

# Accessibility and Test Driven Development

A perfect match

---

Rita Castro | SDC:LX @ VW Digital Solutions | 2022-06-16 JS Nation

# Hello World JS Nation!





SOFTWARE  
DEVELOPMENT  
CENTER: LISBON



# Accessibility and Test Driven Development

*“easily used or accessed by people with disabilities”*

*“adapted for use by people with disabilities”*

*“capable of being understood or appreciated and reached”*

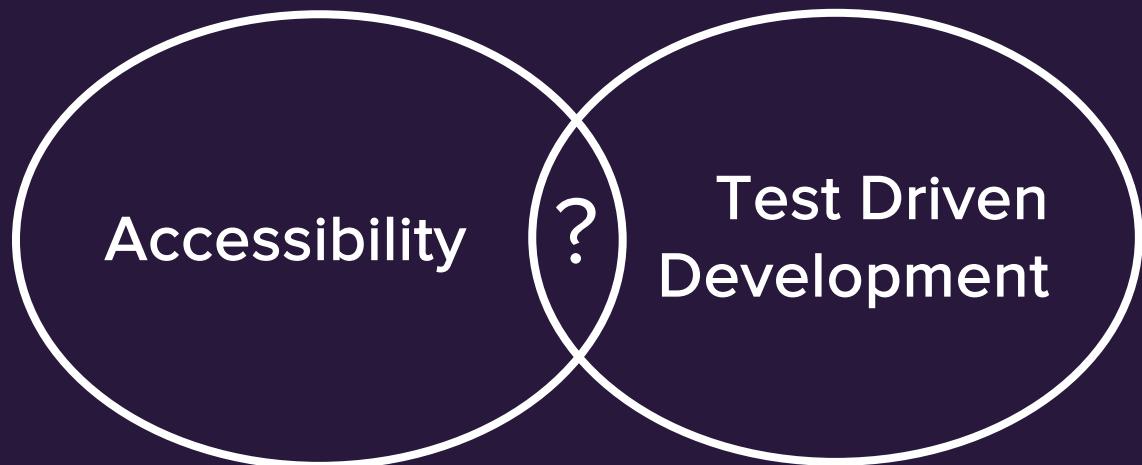


*People talking vector created by pch.vector [Adapted]*

# Accessibility and Test Driven Development

*"Software Requirements are converted into test cases before SW is developed"*





*Bad food vector created by pch.vector - www.freepik.com [Adapted]*

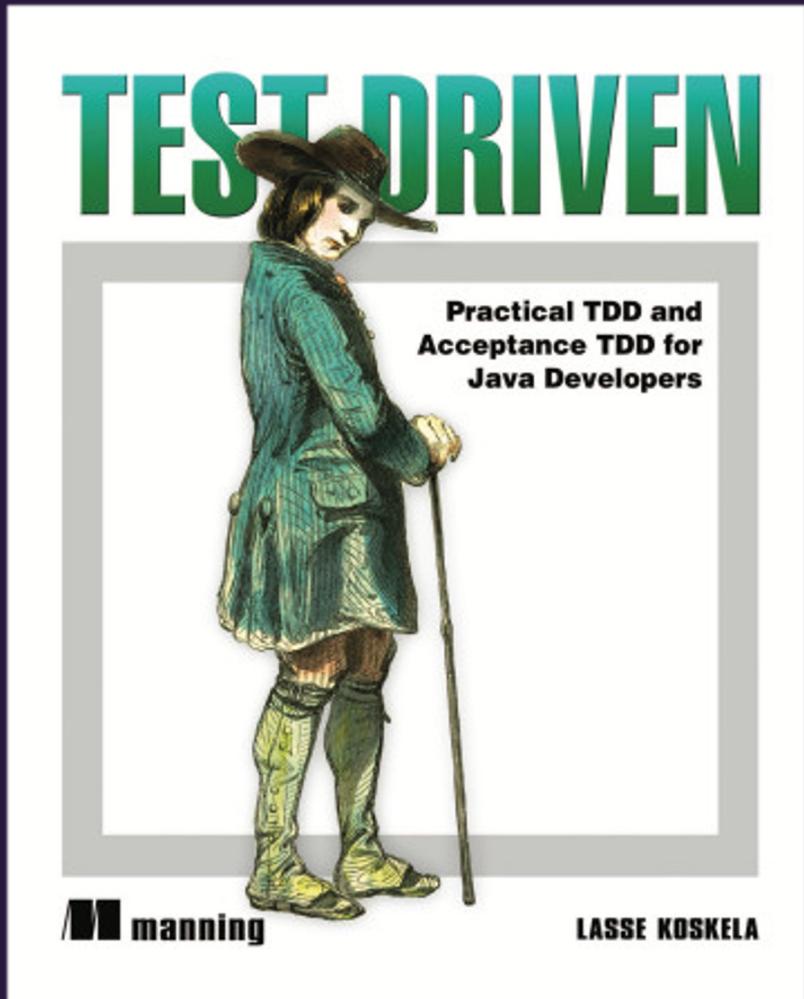
# Accessibility

a11y

*“enabling as many people as possible to use websites  
even when those people's abilities are limited in some way”*



# Test Driven Development



# The Potato example



**The User wants to be able to peel a potato.**

With a knife?

