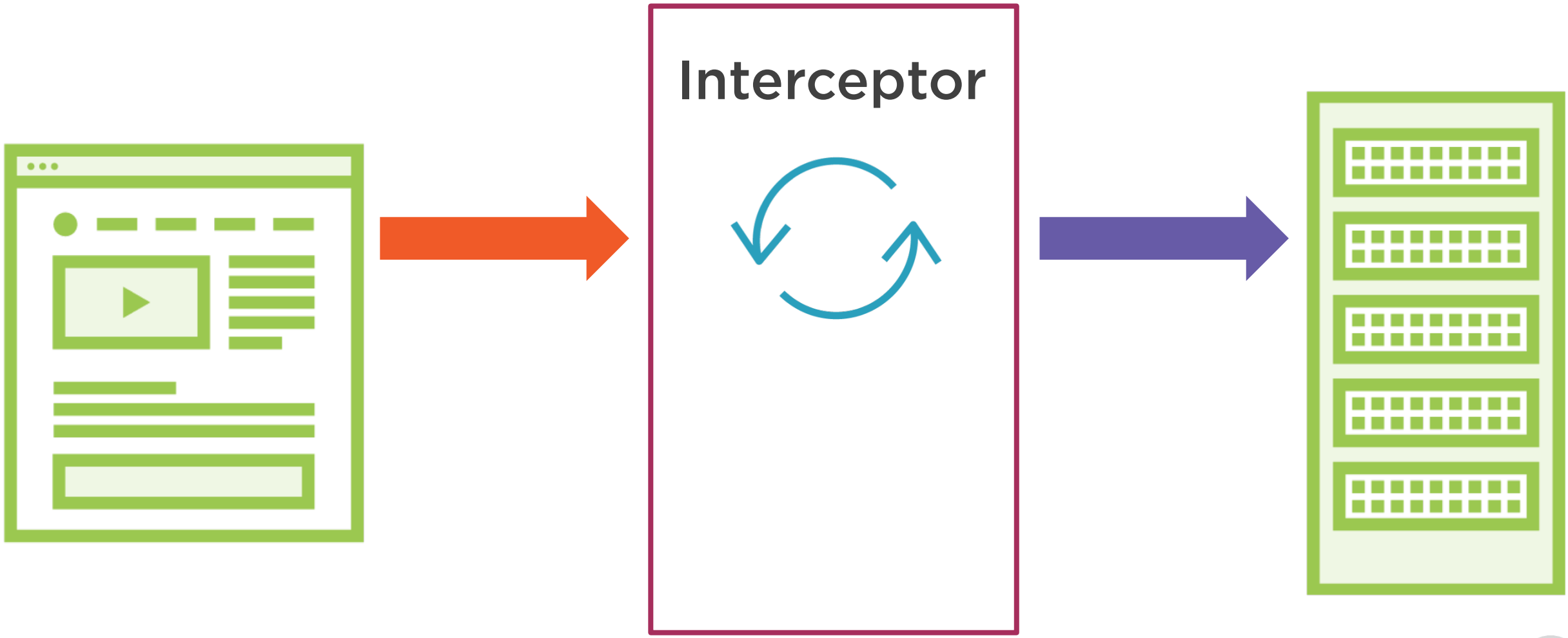
Interceptor



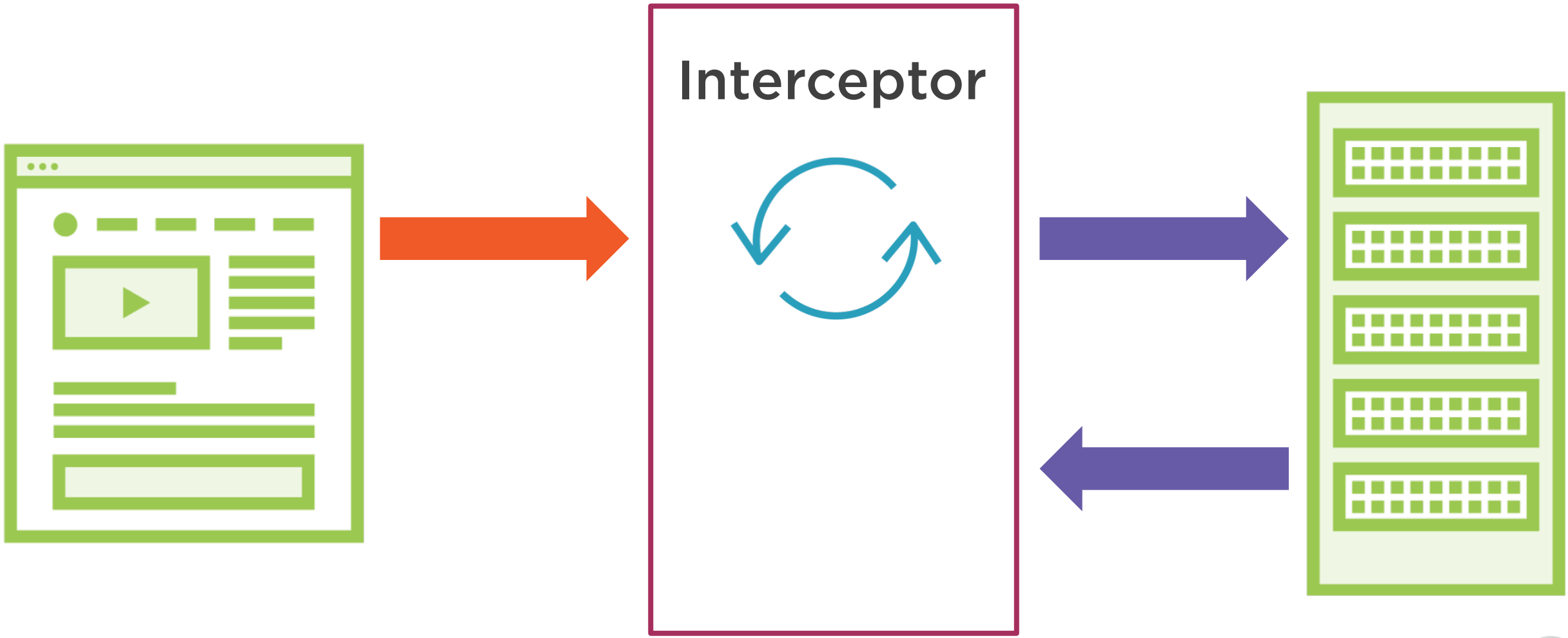
Intercepting Requests and Responses



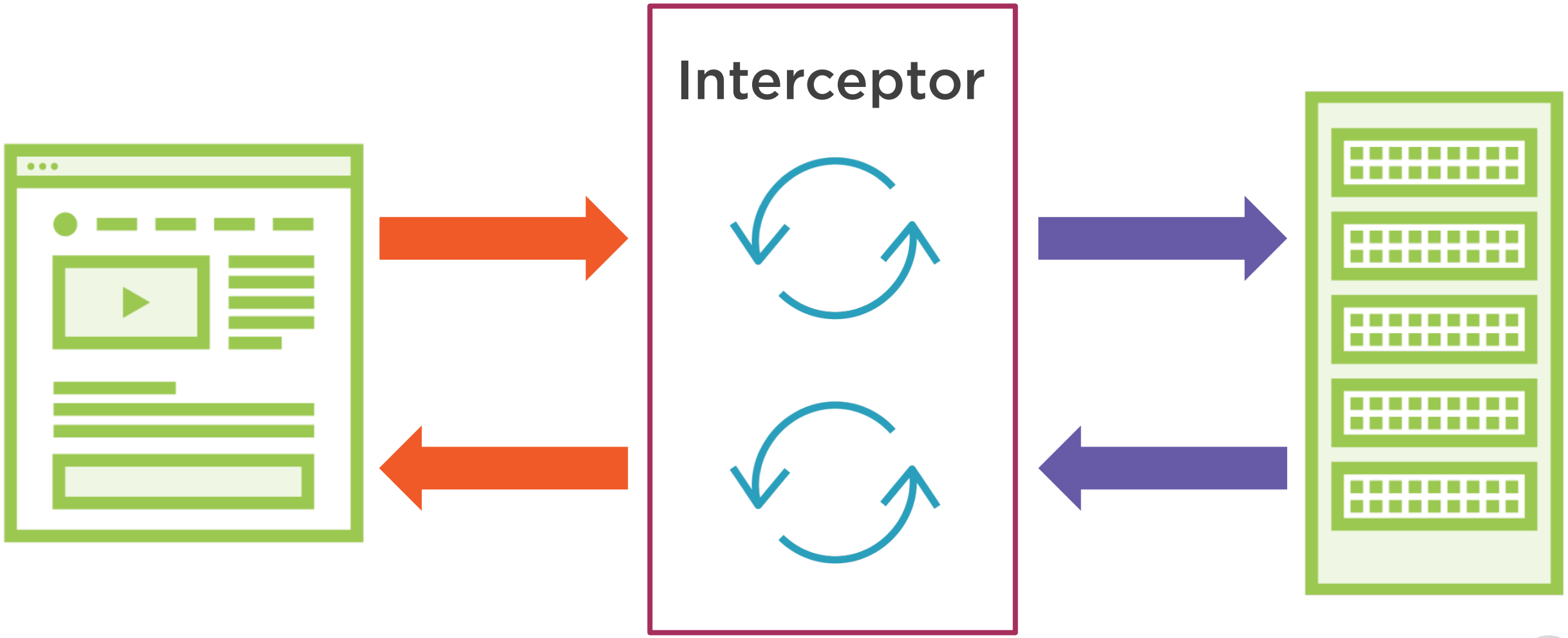
Intercepting Requests and Responses



Intercepting Requests and Responses



Intercepting Requests and Responses



Uses for Interceptors

Adding headers to all requests

Uses for Interceptors

Adding headers to all requests

Logging

Uses for Interceptors

Adding headers to all requests

Logging

Reporting progress events

Uses for Interceptors

Adding headers to all requests

Logging

Reporting progress events

Client-side caching

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {



}
```

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {




}
```

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const modifiedRequest = req.clone();




}
```

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const modifiedRequest = req.clone();




}
```

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const modifiedRequest = req.clone();
    // change modifiedRequest here




}
```

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const modifiedRequest = req.clone();
    // change modifiedRequest here
    return next.handle(modifiedRequest)



}
```

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const modifiedRequest = req.clone();
    // change modifiedRequest here
    return next.handle(modifiedRequest)
}
```

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const modifiedRequest = req.clone();
    // change modifiedRequest here
    return next.handle(modifiedRequest)



}
```

