ARS

Motivation

- Funktionsfähigkeit von komplexem Code gewährleisten
  - Testausführung auf unterschiedlichen Granularitätsebenen
- Codeänderungen selbstbewusster vornehmen können
- Automatisch bei einem Deployment testen, dass das System sehr wahrscheinlich* noch so funktioniert wie zuvor

* Nichts im Leben ist 100% sicher!

ARS

Lernziele

- Sie stellen Aufwand und Nutzen des Testens auf unterschiedlichen Ebenen anhand der Testpyramide gegenüber.
- Sie benennen Vor- und Nachteile von Testautomatisierung gegenüber manuellen Tests
- Sie benennen wichtige Jasmine- und BDD-spezifische Konstrukte für die Definition von Testfällen.
- Sie beschreiben mit eigenen Worten den Unterscheid zwischen Jasmine und Karma.
- Sie erklären den Nutzen von Spies.

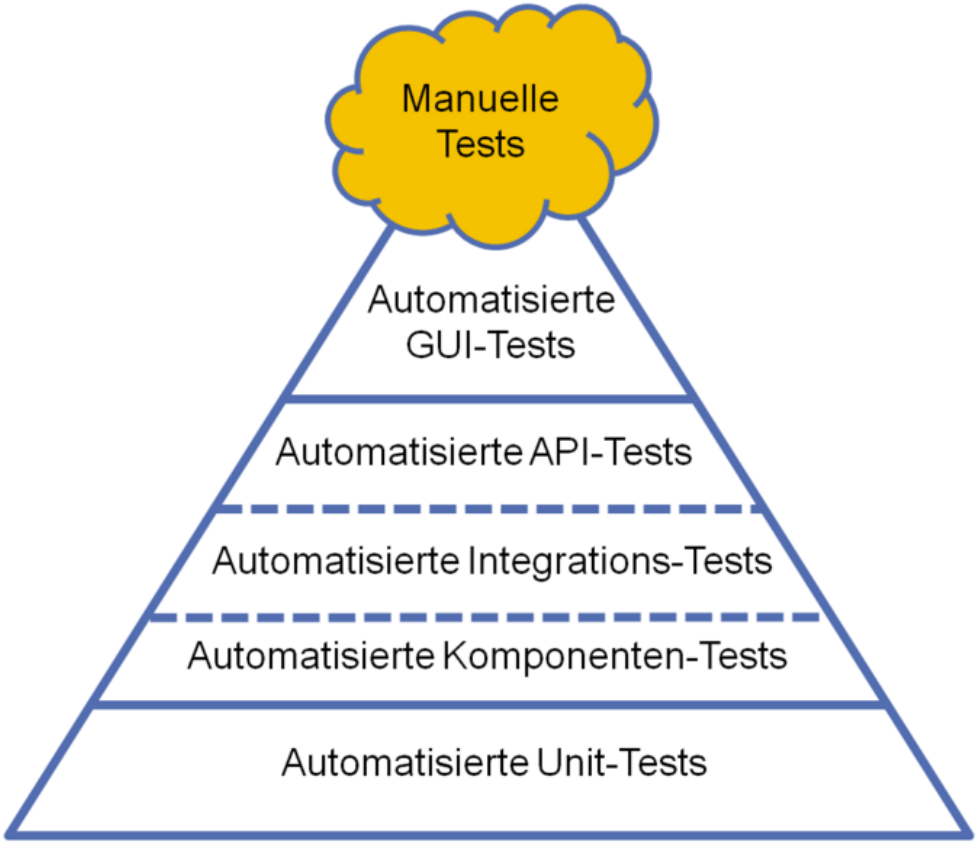
ARS

Testautomatisierung vs. Manuell Testen

Automatisiert testen	Manuell testen
Implementierung ggf. aufwändig, Ausführung schnell und beliebig oft wiederholbar.	Jede Testdurchführung ist gleich aufwändig. Deutlich langsamer als automatisierter Test.
Bei Bedarf schnell und einfach parallelisierbar	Personalbedarf steigt linear zum Grad der Parallelisierung
In jeder Schicht möglich, vom Unit-Test bis zum UI-Test.	Nur „Black-Box“-Tests sind möglich, also Oberfläche und ggf. REST-Schnittstelle.
Jede Testausführung läuft exakt so ab wie die zuvor.	Mensch macht Fehler, kann dafür aber auch „explorativ“ testen.
Testergebnis ist maschinell verarbeitbar (Jenkins, Continuous Integration)	Testdokumentation und Ergebnisverarbeitung aufwändig.

