Web Apps don't need no manual testing 🧐

by Hendrik Wallbaum

Web Apps don't need no manual testing 🧐

by Hendrik Wallbaum

Hendrik

Developer for fun 🎉

The why and how of End-to-End testing for Web Apps.

Testing - Why we do it

- Find bugs before users do
- Ensure we fulfil specification
- Help future developers (us)

Story time

Story time - Development tasks

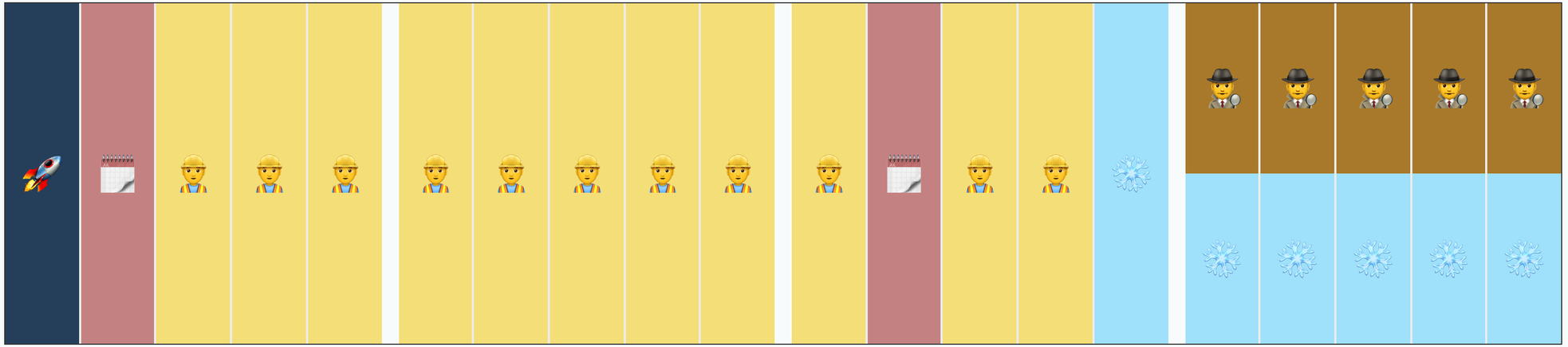
Meetings

Work on product

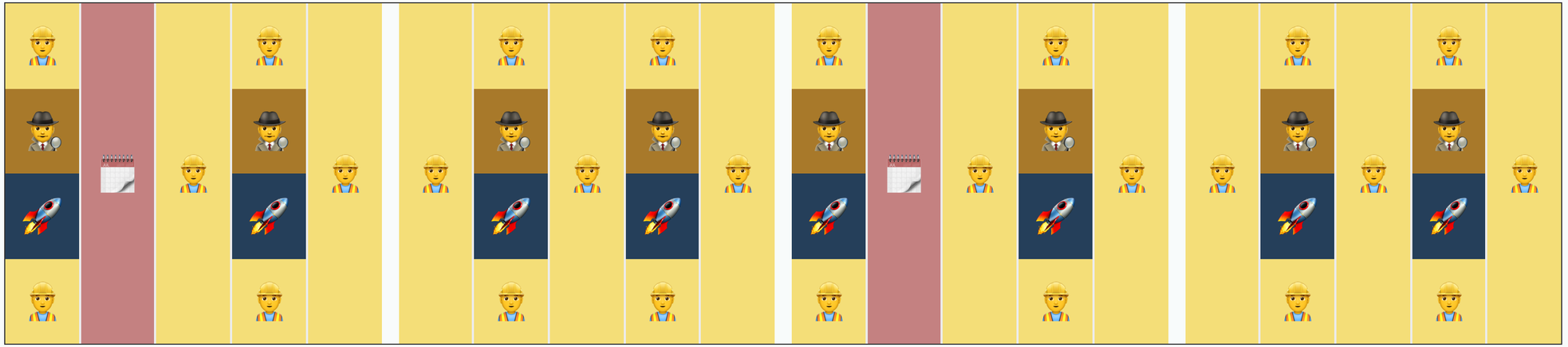
Manual testing

Releasing software

