ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Angular Testing

3 - Unit Tests Advanced

Parameterisable Tests

```
it.each([
  ['Veni vidi vici', 3],
  ['Lorem ipsum', 2],
  ['The brown 🦊 jumped over the lazy 🐶', 8],
  ['Some  space ', 2]
])('%s should have %d words', (sentence, wordcount) => {
  expect(sentence.split(' ').filter((word) => word)).toHaveLength(wordcount);
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Testing Goodies

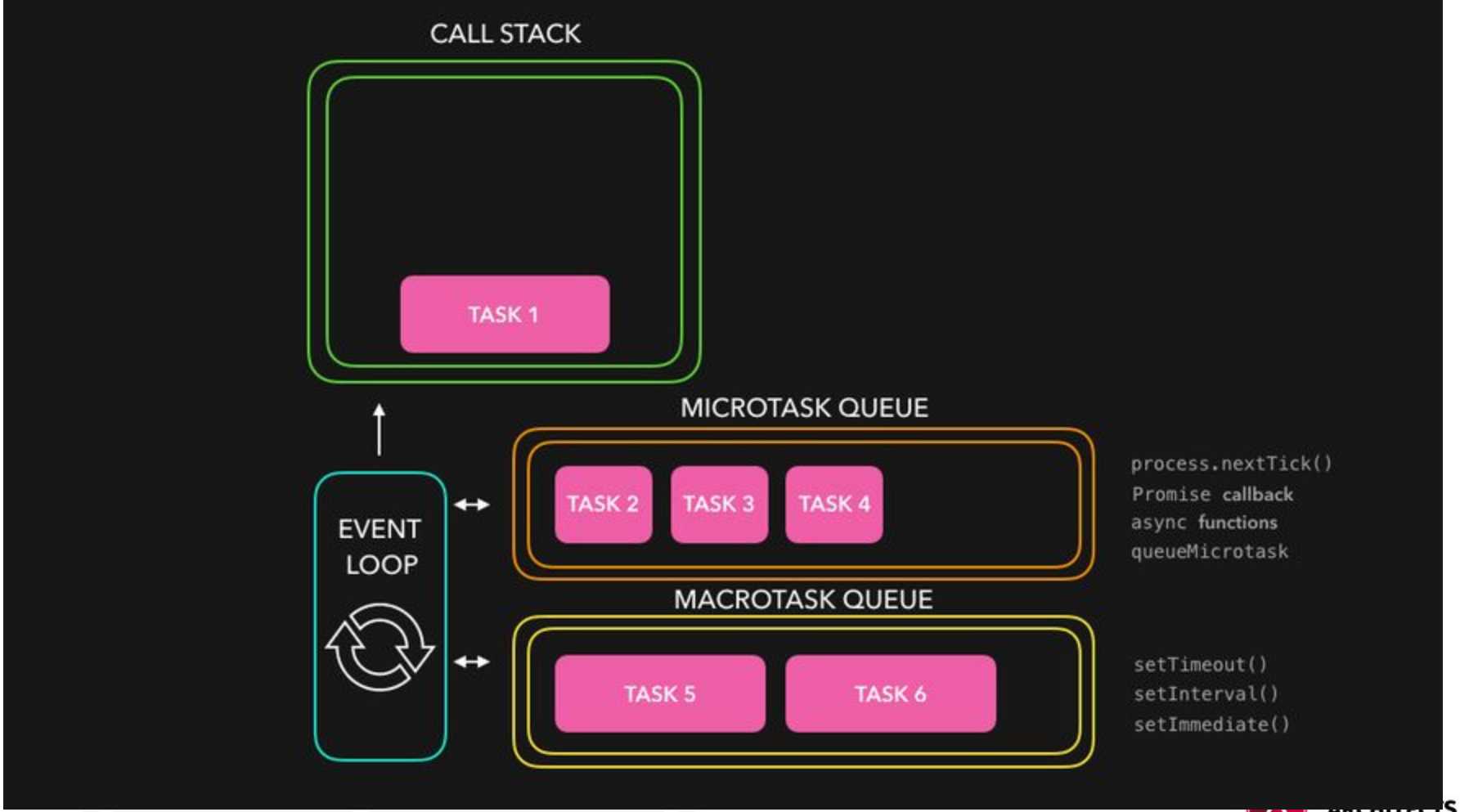
- it.skip
- it.only
- it.todo

Asynchronity