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const modifiedRequest = req.clone();
    // change modifiedRequest here
    return next.handle(modifiedRequest)



}
```

Defining an Interceptor

```
export class FirstInterceptor implements HttpInterceptor {
  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
    const modifiedRequest = req.clone();
    // change modifiedRequest here
    return next.handle(modifiedRequest)
      .pipe(
        .tap(event => {
          if (event instanceof HttpResponse) {
            // modify the HttpResponse here
          }
        })
      );
  }
}
```

Providing an Interceptor

```
@NgModule({
  imports: [],
  declarations: [],
  providers: [



  ]
})
export class AppModule { }
```

Providing an Interceptor

```
@NgModule({
  imports: [],
  declarations: [],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },


  ]
})
export class AppModule { }
```

Providing an Interceptor

```
@NgModule({
  imports: [],
  declarations: [],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
  ]
})
export class AppModule { }
```

Providing an Interceptor

```
@NgModule({
  imports: [],
  declarations: [],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
  ]
})
export class AppModule { }
```

Providing an Interceptor

```
@NgModule({
  imports: [],
  declarations: [],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
  ]
})
export class AppModule { }
```

Providing an Interceptor

```
@NgModule({
  imports: [],
  declarations: [],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
  ]
})
export class AppModule { }
```

Providing an Interceptor