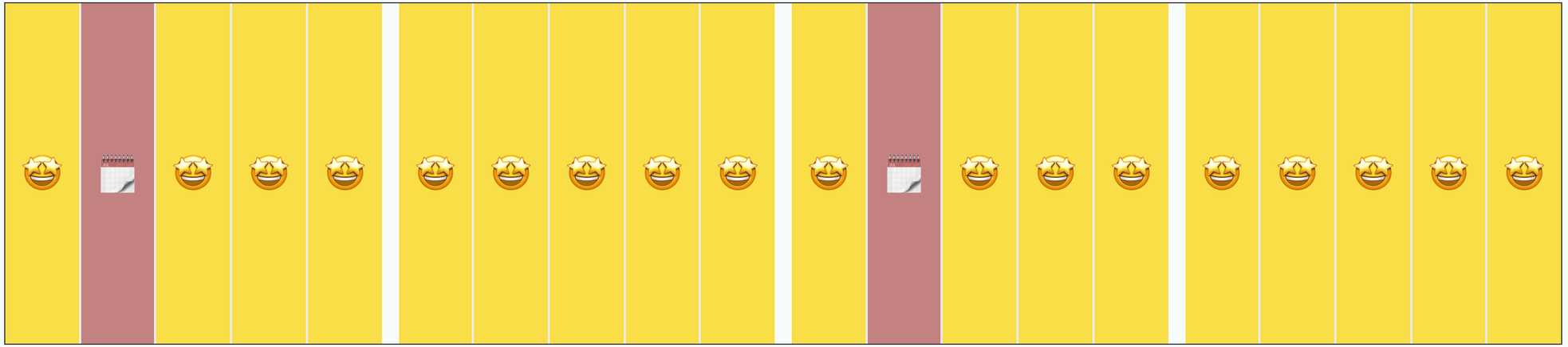
Story time - Code Freeze



Story time - Manual Releases



Story time - My dream



Story time - Time investment

Dev	Test	Meet	Releasing	Releases
11	5	2	1	1
14	2	2	2	8

Timeinvestment in days

Story time - Time investment

Dev	Test	Meet	Releasing	Releases	Bugs
11	5	2	1	**1**	🐛🐛🐛🐛🐛
14	2	2	2	**8**	🐛🐛

Timeinvestment in days

Story time - Numbers

High performing organizations:

- Deploy **200** times as often
- Have a **3** times lower change failure rate
- Recover **24** times faster from failures