ARS

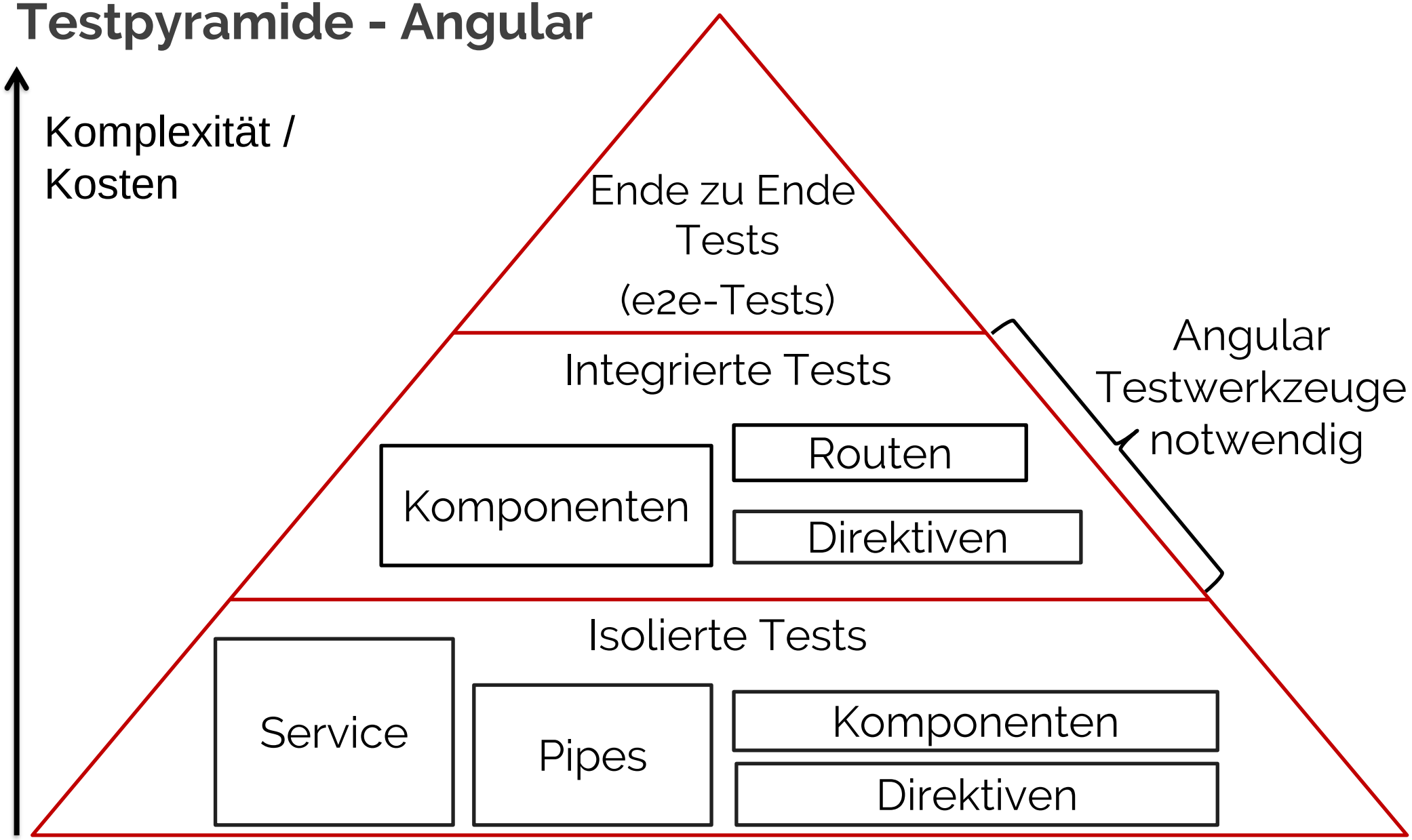
Testpyramide



http://www.informatik-aktuell.de/entwicklung/methoden/tests-erst-in-produktion-was-wir-von-tests-bei-microservices-lernen-koennen.html