```
@NgModule({
  imports: [],
  declarations: [],
  providers: [
    { provide: HTTP_INTERCEPTORS, useClass: FirstInterceptor, multi: true },
    { provide: HTTP_INTERCEPTORS, useClass: SecondInterceptor, multi: true },
  ]
})
export class AppModule { }
```

Demo

Creating an interceptor to add headers to all requests

Demo

Intercepting responses and working with multiple interceptors

Caching HTTP Requests with Interceptors

Brice Wilson

@brice_wilson www.BriceWilson.net

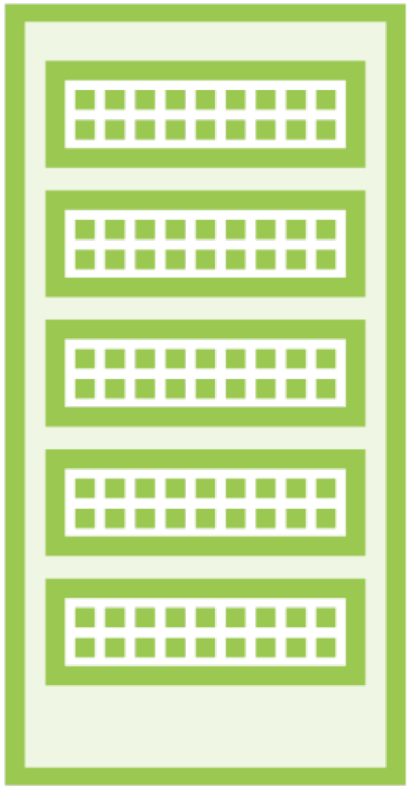
Benefits of Caching

Reduced network utilization

Reduced load on backend systems

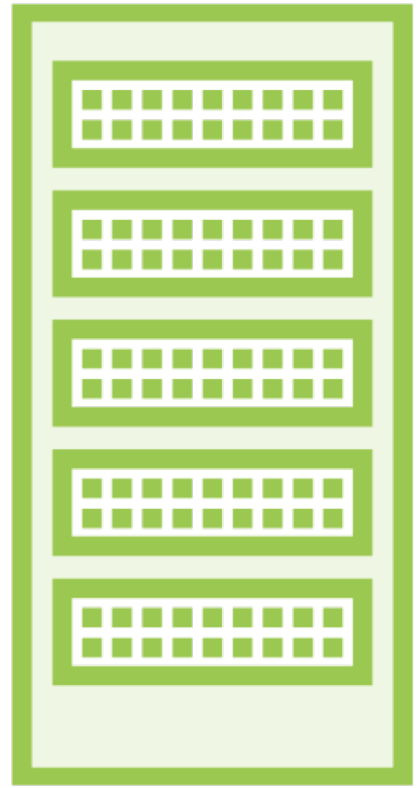
Improved responsiveness for the end user

Caching with Interceptors



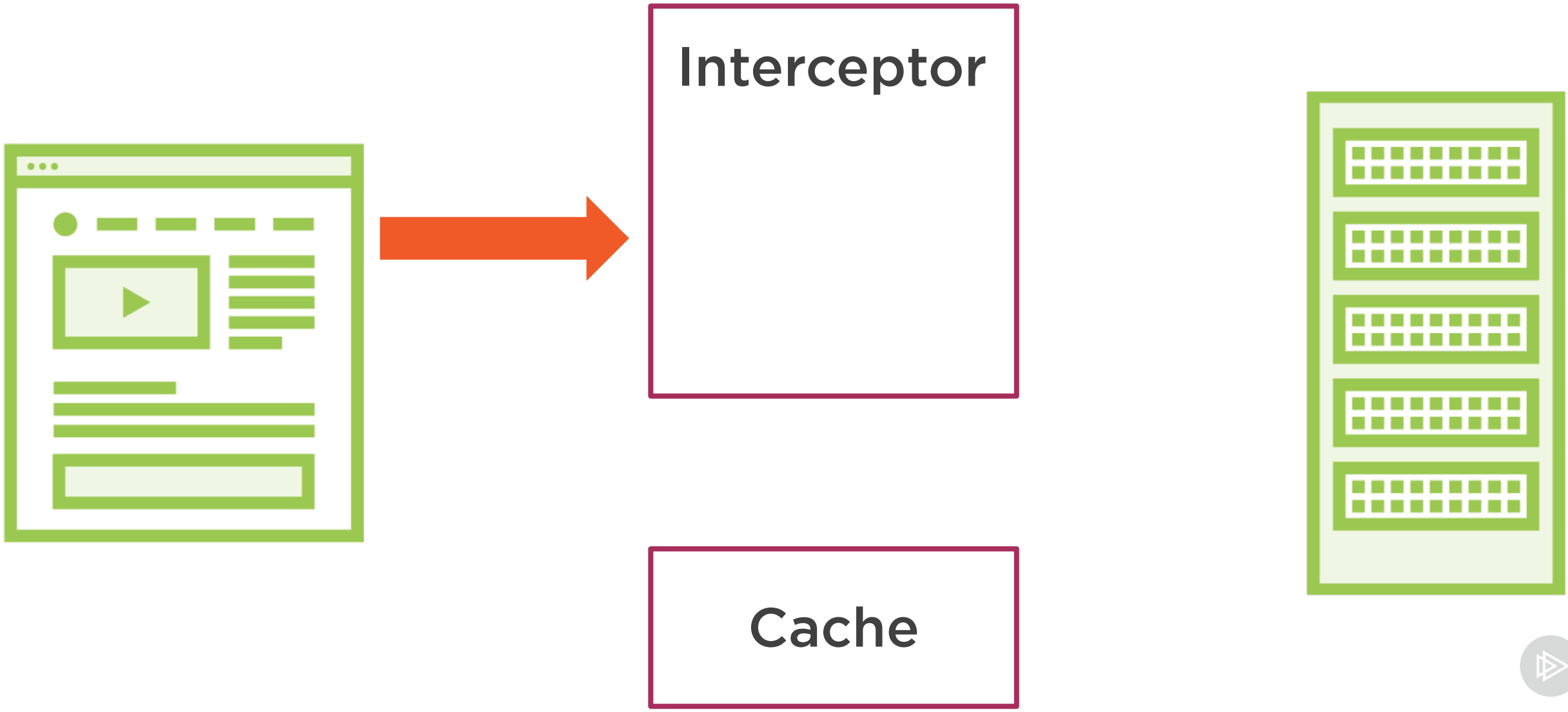
Caching with Interceptors

Interceptor