DevOps Report 2016

My dream - New stats

Dev	Test	Meet	Releasing	Releases	Bugs
11	5	2	1	1	
14	2	2	2	8	
18	0	2	0	200	

Timeinvestment in days

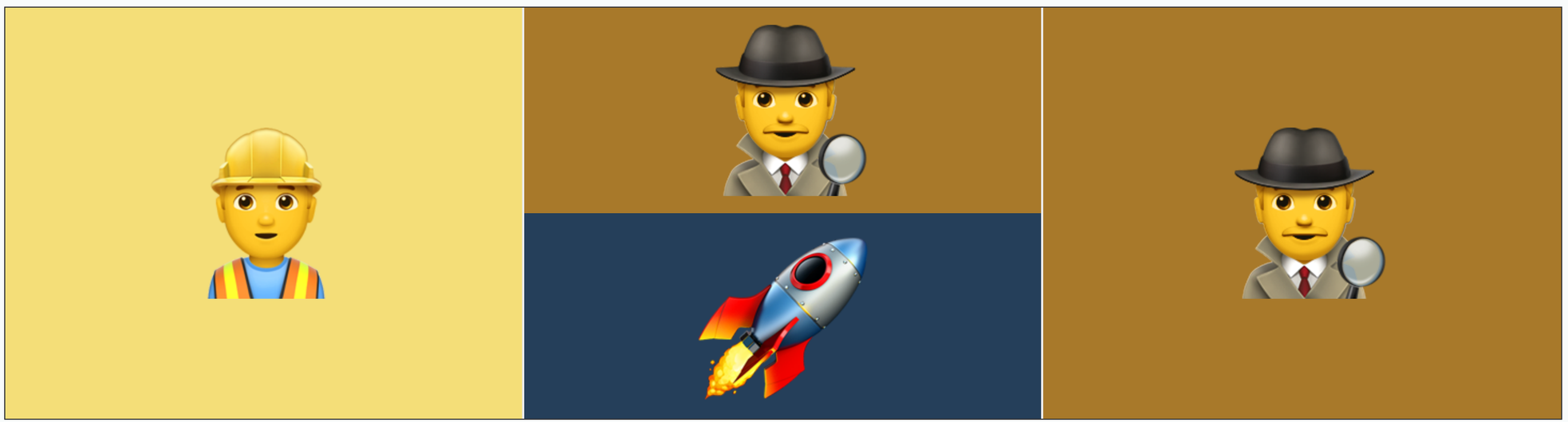
My dream - Required actors

Developer

Tester / QA

Computer

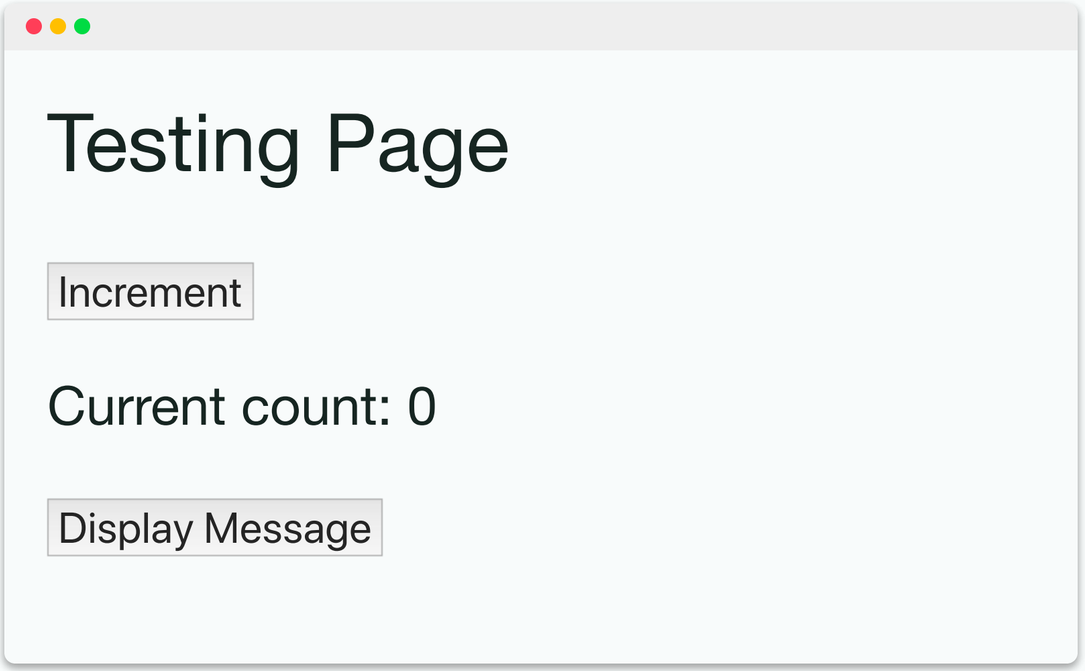
My dream - The perfect day



🔧 Use the right tool for the right job

Cypress - E2E and integration testing