```
if places.count > 0 {

    for i in 0..<places.count {
        for j in 0..<places.count {
            if let nameI = places[i]["name"] {
                if let cityI = places[i]["city"] {
                    if let nameJ = places[j]["name"] {
                        if let cityJ = places[j]["city"] {
                            if let latI = places[i]["lat"] {
                                if let lonI =  places[i]["lon"] {
                                    if let latitudeI = Double(latI) {
                                        if let longitudeI = Double(lonI) {
                                            if let latJ = places[j]["lat"] {
                                                if let lonJ =  places[j]["lon"] {
                                                    if let latitudeJ = Double(latJ) {
                                                        if let longitudeJ = Double(lonJ) {

                                                            if(i != j) {

                                                                let coordinateI = CLLocation(latitude: latitudeI, longitude: longitudeI)
                                                                let coordinateJ = CLLocation(latitude: latitudeJ, longitude: longitudeJ)

                                                                let distanceInMeters = coordinateI.distance(from: coordinateJ) // result is in meters
                                                                let distanceInMiles = distanceInMeters/1609.344


                                                                var distances = [Distance]()
                                                                distances.append(Distance(
                                                                    distanceInMiles: distanceInMiles,
                                                                    distanceInMeters: distanceInMeters,
                                                                    places: [
                                                                        Place(name: nameI, city: cityI, lat: latitudeI, long: longitudeI, coordinate: coordinateI),
                                                                        Place(name: nameJ, city: cityJ, lat: latitudeJ, long: longitudeJ, coordinate: coordinateJ),
                                                                        ]
                                                                ))
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

https://levelup.gitconnected.com/escape-the-pyramid-of-doom-c58edd326225

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE



https://dev.to/lydiahallie/javascript-visualized-promises-async-await-5gke

Potential Problems

- Expects not running
- Timeouts
- Cryptic error messages

Testing Asynchronity

- MacroTasks do not run
  - setInterval
  - setTimeout
- MicroTasks run too late
  - Promise
  - async/await

expect.hasAssertions

```
it("should test with hasAssertion", () => {

  expect.hasAssertions();

  let a = 1;

  Promise.resolve().then(() => {

    return a++;

    expect(a).toBe(2);

  });

});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

done

```
it("should test with done", (done) => {
  let a = 1;
  Promise.resolve()
    .then(() => {
      a++;
      expect(a).toBe(1);
    })
    .then(done, done);
});
```

return the Promise

```
it("should return the promise", () => {

  let a = 1;

  return Promise.resolve().then(() => {

    a++;

    expect(a).toBe(2);

  });

});
```

expect().resolves

```
it("should test with expect.resolves", () => {

  let a = 1;

  const promise = Promise.resolve().then(() => a + 1);

  return expect(promise).resolves.toBe(2);

});
```

Use async/await

```
it("should test with done", async () => {

  let a = 1;

  await Promise.resolve().then(() => {

    a++;

  });

  expect(a).toBe(2);

});
```

waitForAsync

```
test('async', waitForAsync(() => {
    expect.hasAssertions();
    let a = 1;
    Promise.resolve().then(() => {
      a++;
      expect(a).toBe(2);
    });

    window.setTimeout(() => {
      a++;
      expect(a).toBe(3);
    }, 1000);
  })
);
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Bending Time