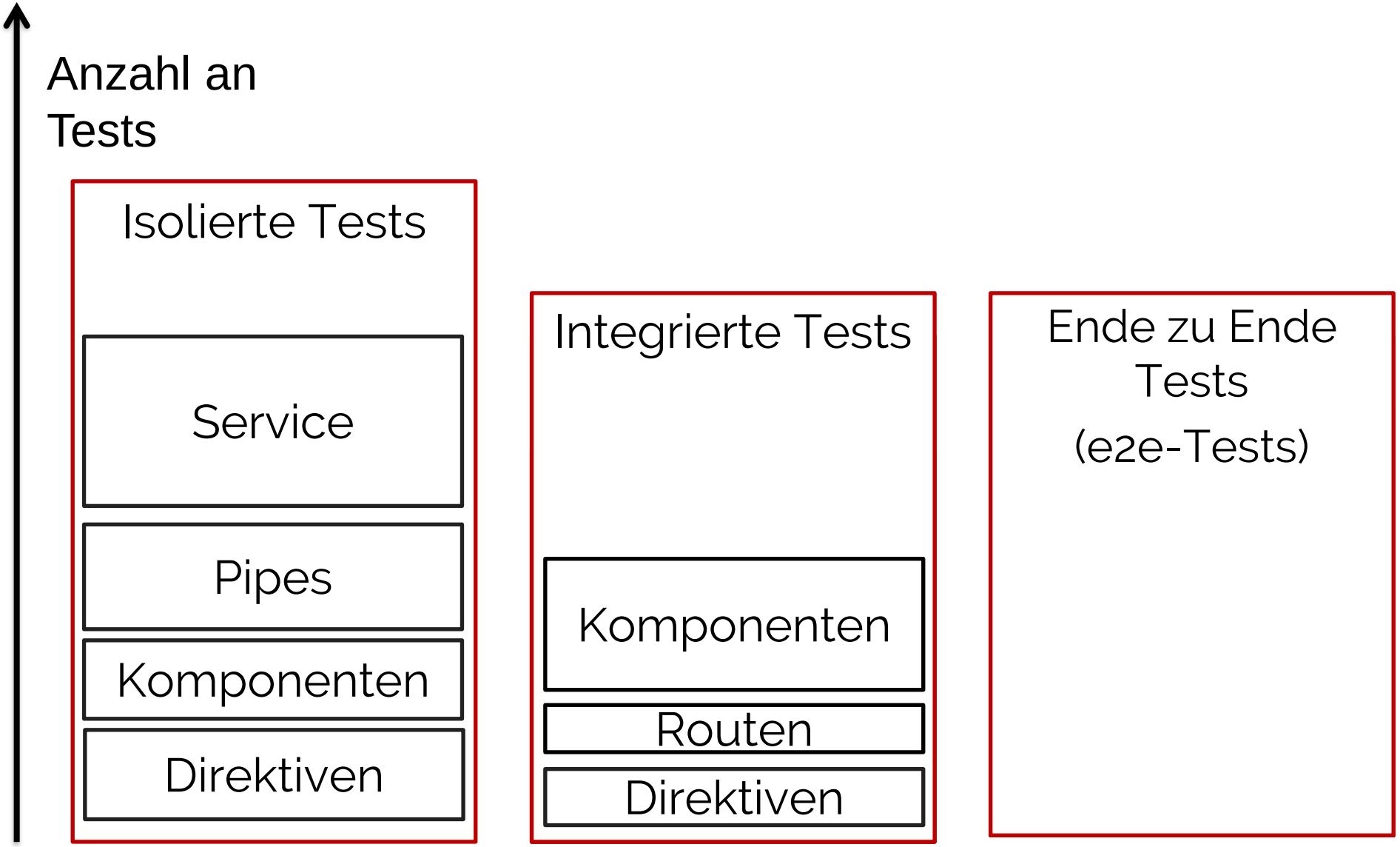
ARS

Testpyramide - Angular



ARS

Einführung



ARS

Überblick

- Tests sollen die Funktionsfähigkeit einer Code-Einheit sicherstellen
- Jasmine = Framework für u. a. Behavior Driven Development (BDD)
  - Dokumentierte Ziele und Ergebnisse für Features in Textform (z. B. Use Cases)
  - Erstellung von automatisierten Tests, die dieses überprüfen
- Vorteile von Jasmine
  - Offizieller, unterstützter Weg des Angular-2-Teams
  - Assertion-Bibliothek
  - Mock-Bibliothek
- Testrunner: Karma
  - Testframework = Stellt Methoden für den Aufbau und die Definition von Testfällen bereit
  - Testrunner = Führt definierte Tests eines Testframeworks aus