Does the knife have a wooden handle?

With a potato peeler?

Sideways or cross-section?

We should boil the potato and peel it afterwards!

We should peel it top to bottom!

# The Potato example

```
describe("The Potato example", () => {  
  
  it("is peeled!", () => {  
    // Arrange  
    let potato = Farm.get("Potato")  
  
    // Act  
    potato.peel()  
  
    // Assert  
    expect(potato).toBePeeled()  
  })  
})
```

## JS Nation asks stuff

### Personal details

**Enter your name:**

**Enter your surname:**

**Enter your email:**

### Will you also attend React Summit?

- Yes, of course!  Wouldn't miss it!

### Hangout Day

**Are you going?**

- Yes, of course!  
 I can't :(

**How are you spending it?**

**Send it to Lera!**

```
describe("Filling out Lera's form", () => {
  it ("we got this!", () => {
    // We open the page

    // In the page, we find the section of the Personal Details

    // We click "Enter your name" and...

    // ... we start typing on the input

    // We press tab to go for the "surname" field

    // ... and we keep tab'ing until we have filled the entire form

    // We look for the button with the text "Send it to Lera"...

    // ... and we click it!

    // The End :)
  })
})
```

JS Nation asks stuff

Personal details

Enter your name:

Enter your surname:

Enter your email:

Will you also attend React Summit?

Yes, of course!  Wouldn't miss it!

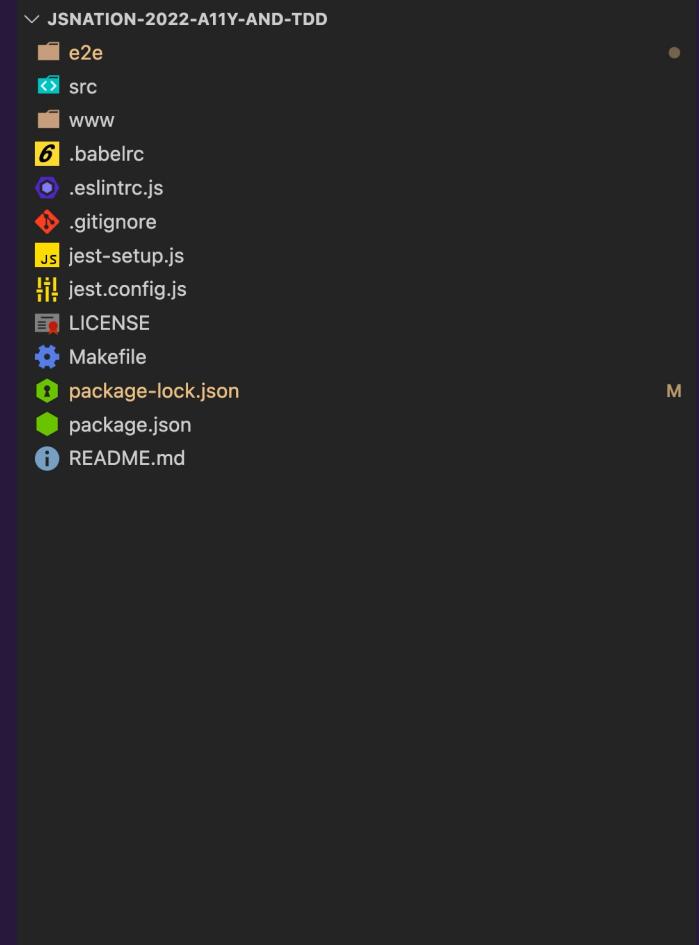
Hangout Day

Are you going?  
 Yes, of course!  
 I can't :(

How are you spending it? --Please choose an option-- ▾

Send it to Lera!