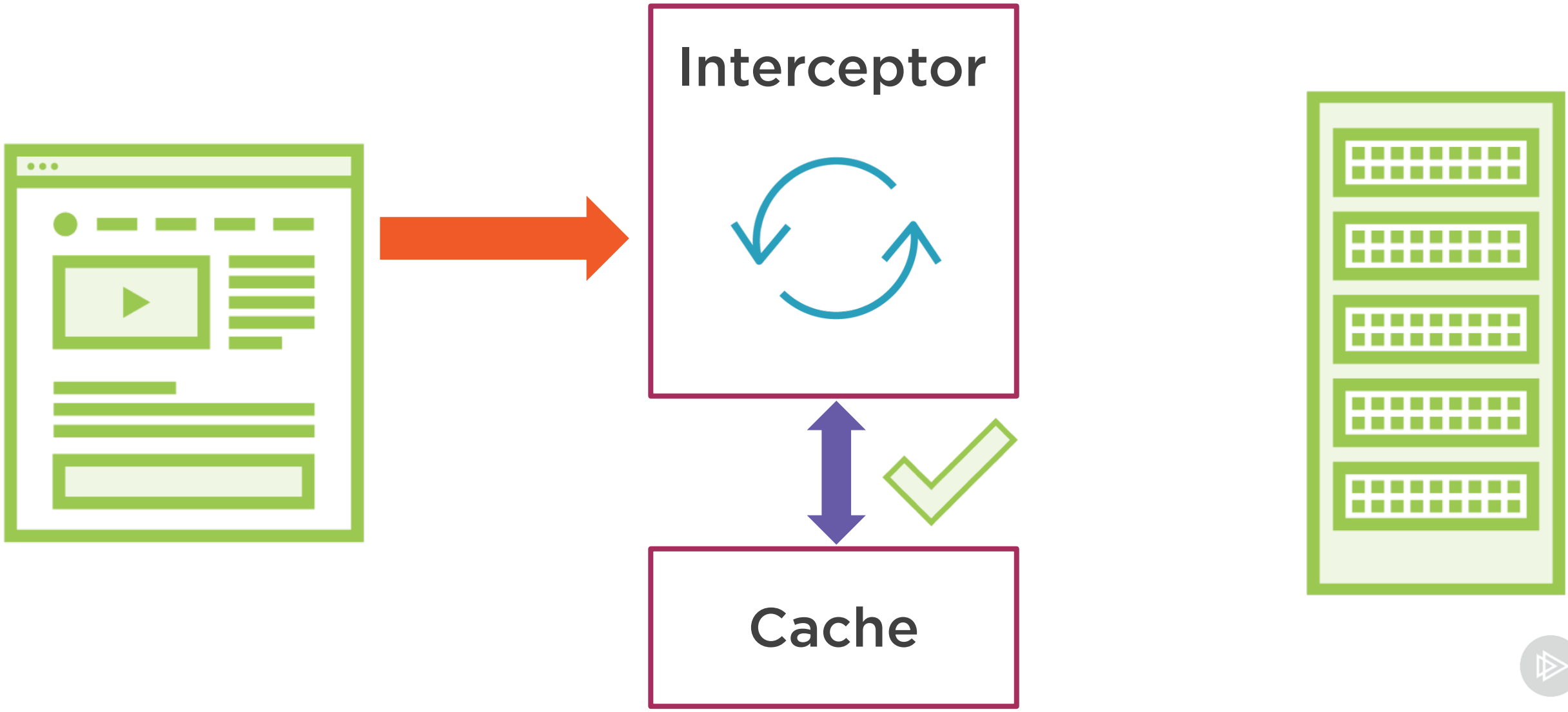


Cache

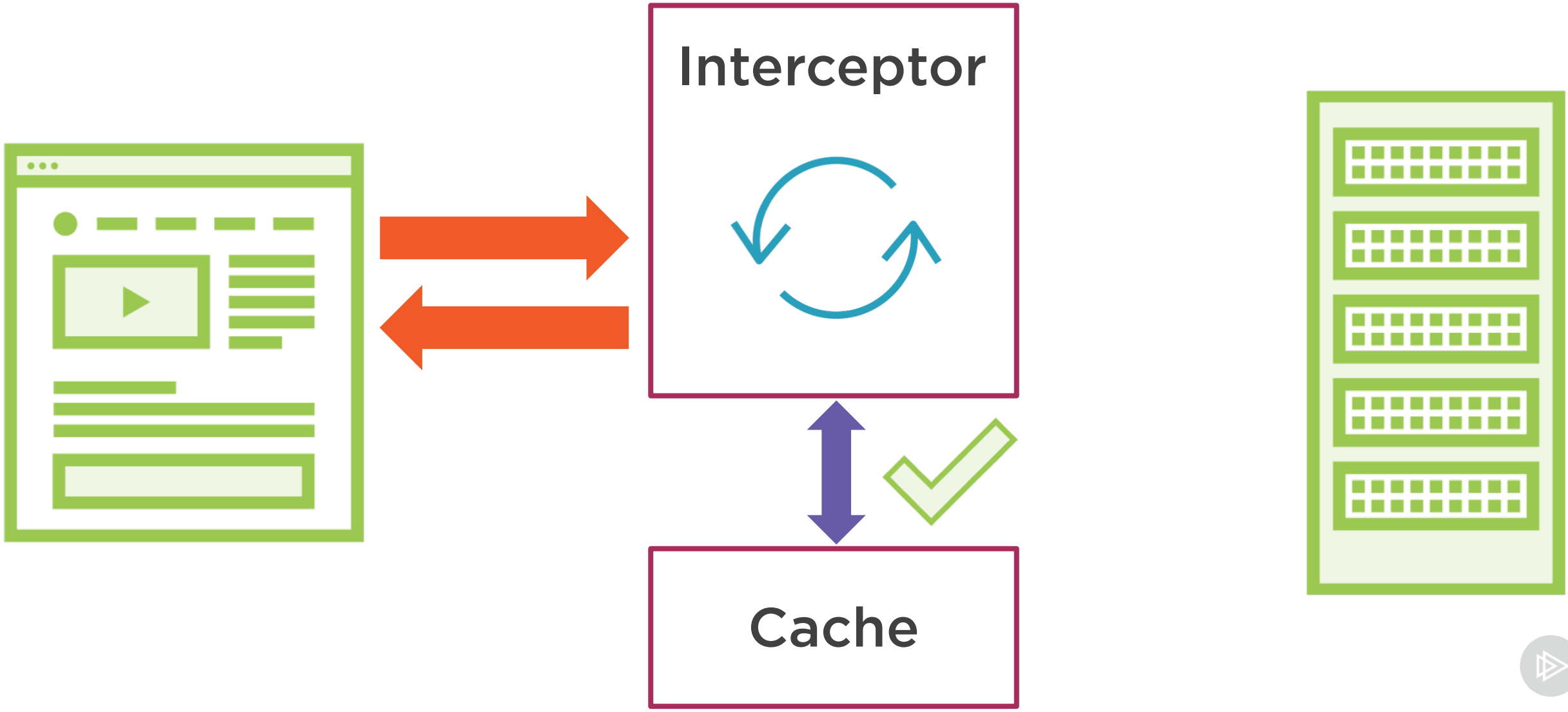
Caching with Interceptors



Caching with Interceptors

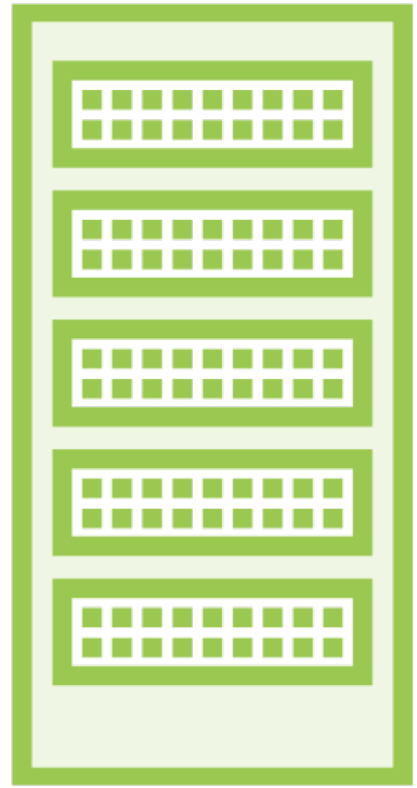


Caching with Interceptors



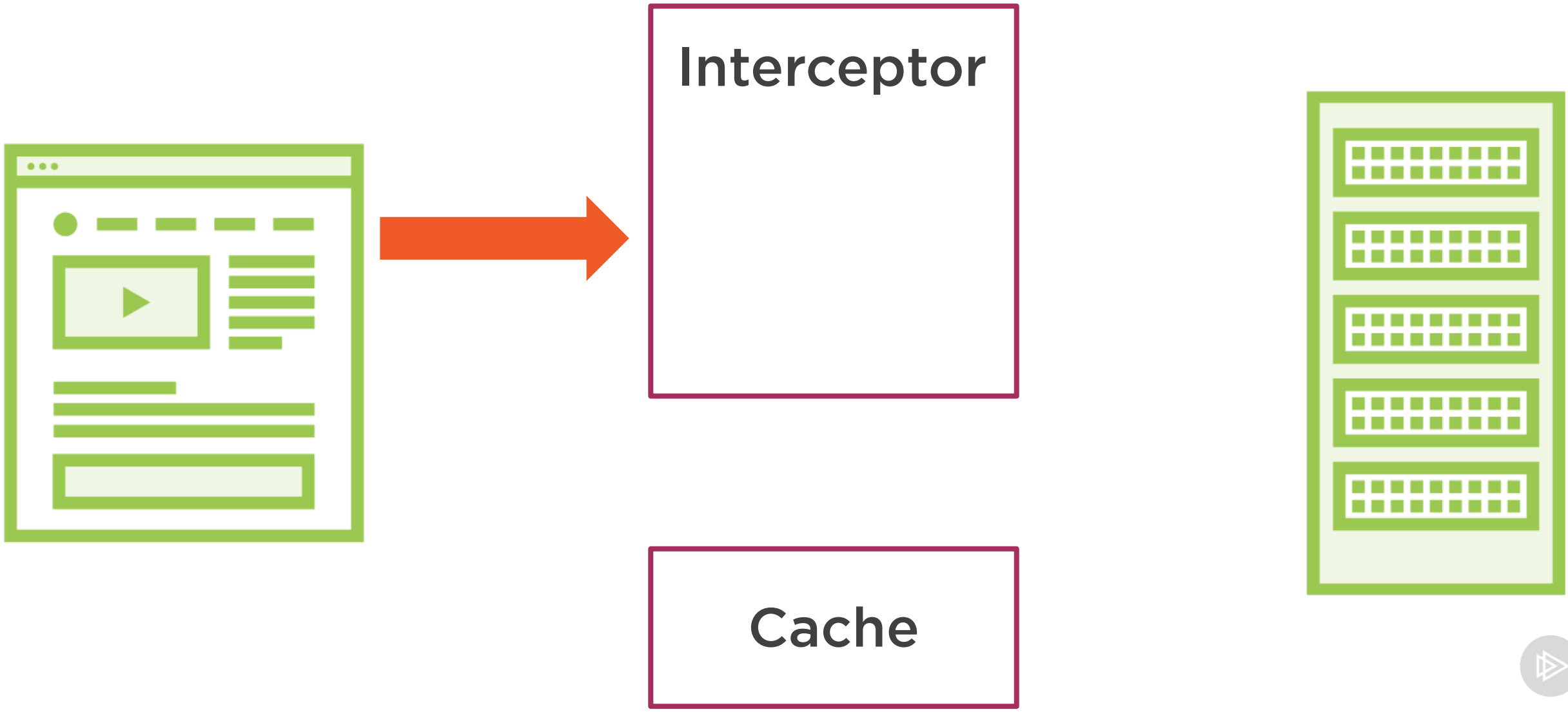
Caching with Interceptors

Interceptor

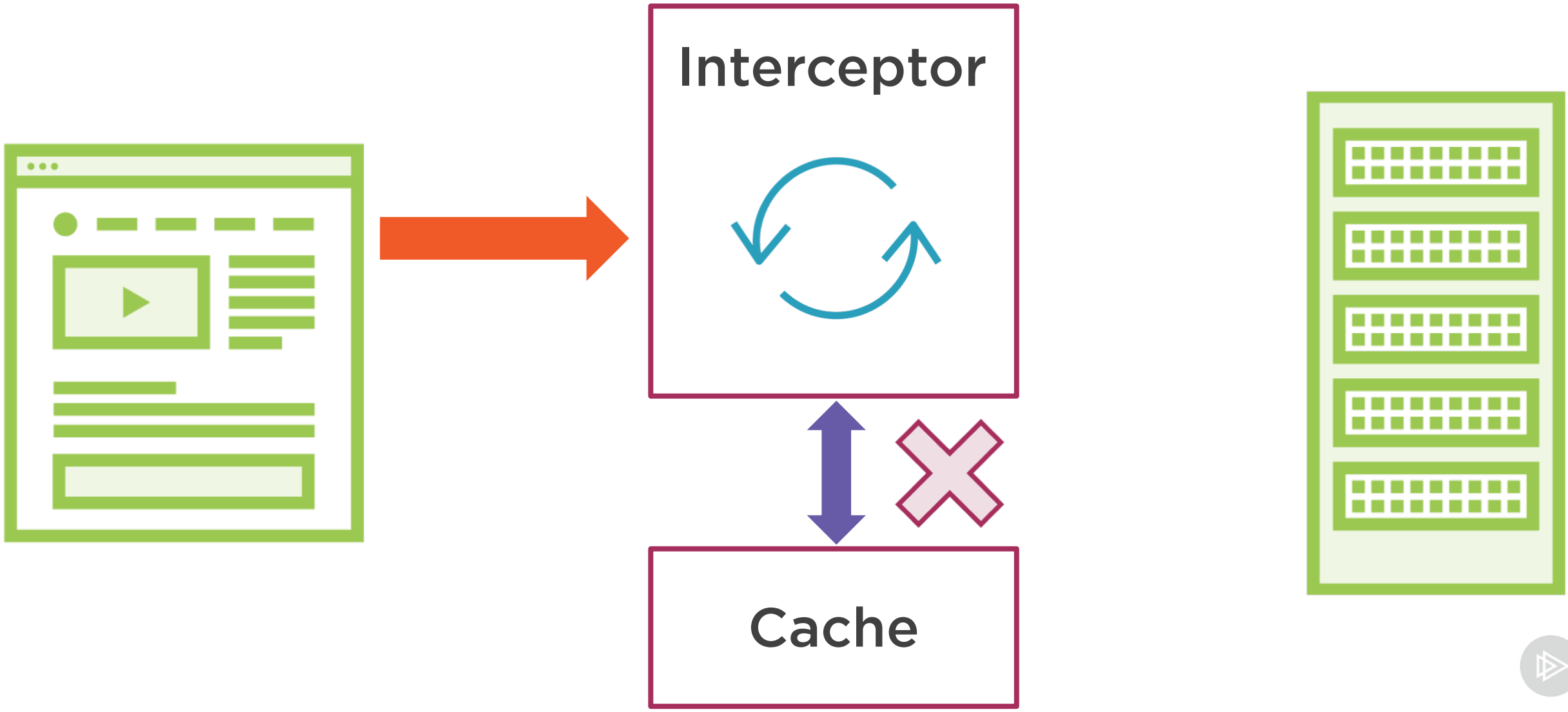


Cache

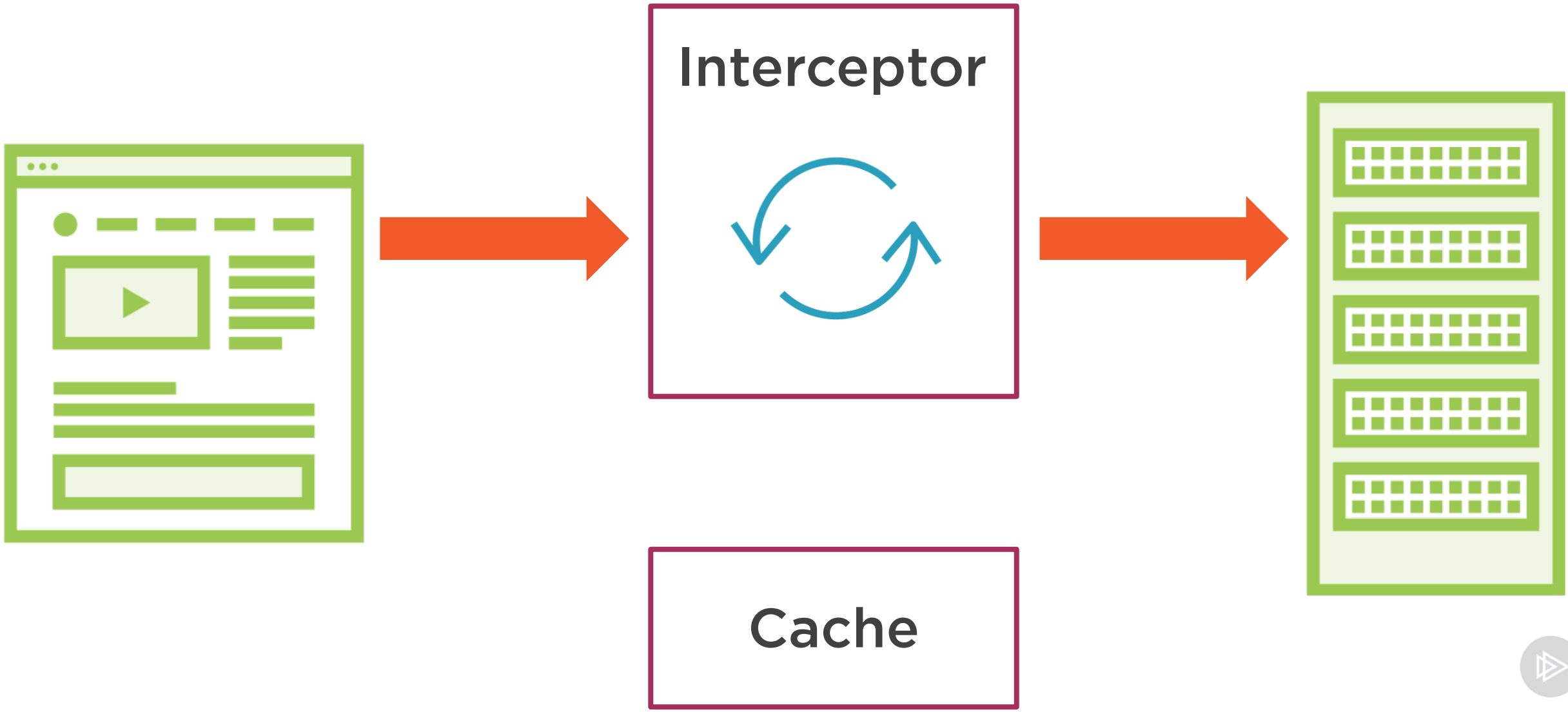
