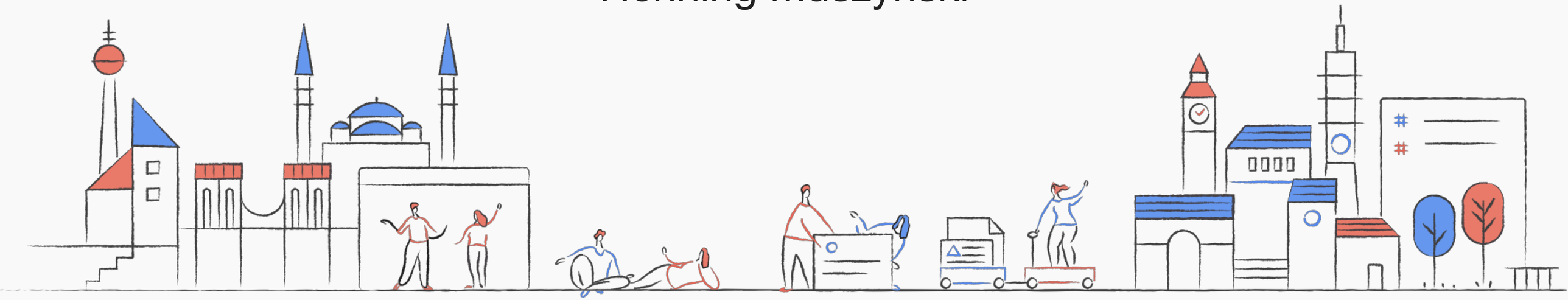
Intro to Frontend Testing

Henning Muszynski



@henningmus – Doist

Hi, I’m Henning 👋

👨‍💻 Software Engineer at Doist

🗣️ Conference Speaker

🍻 Beer Nerd & Brewer

Why Testing

- ✓ Increase Quality
- ✓ Easier & Faster Refactorings
- ✓ Avoid Regressions
- ✓ Tests as Documentation
- ✓ Reduce Fear of Changes