© Rita Castro 2022



# Atomic Design

*It “details all that goes into creating and maintaining robust design systems, allowing you to roll out higher quality, more consistent UIs faster than ever before”.*



[Brad Frost, <https://bradfrost.com/blog/post/atomic-web-design/>]

# text-input molecule



```
import "./text-input"

describe("text-input module", () => {

  it("has the title and the input box", () => {
    document.body.innerHTML = `<text-input></text-input>` 

    expect(document.body.querySelector("p")).toHaveTextContent("Hi, welcome")

    expect(document.body.querySelector("input")).toBeInTheDocument()
  })
})
```

# text-input molecule

```
FAIL  src/ui/molecules/text-input/text-input.test.js
      text-input module
        ✕ has the title and the input box (13 ms)

● text-input module > has the title and the input box

  expect(received).toHaveTextContent()

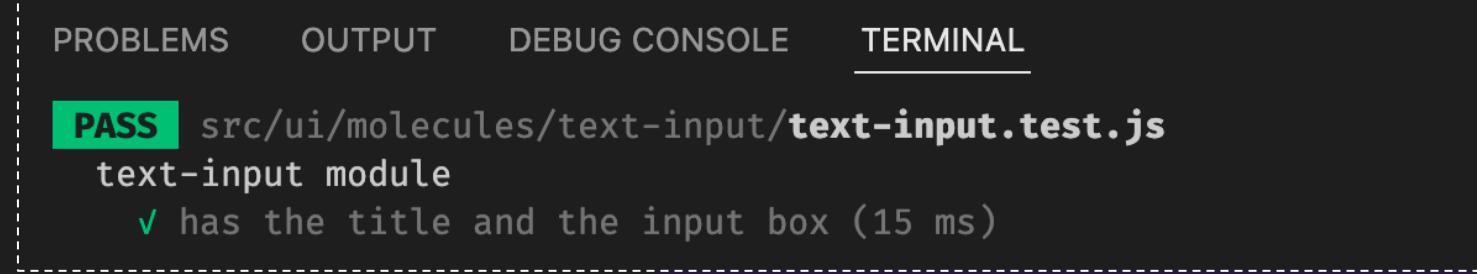
  received value must be a Node.
  Received has value: null

    6   |     document.body.innerHTML = `<text-input></text-input>`
    7   |
  > 8   |     expect(document.body.querySelector("p")).toHaveTextContent("Hi, welcome")
    9   |
  10  |       expect(document.body.querySelector("input")).toBeInTheDocument()
  11  |

  at __EXTERNAL_MATCHER_TRAP__ (node_modules/expect/build/index.js:346:30)
  at Object.throwingMatcher [as toHaveTextContent] (node_modules/expect/build/index.js:347:15)
  at Object.toHaveTextContent (src/ui/molecules/text-input/text-input.test.js:8:50)
```

# text-input molecule

```
class TextInput extends HTMLElement {  
  
  constructor() {  
    super()  
  }  
  
  connectedCallback() {  
    this.innerHTML = `  
      <p>Hi, welcome</p>  
      <input/>  
    `.  
  }  
  
}  
  
window.customElements.define("text-input", TextInput)
```



# text-input molecule

```
import "./text-input"

describe("text-input module", () => {

  it ("has a custom title and is a controlled input", () => {

    document.body.innerHTML = `<text-input title="Some random title" value="Awesome"></text-input>`

    expect(document.body.querySelector("p")).toHaveTextContent("Some random title")
    expect(document.body.querySelector("input")).toHaveValue("Awesome")
  })
})
```

# text-input molecule

```
FAIL  src/ui/molecules/text-input/text-input.test.js
  text-input module
    ✓ has the title and the input box (13 ms)
    ✗ has a custom title and is a controlled input (4 ms)

  ● text-input module > has a custom title and is a controlled input

    expect(element).toHaveTextContent()

    Expected element to have text content:
      Some random title
    Received:
      Hi, welcome

    15 |       document.body.innerHTML = `<text-input title="Some random title" value="Awesome"></text-input>`
    16 |
    > 17 |       expect(document.body.querySelector("p")).toHaveTextContent("Some random title")
          ^
    18 |       expect(document.body.querySelector("input")).toBeInTheDocument()
    19 |     })
    20 |   })

  at Object.toHaveTextContent (src/ui/molecules/text-input/text-input.test.js:17:50)

Test Suites: 2 failed, 2 total
Tests:       1 failed, 1 passed, 2 total
Snapshots:  0 total
Time:        1.304 s, estimated 2 s
```

# text-input molecule

```
class TextInput extends HTMLElement {  
  
  constructor() {  
    super()  
    this.value = this.getAttribute("value")  
  }  
  
  connectedCallback() {  
    this.innerHTML = `  
      ${this.title}  
      <input value=${this.value}></input>  
    `  
  }  
  
  window.customElements.define("text-input", TextInput)
```

# text-input molecule

## Attributes

The `<input>` element is so powerful because of its attributes; the `type` attribute, described with examples above, being the most important. Since every `<input>` element, regardless of type, is based on the `HTMLInputElement` interface, they technically share the exact same set of attributes. However, in reality, most attributes have an effect on only a specific subset of input types. In addition, the way some attributes impact an input depends on the input type, impacting different input types in different ways.

