

Module 9

Testing

Testing

What

Testing

What - Example

Build a website that sells textbooks

If the student attends a “partner” university, offer a discount

Otherwise – charge full price

Testing

What - Example



Testing What - Example

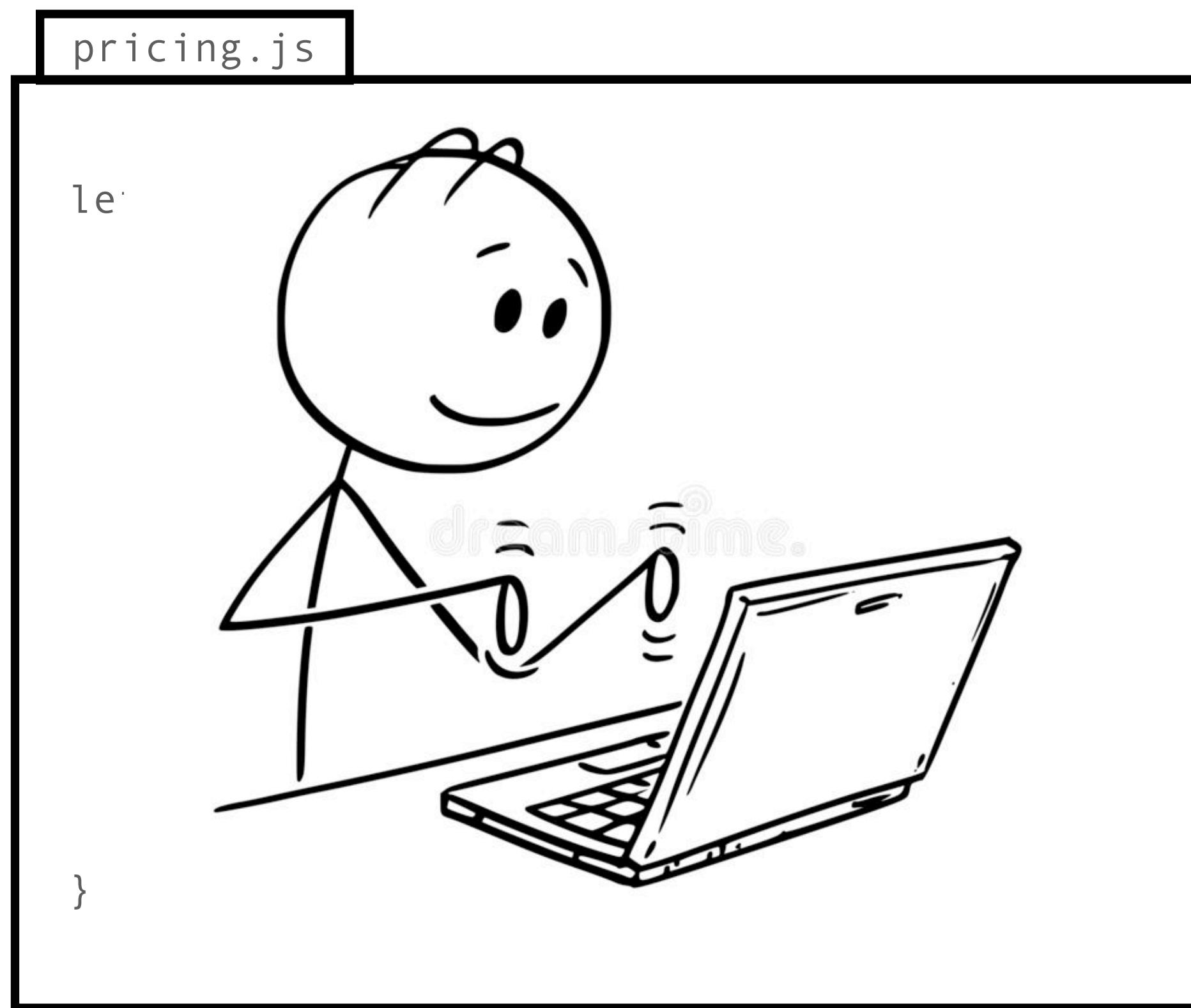