Caching with Interceptors



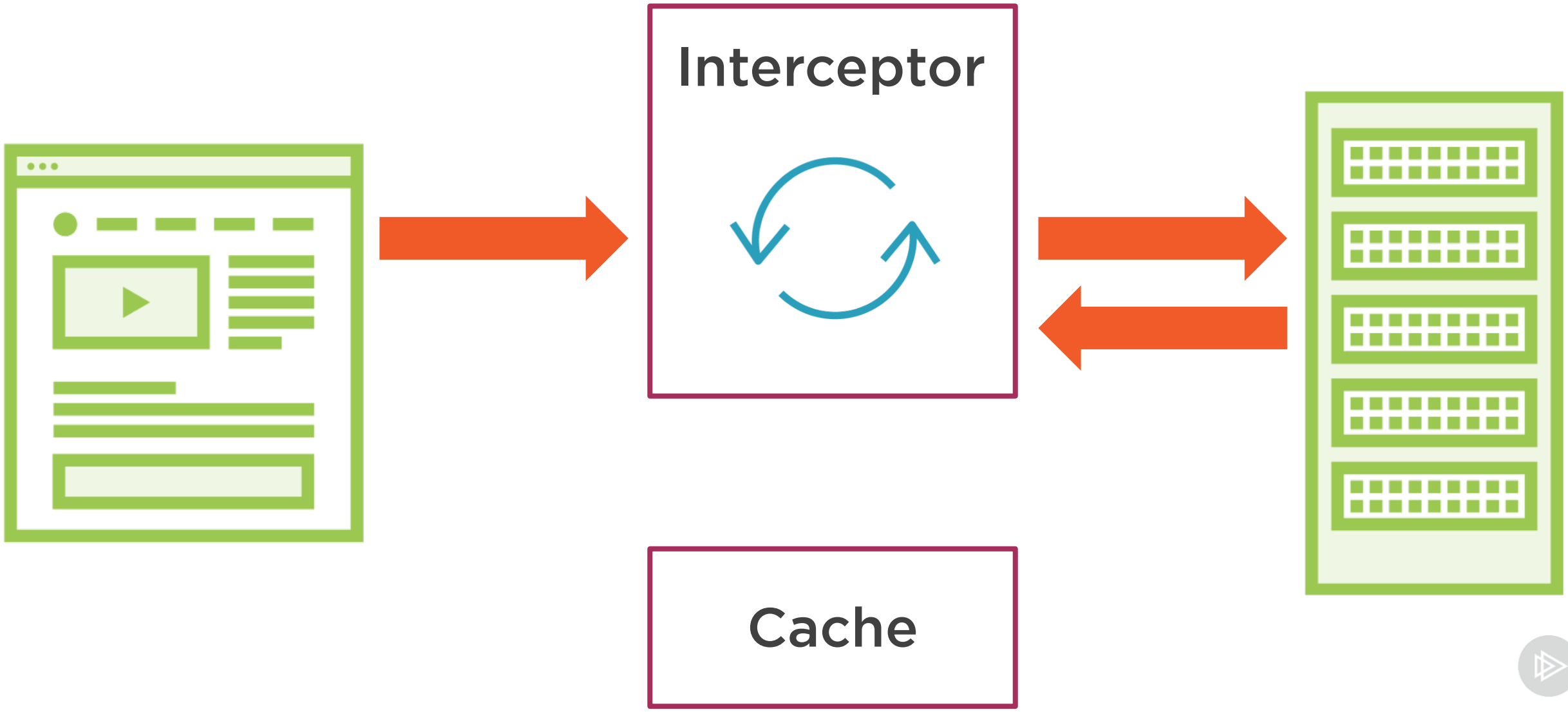
Caching with Interceptors



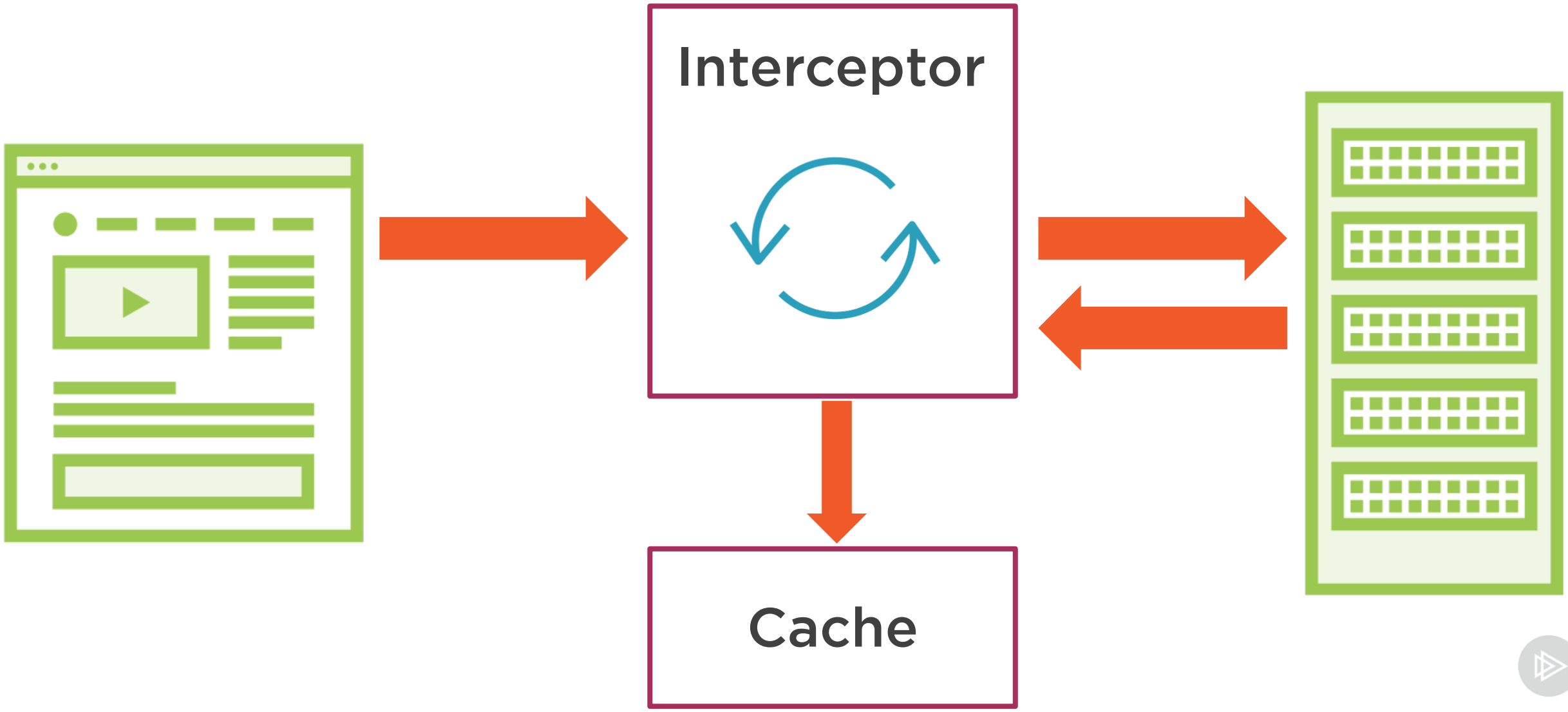
Caching with Interceptors



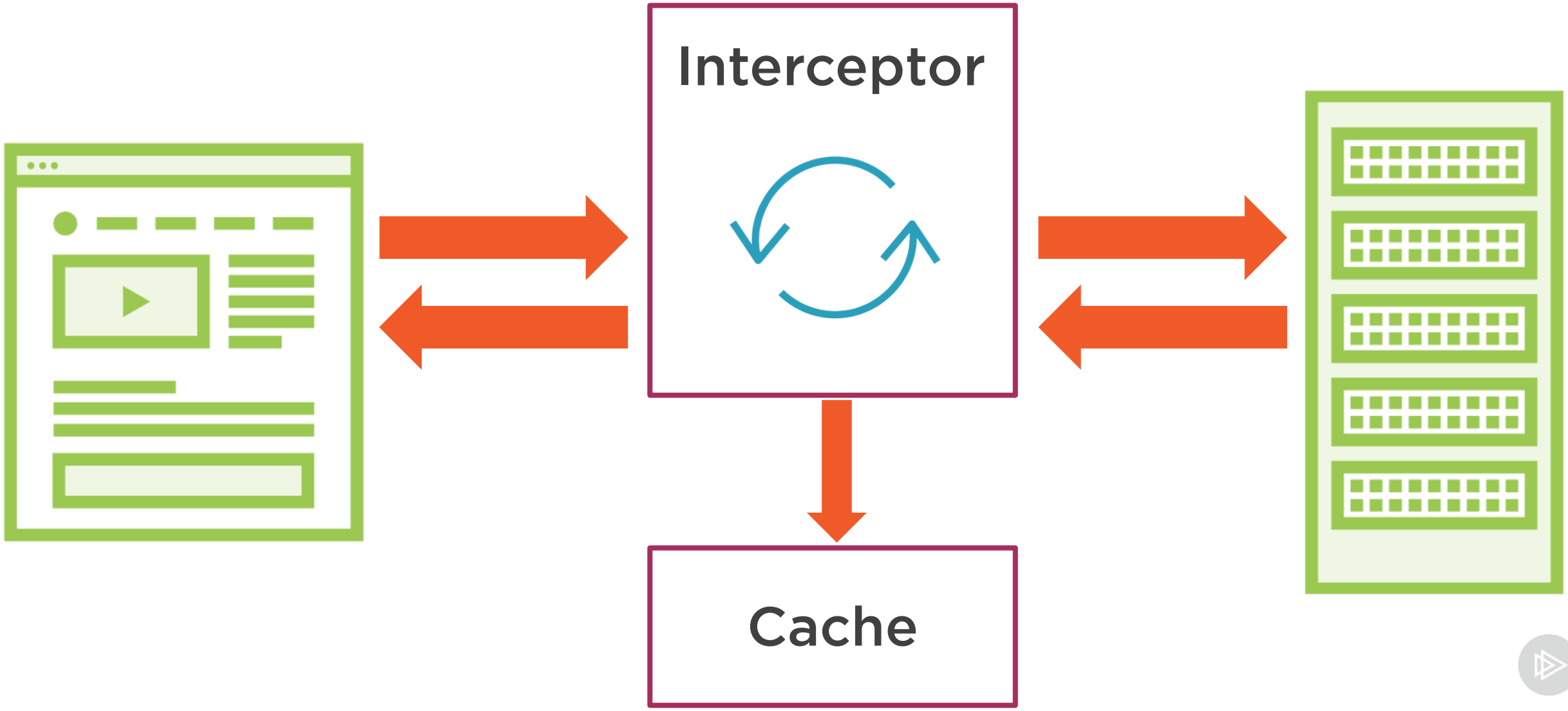
Caching with Interceptors



Caching with Interceptors



Caching with Interceptors



Role of the Cache Service

- Provide a data structure for the cached data
- Add items to the cache
- Retrieve items from the cache
- Remove items from the cache (cache invalidation)

Demo

Create a service to store cached HTTP requests

Demo

Create an interceptor to cache HTTP requests

Testing HTTP Requests

Brice Wilson

@brice_wilson www.BriceWilson.net

Default Angular Unit Testing Tools

- Jasmine
- Karma
- Angular testing utilities
- Angular CLI
- npm scripts

Structure of Jasmine Unit Tests