This section provides a table listing all the attributes with a brief description. This table is followed by a list describing each attribute in greater detail, along with which input types they are associated with. Those that are common to most or all input types are defined in greater detail below. Attributes that are unique to particular input types—or attributes which are common to all input types but have special behaviors when used on a given input type—are instead documented on those types' pages.

Attributes for the `<input>` element include the [global HTML attributes](#) and additionally:

Attribute	Type or Types	Description
<code>accept</code>	file	Hint for expected file type in file upload controls
<code>alt</code>	image	alt attribute for the image type. Required for accessibility
<code>autocomplete</code>	all	Hint for form autofill feature
<code>capture</code>	file	Media capture input method in file upload controls
<code>checked</code>	radio, checkbox	Whether the command or control is checked
<code>dirname</code>	text, search	Name of form field to use for sending the element's directionality in form submission
<code>disabled</code>	all	Whether the form control is disabled



*Restrictions vector created by pch.vector [Adapted]*



*Teamwork people vector created by pch.vector [Adapted]*

# form organism

```
import "./form"

describe("form organism", () => {

  it ("asks for first name and last name and sends it", () => {
    document.body.innerHTML = `<jsnation-form></jsnation-form>

      expect(document.body.querySelector("p")).toHaveTextContent("First Name")
      expect(document.body.querySelector("p")).toHaveTextContent("Last Name")

      expect(document.body.querySelector("button")).toHaveTextContent("Send it to Lera!")
    })
  })
})
```

# form organism

```
import "../molecules/text-input/text-input"

class JsNationForm extends HTMLElement {

    constructor() {
        super()
    }

    connectedCallback() {
        this.innerHTML =
            <text-input title="First Name"></text-input>
            <text-input title="Last Name"></text-input>

            <button>Send it to Lera!</button>
    }
}

window.customElements.define("jsnation-form", JsNationForm)
```

# form organism

```
FAIL  src/ui/organisms/form/form.test.js
  form organism
    ✕ asks for first name and last name and sends it (20 ms)

  ● form organism > asks for first name and last name and sends it

    expect(element).toHaveTextContent()

      Expected element to have text content:
        Last Name
      Received:
        First Name

    7 |
    8 |       expect(document.body.querySelector("p")).toHaveTextContent("First Name")
    > 9 |       expect(document.body.querySelector("p")).toHaveTextContent("Last Name")▲
    10|
    11|         expect(document.body.querySelector("button")).toHaveTextContent("Send it to Lera!")
    12|     }

  at Object.toHaveTextContent (src/ui/organisms/form/form.test.js:9:50)
  at TestScheduler.scheduleTests (node_modules/@jest/core/build/TestScheduler.js:317:13)
  at runJest (node_modules/@jest/core/build/runJest.js:407:19)

Test Suites: 1 failed, 1 total
Tests:    1 failed, 1 total
Snapshots: 0 total
Time:    0.491 s, estimated 1 s
Ran all test suites matching /src\ui\organisms\form\form\.test\.js/i.

Watch Usage: Press w to show more.□
```

# form organism

```
FAIL  src/ui/organisms/form/form.test.js
  form organism
    ✕ asks for first name and last name and sends it (20 ms)

  ● form organism › asks for first name and last name and sends it

    expect(element).toHaveTextContent()

    Expected element to have text content:
      Last Name
    Received:
      First Name

    7 |
    8 |
    > 9 |       expect(document.body.querySelector("p")).toHaveTextContent("First Name")
        expect(document.body.querySelector("p")).toHaveTextContent("Last Name") ^
    10|
    11|       expect(document.body.querySelector("button")).toHaveTextContent("Send it to Lera!")
    12|     }

  at Object.toHaveTextContent (src/ui/organisms/form/form.test.js:9:50)
  at TestScheduler.scheduleTests (node_modules/@jest/core/build/TestScheduler.js:317:13)
  at runJest (node_modules/@jest/core/build/runJest.js:407:19)

Test Suites: 1 failed, 1 total
Tests:       1 failed, 1 total
Snapshots:   0 total
Time:        0.491 s, estimated 1 s
Ran all test suites matching /src\ui\organisms\form\form\.test\.js/i.

Watch Usage: Press w to show more.[]
```