pricing.js

```
let shouldOfferDiscount = (shopper) => {
  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

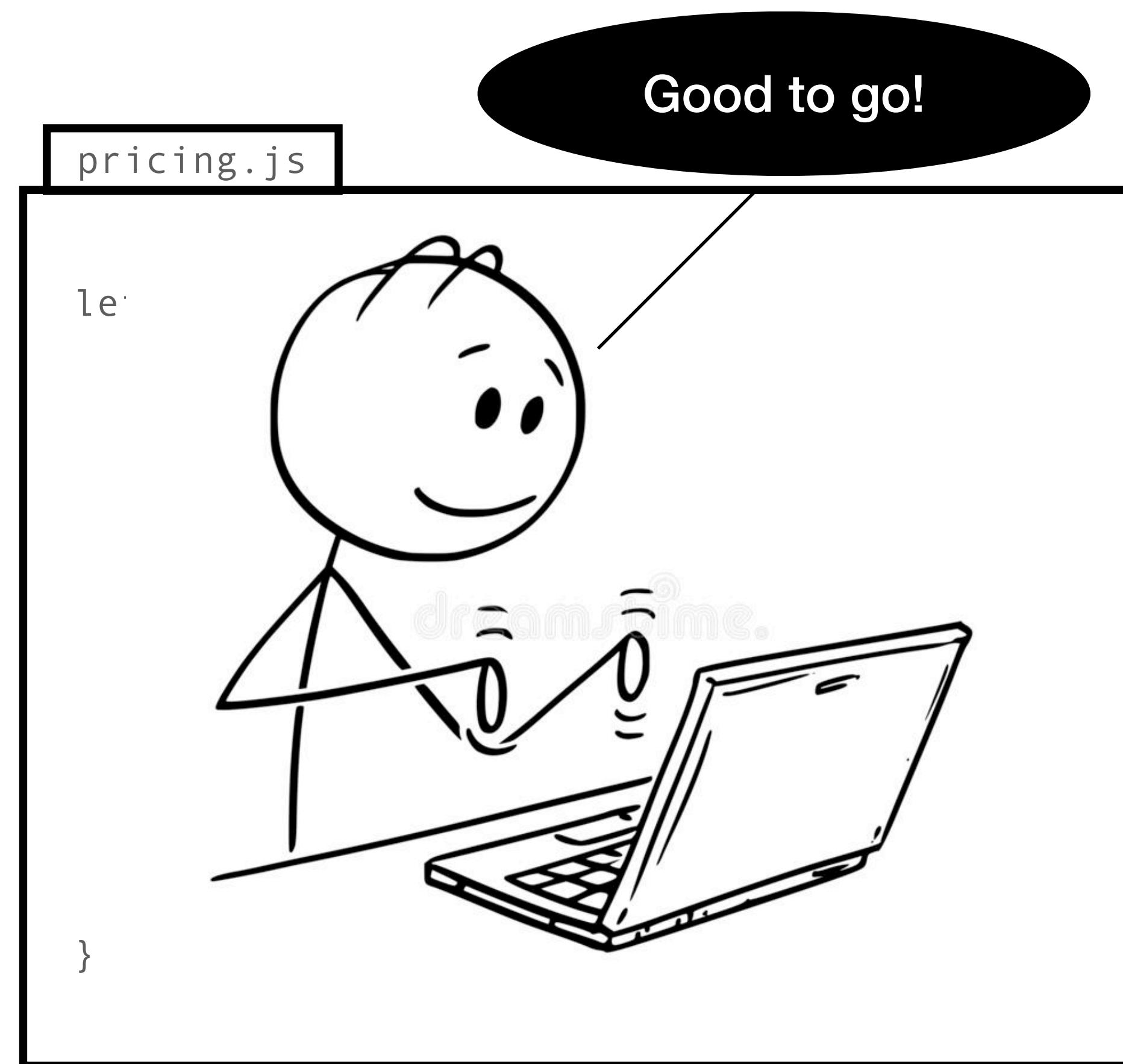
  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner)) {
      match = true;
      break;
    }
  }
  return match;
}
```