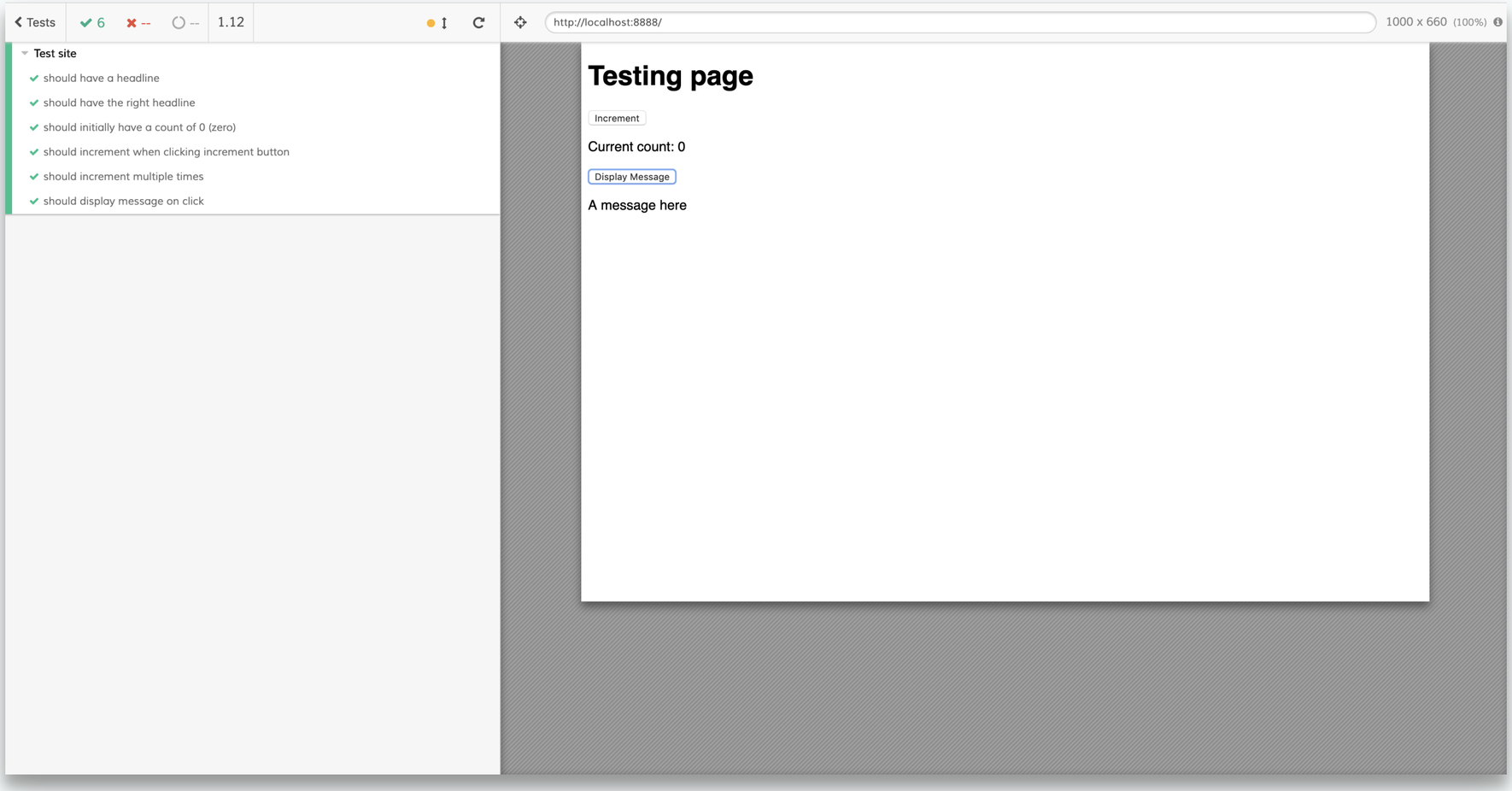
Cypress - Example app



Cypress - Testcases

1. Basic render 🎉
2. Counter logic 🧮
3. Message display 📜

Cypress - In action



Cypress - Setup

```
// Basic file structure some.spec.js.

/// <reference types="Cypress" />

describe('Some component', () => {
  it('should do something', () => {
    cy.doThings().assert('to.be', true)
  })
})
```