fakeAsync / useFakeTimers

```
test("microtasks", fakeAsync(() => {
  let a = 1;
  Promise.resolve().then(() => (a = 2));
  expect(a).toBe(1);

  flushMicrotasks();
  expect(a).toBe(2);
}));
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

fakeAsync / useFakeTimers

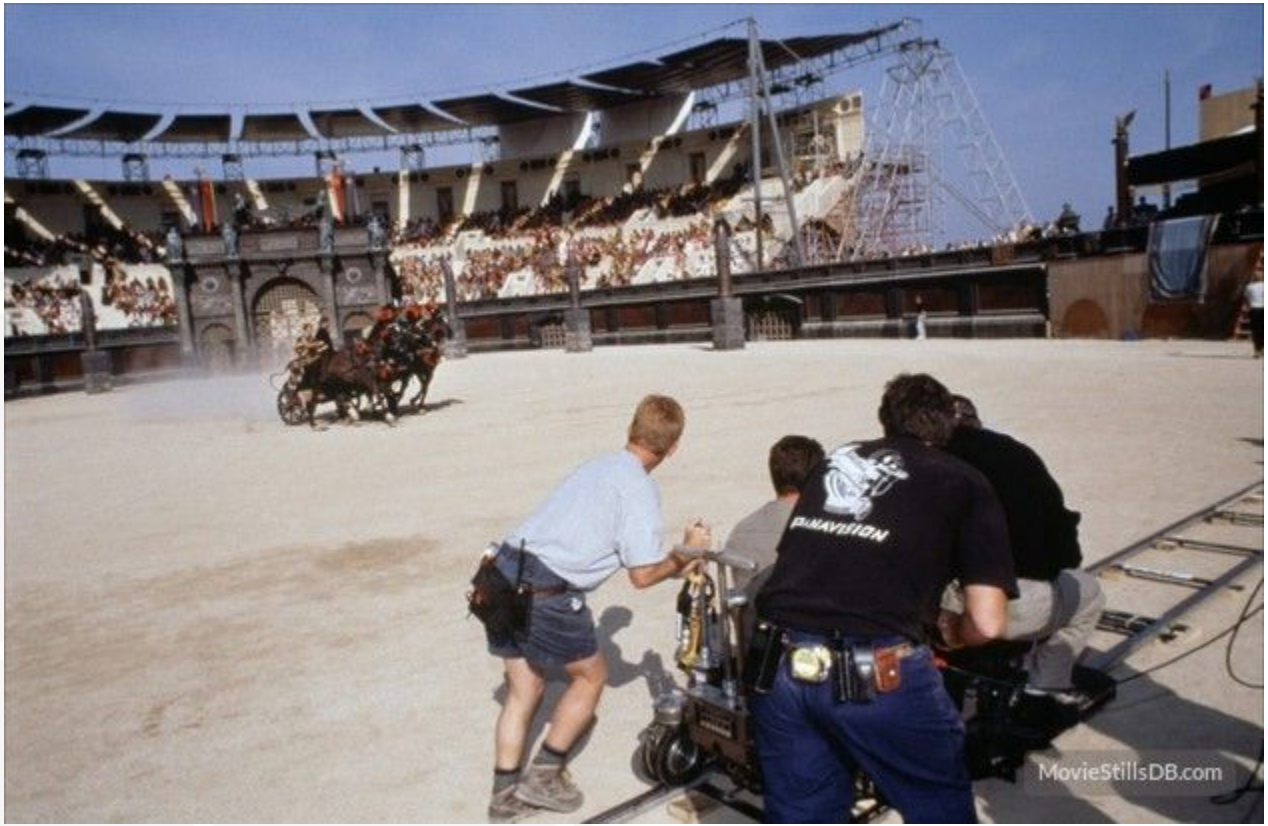
```
test("immediate macrotasks", fakeAsync(() => {
  let a = 1;
  window.setTimeout(() => (a = 2));
  expect(a).toBe(1);


  tick();
  expect(a).toBe(2);
}));
```

fakeAsync / useFakeTimers

```
test("delayed macrotasks", fakeAsync(() => {
  let a = 1;
  window.setTimeout(() => (a = 2), 2000);
  expect(a).toBe(1);


  tick(2000);
  expect(a).toBe(2);
}), 1000);
```

Observables are
not asynchronous
by definition

Mocking (Test Doubles)



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Two Types

1. Verify a call to a dependency: Mock
    a. A "side-effect only" dependency
    b. Usage has to be verified
    c. e.g. SnackBar, Router navigation
2. Replace a dependency: Stub
    a. When dependency returns a value
    b. No need to verify it is called
    c. e.g. HTTP Request
    d. Is enough in most cases

Mocking Functions

```
export interface AddressSource {
  value: string;
  expiryDate: Date;
}


export function isValidAddress(addressSource: AddressSource): boolean {
  return isPast(addressSource.expiryDate);
}
```

Mocking Functions

```
export class ValidAddressLookuper {
  constructor(private addresses: () => AddressSource[]) {}

  lookup(query: string): boolean {
    return this.addresses()
      .filter(isValidAddress)
      .some((address) => address.value.startsWith(query));
  }
}
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Mocking Function

__mocks__/is-valid-address.ts

```
import { AddressSource } from "../address-source";

export function isValidAddress(addressSource: AddressSource): boolean {
  return true;
}
```