Testing

What - Example



Testing What - Example



Testing What - Example



Testing

What - Example



Testing What - Example

Email (1)

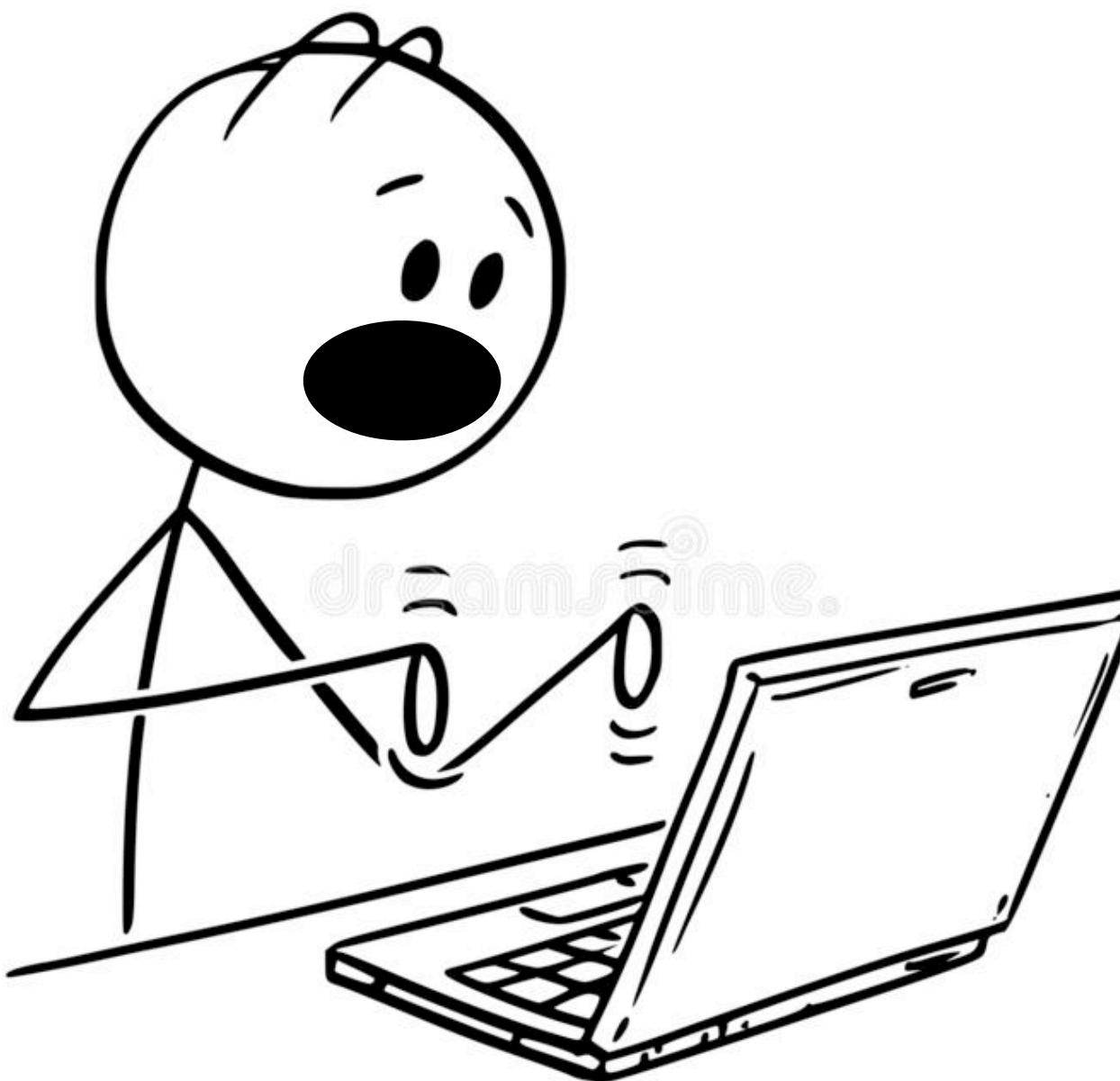
To: Nathan
From: Boss

WHY IS EVERYBODY GETTING
A DISCOUNT

this is coming out of your paycheck

Testing

What - Example



Testing What - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner)) {
      match = true;
      break;
    }
  }
  return match;
}
```

Testing What - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

Testing

What - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]
  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure non-partners don't get discount
console.assert(shouldOfferDiscount("nathan@gmail.com") === false);
console.assert(shouldOfferDiscount("john@hotmail.com") === false);
console.assert(shouldOfferDiscount("elsie@yahoo.com") === false);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);
```

Testing

What - Example



Testing

What - Example



Testing

What - Example

Email (1)

To: Nathan
From: Boss

I forgive you for bug

Can you make my nephew always get
a discount?

Testing

What - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure non-partners don't get discount
console.assert(shouldOfferDiscount("nathan@gmail.com") === false);
console.assert(shouldOfferDiscount("john@hotmail.com") === false);
console.assert(shouldOfferDiscount("elsie@yahoo.com") === false);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);
```

Testing What - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);

# Ensure non-partners don't get discount
console.assert(shouldOfferDiscount("nathan@gmail.com") === false);
console.assert(shouldOfferDiscount("john@hotmail.com") === false);
console.assert(shouldOfferDiscount("elsie@yahoo.com") === false);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);
```