# form organism

```
import "./form"

describe("form organism", () => {

  it ("asks for first name and last name and sends it", () => {
    document.body.innerHTML = `<jsnation-form></jsnation-form>

    expect(document.body.querySelectorAll("p")[0]).toHaveTextContent("First Name")
    expect(document.body.querySelectorAll("p")[1]).toHaveTextContent("Last Name")

    expect(document.body.querySelector("button")).toHaveTextContent("Send it to Lera!")
  })
})
```

PASS src/ui/organisms/form/**form.test.js**  
form organism  
✓ asks for first name and last name and sends it (5 ms)

**Test Suites:** 1 passed, 1 total  
**Tests:** 1 passed, 1 total  
**Snapshots:** 0 total  
**Time:** 0.255 s, estimated 1 s  
Ran all test suites matching /src/ui/organisms/form/.test/i.

**Watch Usage:** Press w to show more.

# form organism

```
connectedCallback() {
  this.innerHTML =
    <text-input title="Last Name"></text-input>
    <text-input title="First Name"></text-input>

    <button>Send it to Lera!</button>
}

const button = this.querySelector("button")
button.addEventListener("click", () => {
  this.dispatchEvent(
    new CustomEvent("onClick")
  )
})
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

**FAIL** src/ui/organisms/form/**form.test.js**

form organism  
x asks for first name and last name and sends it (6 ms)

● **form organism > asks for first name and last name and sends it**

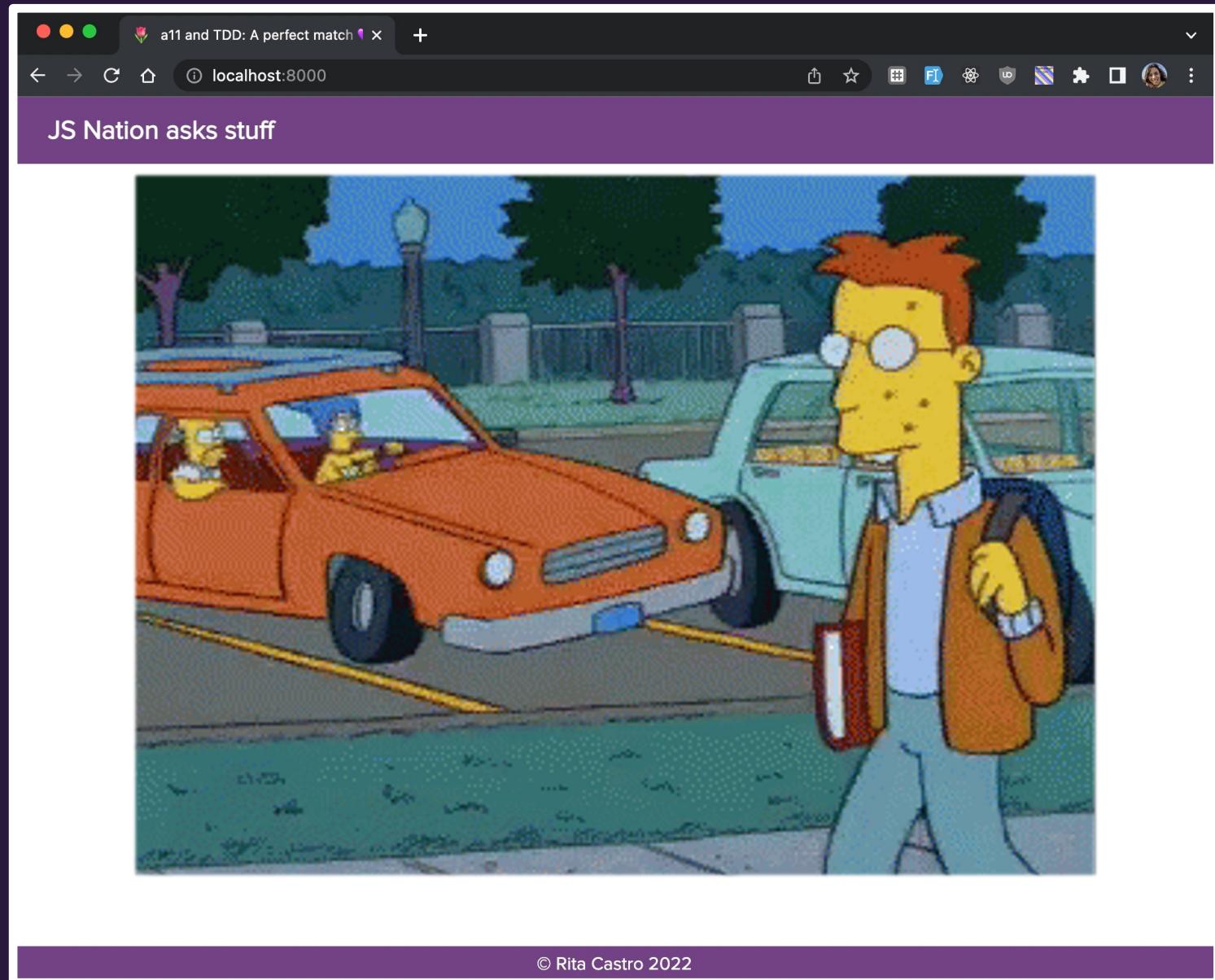
```
expect(element).toHaveTextContent()