Mocking Function

```
jest.mock('./is-valid-address');

it('should mock expired address source', () => {
  const lookuper = new ValidAddressLookuper(() => [
    {
      value: 'Domgasse 5',
      expiryDate: new Date(2000, 0, 1)
    }
  ]);

  expect(lookuper.lookup('Domgasse 5')).toBe(true);
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Alternative (Automatic Mocking)

```
import { isValidAddress } from "./is-valid-address";

jest.mock("./is-valid-address", () => {
  isValidAddress: () => true;
});

it('should mock expired address source', () => {
  const lookuper = new ValidAddressLookuper(() => [
    {
      value: 'Domgasse 5',
      expiryDate: new Date(2000, 0, 1)
    }
  ]);

  expect(lookuper.lookup('Domgasse 5')).toBe(true);
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

But isn't Angular all about DI and Classes?

```
export class ValidAddressLookuper {
  constructor(
    private addresses: () => AddressSource[],
    private addressValidator: AddressValidatorService
  ) {}

  lookup(query: string): boolean {
    return this.addresses()
      .filter((addressSource) => this.addressValidator.isValidAddress(addressSource))
      .some((address) => address.value.startsWith(query));
  }
}
```

Automatic Mock for Classes

```
class AddressValidator {

 isValidAddress(addressSource: AddressSource): boolean {

   return isPast(addressSource.expiryDate);

 }

}
```

Mocking Functions

return value

arguments of functions

```
const validatorFn = jest.fn<boolean, [AddressSource]>(
  (addressSource) => true
);
```

optional implementation

Automatic Mock for Classes

```
jest.mock("./address-validator", () => ({

  AddressValidator: jest.fn<AddressValidator, []>().mockImplementation(() => ({

    isValidAddress: jest.fn<boolean, [AddressSource]>((as) => true),

  })),

}))
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Pragmatic mocking

```
const addressValidator: Partial<AddressValidator> = {
  isValidAddress: jest.fn<boolean, [AddressSource]>((addressSource) => true),
};
```

jest.fn

```
it('should mock validator', () => {
  const validator = { isValidAddress: jest.fn(() => true) };
  const lookuper = new ValidAddressLookuper(
    () => [
      {
        value: 'Domgasse 5',
        expiryDate: new Date(2000, 0, 1)
      }
    ],
    (validator as unknown) as AddressValidatorService
  );

  expect(lookuper.lookup('Domgasse 5')).toBe(true);
});
```

No Verification of mock's behaviour!!!

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

jest.fn

```
it('should mock validator', () => {
  const validator = { isValidAddress: jest.fn<boolean, [AddressSource]>(() => true) };
  const lookuper = new ValidAddressLookuper(
    () => [
      {
        value: 'Domgasse 5',
        expiryDate: new Date(2000, 0, 1)
      }
    ],
    validator as AddressValidatorService
  );

  expect(lookuper.lookup('Domgasse 5')).toBe(true);

  expect(validator.isValidAddress).toBeCalled();
  expect(validator.isValidAddress).toBeCalledWith({
    value: 'Domgasse 5',
    expiryDate: new Date(2000, 0, 1)
  });
  expect(validator.isValidAddress.mock.calls[0][0].value).toBe('Domgasse 5');
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

What do we actually want to test here?

Just use a Stub!

```
it('should stub the validator', () => {
  const validator = ({ isValidAddress: () => true } as unknown) as AddressValidatorService;
  const lookuper = new ValidAddressLookuper(
    () => [
      {
        value: 'Domgasse 5',
        expiryDate: new Date(2000, 0, 1)
      }
    ],
    validator
  );

  expect(lookuper.lookup('Domgasse 5')).toBe(true);
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Spying

```
it('should check with validator mocked', () => {
 const addressValidator = new AddressValidator();
 const spy = jest.spyOn<AddressValidator, 'isValidAddress'>(
   addressValidator,
   'isValidAddress'
 );
 const addresses = ['Domgasse 15, 1010 Wien'];
 const lookuper = new AddressLookuper(() => addresses, addressValidator);

 lookuper.lookup('Domgassse 15');

 expect(addressValidator.isValidAddress).toHaveBeenCalledWith(
   'Domgasse 15, 1010 Wien'
 );
});
```

Type Safe

Lab Time