ARS

Jasmine

- Framework für u. a. Behavior Driven Development (BDD)
  - Dokumentierte Ziele und Ergebnisse für Features in Textform (z. B. Use Cases)
  - Erstellung von automatisierten Tests, die dieses überprüfen
- Vorteile
  - Offizieller, unterstützter Weg des Angular-2-Teams
  - Assertion-Bibliothek
  - Mock-Bibliothek

ARS

Karma

- Test-Runner von Angular / Angular-CLI
  - Führt definierte Tests eines Testframeworks aus
  - Kann unterschiedliche Test-Frameworks verwenden (Jasmine, Mocha, …)

Jasmine = Testbeschreibung

Karma = Testausführung

Starten von Karma durch die Angular-CLI:

```
> ng test
```

ARS

Anatomie eines Jasmine-Tests

- describe - Strukturierung von Testfällen
- beforeEach
  - Wird vor jedem Test ausgeführt
  - Vorbereitung der Umgebung für einen Test
- afterEach
  - Wird nach jedem Test ausgeführt
  - Aufräumen der Umgebung nach einem Test
- it – definiert einen neuen Testfall

```
describe('HelloWorld', () => {
  let helloWorld;

  beforeEach(() => {
    helloWorld = 'Hello World!';
  });

  afterEach(() => {
    helloWorld = null;
  });

  it('should initialize variable
      helloWorld', () => {
      …
  });
});
```

ARS

Überprüfung von Testbedingungen

- expect –zu überprüfender Wert wird übergeben
- Verschiedene Matcher möglich
  - toBeDefined – überprüft ob der Wert nicht null oder undefined ist
  - toEqual – Vergleicht den gegebenen Wert mit einem erwarteten Wert
  - toBeThruthy – Schaut ob der Wert „true“ ist

```
describe('HelloWorld', () => {
  const helloWorld = 'Hello World!';

  it('should initialize helloWorld ',
    () => {
      expect(helloWorld).toBeDefined();
      expect(helloWorld)
            .toEqual('Hello World!');
    });
});
```

- Weitere Matcher unter: https://tinyurl.com/gs-jasmine-matcher
- Definition von Testbedingung soll soweit wir möglich komplette Sätze formen: „expect variable to equal ‚Hello World‘ “

ARS

Tests für asynchrone Werte

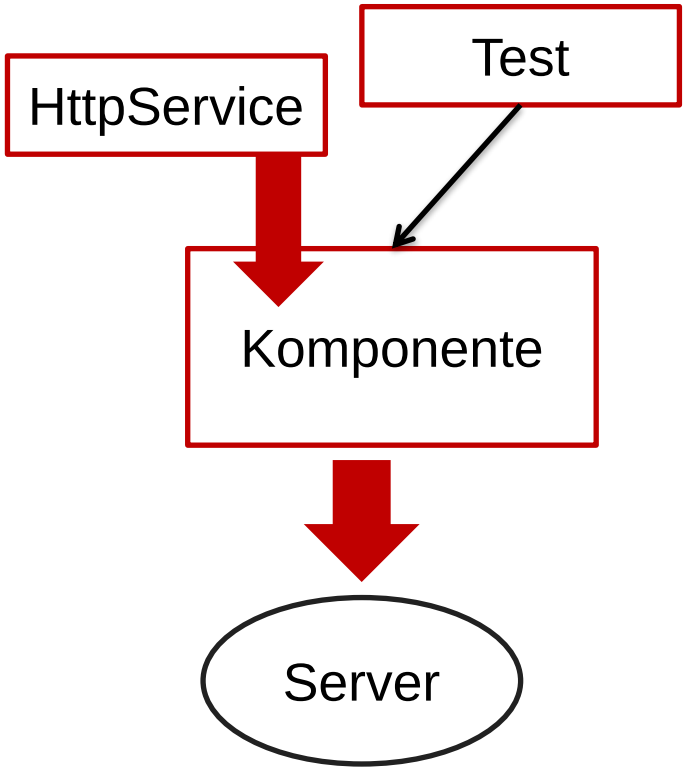
- Jasmine muss auf asynchrone Ausführung warten
- Signalisierung von Fertigstellung über done-Callback
- **Best Practice:** Immer als letztes „then“ in einer Promise-Kette
- **Vorsicht:** Timeouts bei längeren Wartezeiten möglich