Expected element to have text content:
  First Name
Received:
  Last Name
```

9 |       form.addEventListener("onClick", sendItFn)
10 |
> 11 |      expect(document.body.querySelectorAll("p")[0]).toHaveTextContent("First Name")
12 |      expect(document.body.querySelectorAll("p")[1]).toHaveTextContent("Last Name")
13 |
14 |     const sentItButton = document.body.querySelector("button")

at Object.toHaveTextContent (src/ui/organisms/form/form.test.js:11:56)
at TestScheduler.scheduleTests (node\_modules/@jest/core/build/TestScheduler.js:317:13)
at runJest (node\_modules/@jest/core/build/runJest.js:407:19)

**Test Suites:** 1 failed, 1 total  
**Tests:** 1 failed, 1 total  
**Snapshots:** 0 total  
**Time:** 0.302 s, estimated 1 s  
Ran all test suites matching /src/ui/organisms/form/form\.test\.js/i.



*The more your tests resemble  
the way your software is  
used, the more confidence  
they can give you.*



[Kent C. Dodds, <https://twitter.com/kentcdodds/status/977018512689455106?lang=en>]

```
describe("form organism", () => {

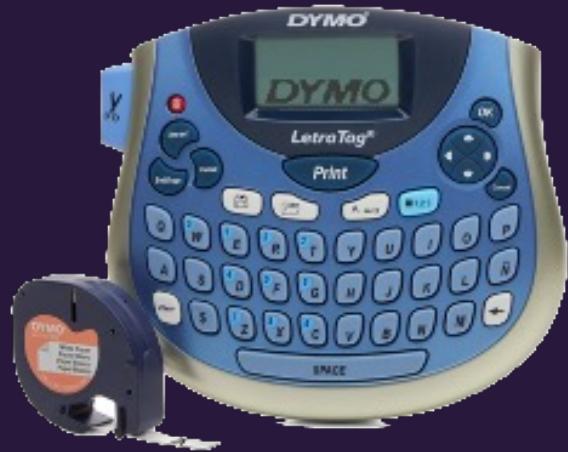
  it ("asks for first name and last name and sends it", async() => {
    const sendItFn = jest.fn()
    document.body.innerHTML = `<jsnation-form></jsnation-form>`
    const form = document.body.querySelector("jsnation-form")
    form.addEventListener("onClick", sendItFn)

    expect(screen.getByLabelText("Last Name")).toBeInTheDocument()
    expect(screen.getByLabelText("First Name")).toBeInTheDocument()

    const sentItButton = screen.getByRole("button")
    expect(sentItButton).toHaveTextContent("Send it to Lera!")

    await userEvent.click(sentItButton)
    expect(sendItFn).toHaveBeenCalled()
  })
})
```

# About labels...



*“a small piece of paper (...) attached to an object and giving information about it”*

The **<label>** HTML element represents a caption for an item in a user interface.



```
it ("populates data and sends it", async() => {
  const sendItFn = jest.fn()
  document.body.innerHTML = "<jsnation-form></jsnation-form>"
  const form = document.body.querySelector("jsnation-form")
  form.addEventListener("onSubmit", sendItFn)

  expect(screen.getByRole("form")).toBeInTheDocument()

  const firstNameInput = screen.getByLabelText("First Name")
  userEvent.click(firstNameInput)

  await userEvent.type(firstNameInput, "Rita")
  expect(firstNameInput).toHaveValue("Rita")

  userEvent.tab()
  expect(screen.getByLabelText("Last Name")).toHaveFocus()

  const lastNameInput = screen.getByRole("textbox", {name: "Last Name"})
  await userEvent.type(lastNameInput, "Castro")

  expect(lastNameInput).toHaveValue("Castro")

  await userEvent.click(screen.getByRole("button"))
  expect(sendItFn).toHaveBeenCalledWith(
    new CustomEvent({detail: {firstName: "Rita", lastName: "Castro"}})
  )
})
```

# text-input molecule revisited