Test Types and Trade-offs

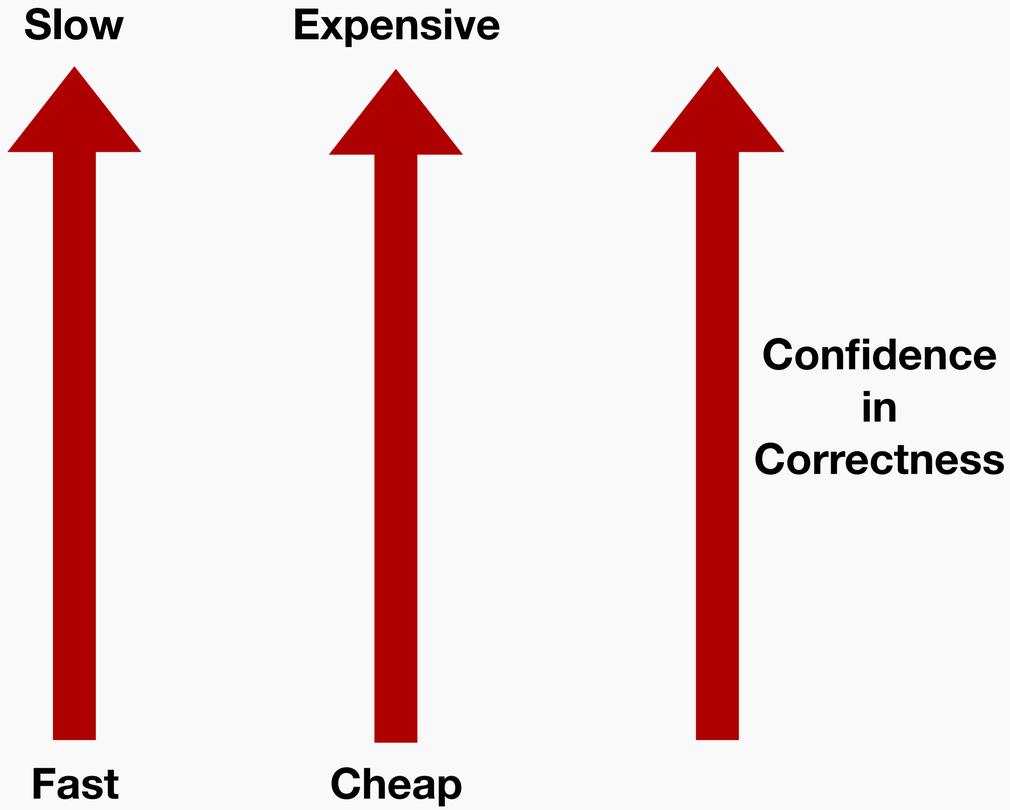
End to End

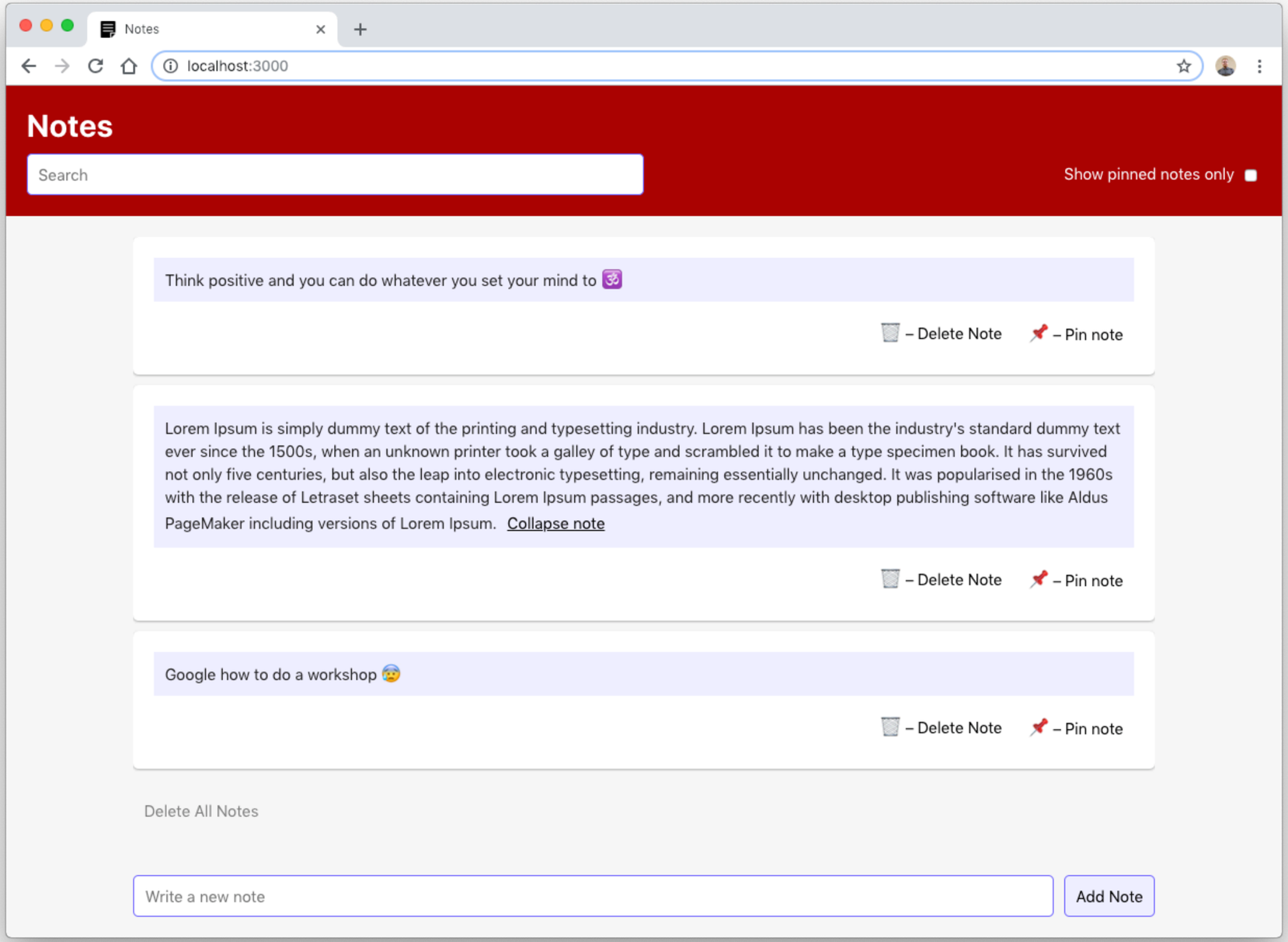
Integration

Unit

Static

testingjavascript.com by Kent C. Dodds,
Illustration by Maggie Appleton




Notes
localhost:3000
Notes
Search
Show pinned notes only
Think positive and you can do whatever you set your mind to 🕉️
– Delete Note
– Pin note
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum. Collapse note
– Delete Note
– Pin note
Google how to do a workshop 😰
– Delete Note
– Pin note
Delete All Notes
Write a new note
Add Note

Technologies

- ✓ Modern JavaScript
- ✓ React
- ✓ Jest
- ✓ Cypress

Recap: Arrow Functions

```
function sayHello() {
    console.log("Hello")
}

const sayHello = () => {
    console.log("Hello")
}
```

Recap: Arrow Functions

```
function getSumOfApples(bucket1, bucket2) {
    const sumOfApples = bucket1 + bucket2
    return sumOfApples
}

const getSumOfApples = (bucket1, bucket2) => {
    const sumOfApples = bucket1 + bucket2
    return sumOfApples
}

const getSumOfApples = (bucket1, bucket2) => bucket1 + bucket2
```

Recap: Template literals

```
const person = { name: "Henning", age: 27, role: "Engineer" }

console.log(`${person.name} is ${person.age} years old`)
// Henning is 27 years old
```