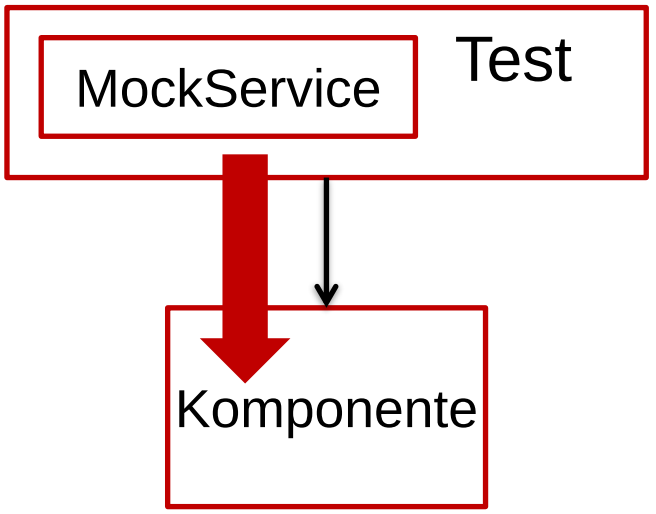
```
describe('HeroService', () => {
  // Initialisierung von HeroService
  let heroService: HeroService;

  it('should return three heroes from new york', (done) => {
    const possibleHeroes = heroService.getByCity('New York');
    possibleHeroes
      .then(heroesFromNY => expect(heroesFromNY.length).toBe(3))
      .then(done);
  });
});
```

ARS

Testen ohne Nebeneffekte





ARS

Spies

- Vergleichbar mit Doppelgänger → Fangen Funktionsaufrufe ab
- Ermöglichen
  - Verfolgung von Funktionsaufrufen
  - Informationen über Übergabeparameter
  - Rückgabewerte überschreiben
  - Mocks und Stubs
- Globale Funktion: spyOn(…)
  - Betroffenes Objekt
  - Name der Methode (für Mocks / Stubs)

- https://tinyurl.com/gs-jasmine-spies

ARS

Spies

```
describe('HeroService', () => {
  let service: HeroService;
  let getMethodSpy: Spy;
  beforeEach(() => {
    // Davor http erzeugen
    getMethodSpy = spyOn(http, 'get');
    getMethodSpy.and.returnValue([{ name: 'Thunderbolt' }]);
    service = new HeroServ
  });
  it('should create url wi
    service.getHeroes();
    const url = getMethodS
    expect(getSpy).toHaveBeenCalled();
    expect(url).toContain('/heroes');
  });
});
```

```
getHeroes() {
  return this.http.get('api/heroes')
}
```

ARS

Zusammenfassung

Um was ging es in diesem Modul?

- Verschiedene Arten automatisierter Tests
- Unit Tests mit Jasmine Framework für Angular Bausteine
- Testen von asynchroner Funktionalität
- Nutzung von Spionen in Unit Tests

Wozu brauche ich das? Was will ich damit machen?

- Sicherstellen dass Methoden das tun, was sie tun sollen
- Beim Refactoring sicherstellen dass sich Funktionalität nicht geändert hat
- Automatisch bei einem Deployment testen, dass das System sehr wahrscheinlich noch so funktioniert wie zuvor

ARS

Kontrollfragen

- Welche Bedeutung haben die Schichten der Testpyramide?
- Welche Vorteile bringt automatisiertes Testen gegenüber manuellen Tests?
- Wozu benutzt man Spione?
- Was ist der Unterschied zwischen Jasmine und Karma?
- Mit Hilfe von welcher Syntax werden Tests in Jasmine definiert?
  Erläutern Sie kurz ihre Bedeutung.