```
describe('DataService Test Suite', () => {



});
```

Structure of Jasmine Unit Tests

```
describe('DataService Test Suite', () => {
  beforeEach(() => {
    // setup code run before each test
  });




});
```

Structure of Jasmine Unit Tests

```
describe('DataService Test Suite', () => {
  beforeEach(() => {
    // setup code run before each test
  });
  it('should do the thing I expect', () => {
    // execute some code and test result
  });



});
```

Structure of Jasmine Unit Tests

```
describe('DataService Test Suite', () => {
  beforeEach(() => {
    // setup code run before each test
  });
  it('should do the thing I expect', () => {
    // execute some code and test result
  });
  afterEach(() => {
    // teardown code run after each test
  });
});
```

Angular HTTP Testing Utilities

HttpClientTestingModule

Angular HTTP Testing Utilities

HttpClientTestingModule

HttpTestingController

Structure of Angular HTTP Unit Tests

```
beforeEach(() => {




});
```

Structure of Angular HTTP Unit Tests

```
beforeEach(() => {
  TestBed.configureTestingModule({




});
```

Structure of Angular HTTP Unit Tests

```
beforeEach(() => {
  TestBed.configureTestingModule({
    imports: [ HttpClientTestingModule ],




});
```

Structure of Angular HTTP Unit Tests

```
beforeEach(() => {
  TestBed.configureTestingModule({
    imports: [ HttpClientTestingModule ],
    providers: [ DataService ]
  });


});
```

Structure of Angular HTTP Unit Tests

```
beforeEach(() => {
  TestBed.configureTestingModule({
    imports: [ HttpClientTestingModule ],
    providers: [ DataService ]
  });
  dataService = TestBed.get(DataService);
  httpTestingController = TestBed.get(HttpTestingController);
});
```

Structure of Angular HTTP Unit Tests

```
it('should return the correct book', () => {

});
```

Structure of Angular HTTP Unit Tests

```
it('should return the correct book', () => {
    dataService.getBookById(2)
      .subscribe(
        data => { /* test response here */ }
      );



 });
```

Structure of Angular HTTP Unit Tests

```
it('should return the correct book', () => {
    dataService.getBookById(2)
      .subscribe(
        data => { /* test response here */ }
      );
    let req = httpTestingController.expectOne('/api/books/2');



});
```

Structure of Angular HTTP Unit Tests

```
it('should return the correct book', () => {
    dataService.getBookById(2)
      .subscribe(
        data => { /* test response here */ }
      );
    let req = httpTestingController.expectOne('/api/books/2');
    // test request here



 });
```

Structure of Angular HTTP Unit Tests

```
it('should return the correct book', () => {
    dataService.getBookById(2)
      .subscribe(
        data => { /* test response here */ }
      );
    let req = httpTestingController.expectOne('/api/books/2');
    // test request here



});
```

Structure of Angular HTTP Unit Tests

```
it('should return the correct book', () => {
    dataService.getBookById(2)
      .subscribe(
        data => { /* test response here */ }
      );
    let req = httpTestingController.expectOne('/api/books/2');
    // test request here
    req.flush(<Book>{



 });
```

Structure of Angular HTTP Unit Tests

```
it('should return the correct book', () => {
    dataService.getBookById(2)
      .subscribe(
        data => { /* test response here */ }
      );
    let req = httpTestingController.expectOne('/api/books/2');
    // test request here
    req.flush(<Book>{



 });
```

Structure of Angular HTTP Unit Tests

```
it('should return the correct book', () => {
    dataService.getBookById(2)
      .subscribe(
        data => { /* test response here */ }
      );
    let req = httpTestingController.expectOne('/api/books/2');
    // test request here
    req.flush(<Book>{
      title: 'Winnie-the-Pooh',
      author: 'A. A. Milne'
    });
 });
```

Demo

Testing HTTP requests and responses

Demo

Testing HTTP errors