Cypress - Coding

1. Basic render 🎉

```
it('should have a headline', () => {
  cy.visit('/')
  cy.get('h1')
})
```

Cypress - Coding

1. Basic render 🎉

```
it('should have the right headline', () => {
  cy.visit('/')
  cy.get('h1').contains('Testing page')
})
```

Cypress - Coding

2. Counter logic 🧮

```
it('should initially have a count of 0', () => {
  cy.visit('/')
  cy.get('[data-testid="count-output"]').contains('0')
})
```

Cypress - Coding

2. Counter logic 🧮

```
it('should increment', () => {
  cy.visit('/')
  cy.get('[data-testid="button-increment"]').click()
  cy.get('[data-testid="count-output"]').contains('1')
})
```

Cypress - Coding

2. Counter logic 🧮

```
it('should increment multiple times', () => {
  cy.visit('/')
  cy.get('[data-testid="button-increment"]').click()
  cy.get('[data-testid="button-increment"]').click()
  cy.get('[data-testid="count-output"]').contains('2')
})
```

Cypress - Coding

3. Messsage display 📜

```
it('should display message on click', () => {
  cy.visit('/')
  cy.get('[data-testid="button-display"]').click()
  cy.get('[data-testid=display]')
})
```

Beyond the basics

What to run?

```
it.only('should only run this', () => {})

it.skip('should skip this', () => {})
```

Cypress - Advanced

Using previous state

```
it('should increment from previous value', () => {
  cy.visit('/')
  cy.get('[data-testid="count-output"]')
    .invoke('text')
    .then(text => {
      cy.get('[data-testid="button-increment"]').click()
      cy.get('[data-testid="count-output"]').contains(parseInt(text) + 1)
    })
})
```

Cypress - Advanced

Cypress returns Promise like values.

You can **not** `await` Cypress commands.

Thus we need to chain `.then()` should we want to use values from previous commands.

Cypress - Advanced

Fixtures

```
it('should compare to fixture', () => {
  cy.visit('/')
  cy.fixture('data').then(dataFixture => {
    cy.get('.data-element').contains(dataFixture)
  })
})
```

Cypress fixtures

Mocks and fixtures

```
it('should mock requests', () => {
  cy.server()
  cy.route(/some\/regex/, 'fixture:response.json').as('getData')
  cy.visit('/loads/data')
  cy.get('[data-testid="data"]')
})
```

Cypress - Advanced

Aliases

```
before(() => {
  cy.fixture('data').as('dataFixture')
})

it('should ...', function() {
  cy.get('element').contains(this.dataFixture)
})
```

Cypress aliases

Cypress - Best Practices

Consider what to E2E-test.

Expose APIs from your application.

`.get()` does not need positive assertions.

Speed up your tests by logging in programmatically.

More best practices

Cypress - Best Practices

```
cy.request({
  method: 'POST',
  url: 'https://your.domain',
  body: {
    password,
    username,
  },
}).then(response => {
  expect(response.isOkStatusCode).to.be.true
  const id = response.body.id
  window.localStorage.setItem('id', id)
  // Cookies are set by Cypress
})
```

Cypress - Best Practices

Custom Commands

```
Cypress.Commands.add('login', (username, password) => {
  // Login programmatically
})

// some.spec.js
beforeEach(() => {
  cy.login('username', 'password')
})
```

Cypress - Reporting

```
// cypress.json configuration.
{
  "baseUrl": "http://localhost:8888",
  "reporter": "junit",
  "reporterOptions": {
    "mochaFile": "results/test-results-[hash].xml"
  }
}
```

Cypress is build on Mocha and has acccess to all build in reporters.

Cypress reporter docs

🧐 Web Apps don't need no manual testing!?

Or do they?

Presentation

Resources

- Cypress Guides
- Cypress API
- Examples
- Repo for this presentation

🏁 The End 🏁

🤔 Questions?

Find my social media and blog on: HendrikWallbaum.de