```
import {screen} from "@testing-library/dom"
import "./text-input"

describe("text-input module", () => {

  it ("has a label and is a controlled input", () => {
    document.body.innerHTML = `
      <text-input labelText="Some random label" name="text" value="Awesome"></text-input>
    `

    expect(screen.getByLabelText("Some random label")).toBeInTheDocument()

    const input = screen.getByRole("textbox", {name: "Some random label"})
    expect(input).toHaveValue("Awesome")
    expect(input).toHaveAttribute("name", "text")
  })
})
```

**FAIL** src/ui/molecules/text-input/**text-input.test.js**

text-input module

✗ has a custom title and is a controlled input (8 ms)

● **text-input module > has a custom title and is a controlled input**

TestingLibraryElementError: Unable to find a label with the text of: Some random title

Ignored nodes: comments, <script />, <style />

<body>

  <text-input

    title="Some random title"

    value="Awesome"

  >

  <p>

    Some random title

  </p>

  <input

    value="Awesome"

  />

  </text-input>

</body>

```
class TextInput extends HTMLElement {  
  
    constructor() {  
        super()  
        this.value = this.getAttribute("value")  
        this.labelText = this.getAttribute("labelText")  
        this.name = this.getAttribute("name")  
    }  
  
    connectedCallback() {  
        this.innerHTML = `  
            <label for="${this.name}">  
                ${this.labelText}  
            </label>  
            <input id="${this.name}" type="${this.type}" name="${this.name}" value=${this.value}></input>  
        `;  
    }  
  
    static get observedAttributes() {  
        return ["value", "labelText", "name", "type"];  
    }  
  
    attributeChangedCallback(name, oldVal, newVal) {  
        if (name === "value") {  
            this.value = newVal;  
        }  
        if (name === "labelText") {  
            this.labelText = newVal;  
        }  
        if (name === "name") {  
            this.name = newVal;  
        }  
        if (name === "type") {  
            this.type = newVal;  
        }  
    }  
}  
  
window.customElements.define("text-input", TextInput)
```

**PASS** src/ui/organisms/form/**form.test.js**

form organism

- ✓ asks for first name and last name and sends it (30 ms)
- ✓ populates data and sends it (94 ms)

**Test Suites:** 1 passed, 1 total

**Tests:** 2 passed, 2 total

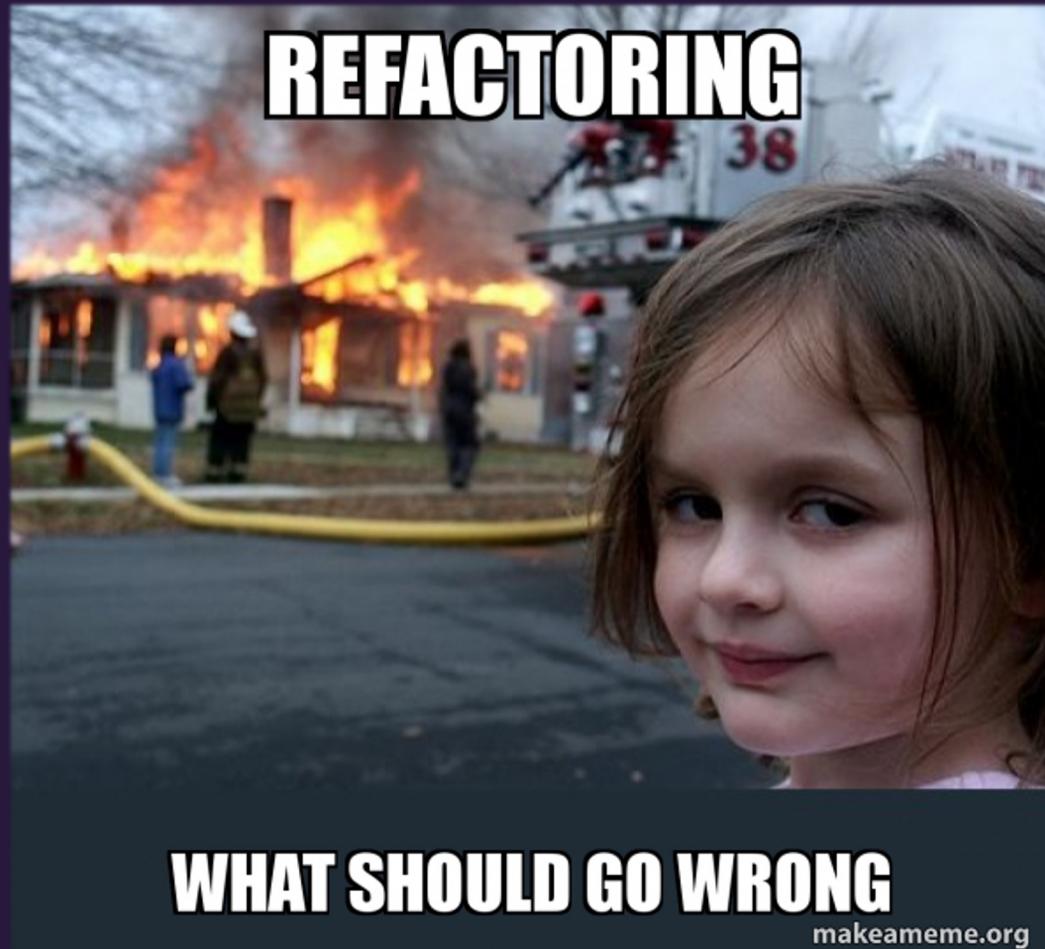
**Snapshots:** 0 total

**Time:** 0.385 s, estimated 1 s

Ran all test suites matching /src/ui/organisms/form/form.test.js/i.

**Watch Usage:** Press w to show more. █

# Refactor time!



# personal-details organism