Recap: Destructuring

```
const person = { name: "Henning", age: 27, role: "Engineer" }

const { name, age } = person

console.log(name + " is " + age + " years old.")
// Henning is 27 years old
```

Recap: Object Spread

```
// Problem: copying / cloning data:

const person = { name: "Henning", age: 27, role: "Engineer" }
const person2 = person
person2.name = "Nina"

console.log(`${person.name} and ${person2.name} are ${person.age} years old`)
// Nina and Nina are 27 years old
```

Recap: Object Spread

```
const person = { name: "Henning", age: 27, role: "Engineer" }
const person2 = { ...person, name: "Nina" }

console.log(`${person.name} and ${person2.name} are ${person.age} years old`)
// Henning and Nina are 27 years old
```

Recap: Array Spread

```
const numbers = [1, 2, 3, 4]
const moreNumbers = [5, 6, 7, 8]

const allTheNumbers = [ ...numbers, ...moreNumbers ]

console.log(allTheNumbers)
// [1, 2, 3, 4, 5, 6, 7, 8]
```

Recap: Array.map()

```
const numbers = [1, 2, 3, 4]

const doubles = numbers.map((number) => {
    return number * 2
})

console.log(doubles)
// [2, 4, 6, 8]
```

Recap: Array.filter()

```
const numbers = [1, 2, 3, 4]

const onlyLargeNumbers = numbers.filter((number) => {
    return number > 2
})

console.log(onlyLargeNumbers)
// [3, 4]
```

Recap: Ternary Operator

```
const number = 3

if (number > 5) {
    console.log('number is larger than 5')
} else {
    console.log('number is lesser or equal 5')
}
// number is lesser or equal 5

number > 5
    ? console.log('number is larger than 5')
    : console.log('number is lesser or equal 5')
// number is lesser or equal 5
```

Unit Testing

Verify that individual and isolated parts (units) work as expected.

- ✓ Helper functions
- ✓ Utility functions
- ✓ Highly reused components

Assignment 1: Unit Testing

Integration Testing

Verify that several units work together as expected.

- ✓ Complex components
- ✓ Pages / views of your app

Assignment 2: Integration Testing

End to End Testing

Run your app to ensure functionality of core workflows.

- ✓ Your whole app with or without backend
- ✓ Ensure visual integrity

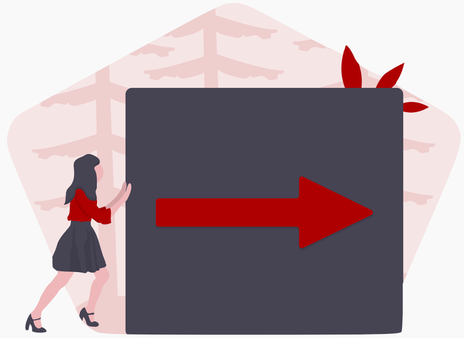
Assignment 3: End to End Testing

Achievements

- ✓ Introduction to Testing
- ✓ Writing Unit Tests
- ✓ Writing Integration Tests
- ✓ Writing End to End Tests

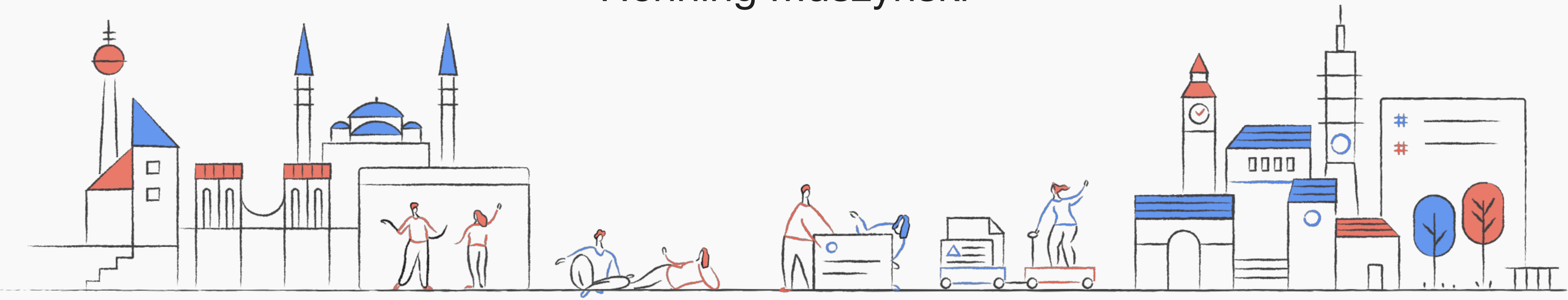
Next Steps

- ✓ Mocking: Handling network, time and randomness
- ✓ Accessibility Testing
- ✓ Testing your style guide
- ✓ Continuously run your tests



Intro to Frontend Testing

Henning Muszynski



@henningmus – Doist