```
import "../../molecules/text-input/text-input"

class PersonalDetails extends HTMLElement {

  constructor() {
    super()
  }

  connectedCallback() {
    this.innerHTML = `
      <fieldset name="Personal details">
        <legend>Personal details</legend>

        <text-input labelText="Enter your name:" name="firstName" value=""></text-input>
        <text-input labelText="Enter your surname:" name="lastName" value=""></text-input>
        <text-input type="email" labelText="Enter your email:" name="email" value=""></text-input>

      </fieldset>
    `
  }
}

window.customElements.define("personal-details", PersonalDetails)
```

```
import "../personal-details/personal-details"

class JsNationForm extends HTMLElement {

    constructor() {
        super()
        this.sendIt = this.getAttribute("sendIt")
    }

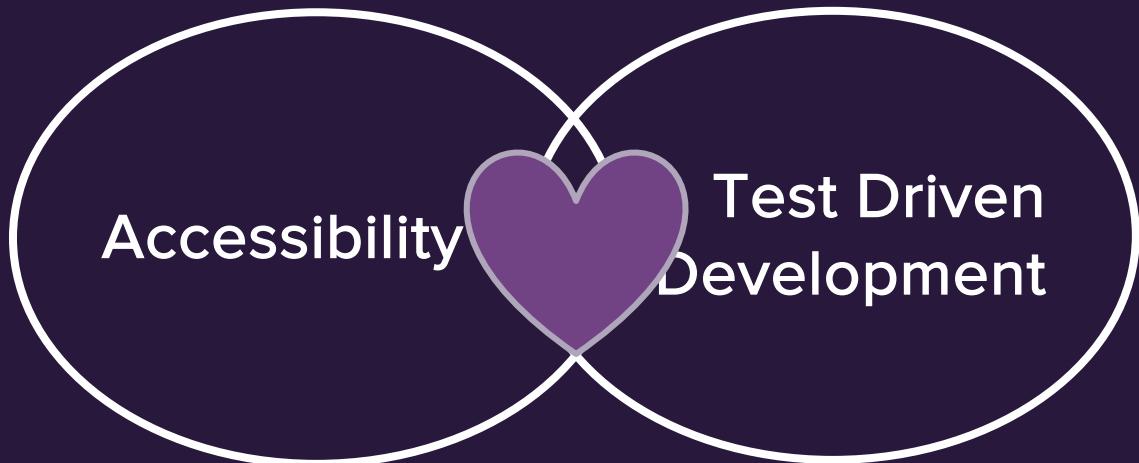
    connectedCallback() {
        this.innerHTML =
            `<form name="jsnation">

                <personal-details></personal-details>

                <input type="submit" value="Send it to Lera!" />
            </form>
`


        // Submitting the data
    }
}
```





*To transform User Stories into tests  
that drive the implementation of  
websites (and tests) that are capable  
of being understood and appreciated.*

*By all of us.*

# Thank you!

 [rita.castro@vwds.pt](mailto:rita.castro@vwds.pt)

 <https://github.com/ritamcastro>

 [ritamcastro@gmail.com](mailto:ritamcastro@gmail.com)

 <https://www.linkedin.com/in/ritamcastro>