Testing

What - Example

Testing

Why

Testing

Why



Testing Why

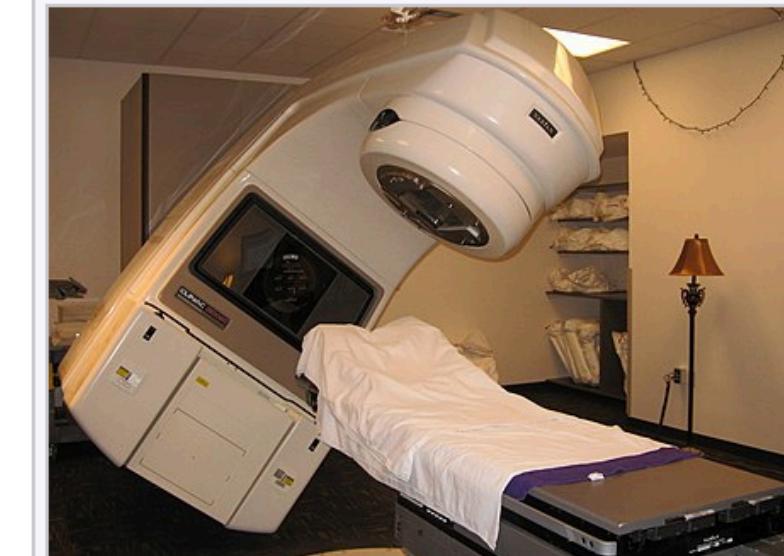
Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

Therac-25

From Wikipedia, the free encyclopedia

The **Therac-25** was a computer-controlled **radiation therapy** machine produced by **Atomic Energy of Canada Limited** (AECL) in 1982 after the Therac-6 and Therac-20 units (the earlier units had been produced in partnership with **CGR** of **France**). It was involved in at least six accidents between 1985 and 1987, in which patients were given massive **overdoses of radiation**.^{[1]:425} Because of concurrent programming errors (also known as race conditions), it sometimes gave its patients radiation doses that were hundreds of times greater than normal, resulting in death or serious injury.^[2] These accidents highlighted the dangers of software control of safety-critical systems, and they have become a standard case study in **health informatics** and **software engineering**. Additionally, the overconfidence of the engineers^{[1]:428} and lack of proper due diligence to resolve reported **software bugs** are highlighted as an extreme case where the engineers' overconfidence in their initial work and failure to believe the end users' claims caused drastic repercussions.



A Radiotherapy Machine similar to the Therac-25

<https://en.wikipedia.org/wiki/Therac-25>

Testing

Why

Lethal Software Defects: Patriot Missile Failure

Thursday, March 13th, 2014 by Michael Barr

During the [Gulf War](#), twenty-eight U.S. soldiers were killed and almost one hundred others were wounded when a nearby Patriot missile defense system failed to properly track a [Scud missile](#) launched from Iraq. The cause of the failure was later found to be [a programming error](#) in the computer embedded in the Patriot's weapons control system.

On February 25, 1991, Iraq successfully launched a Scud missile that hit a U.S. Army barracks near Dhahran, Saudi Arabia. The 28 deaths by that one Scud constituted the [single deadliest incident of the war](#), for American soldiers. Interestingly, the “Dhahran Scud”, which killed more people than all 70 or so of the earlier Scud launches, was apparently the last Scud fired in the Gulf War.

([Link](#))

Testing

Why



Testing

Why

Table 8-1. National Economic Impact Estimates

	The Cost of Inadequate Software Testing Infrastructure (billions)	Potential Cost Reduction from Feasible Infrastructure Improvements (billions)
Software developers	\$ 21.2	\$ 10.6
Software users	\$ 38.3	\$ 11.7
Total	\$ 59.5	\$ 22.2

NIST, The Economic Impacts of Inadequate Infrastructure for Software Testing, 2003 ([link](#))

Testing

Why

Catch bugs now

Prevent bugs later

Build confidence

Testing

How

Testing

How

Tests-Later Development (TLD)

1. Write code first
2. Write tests later

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);

# Ensure non-partners don't get discount
console.assert(shouldOfferDiscount("nathan@gmail.com") === false);
console.assert(shouldOfferDiscount("john@hotmail.com") === false);
console.assert(shouldOfferDiscount("elsie@yahoo.com") === false);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);
```

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);

# Ensure non-partners don't get discount
console.assert(shouldOfferDiscount("nathan@gmail.com") === false);
console.assert(shouldOfferDiscount("john@hotmail.com") === false);
console.assert(shouldOfferDiscount("elsie@yahoo.com") === false);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);
```

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);

# Ensure non-partners don't get discount
console.assert(shouldOfferDiscount("nathan@gmail.com") === false);
console.assert(shouldOfferDiscount("john@hotmail.com") === false);
console.assert(shouldOfferDiscount("elsie@yahoo.com") === false);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);
```

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

Testing Strategy

What should the code do?

What should the code NOT do?

Testing Strategy

What should the code do?

It should

- * give normal radiation doses**
- * for a safe amount of time**

What should the code NOT do?

It shouldn't

- * give unsafe radiation doses**
- * for extended periods of time**

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

What should the code do?

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

What should the code do?

It should

*** give nephew a discount**

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

What should the code do?

It should

- * give nephew a discount**
- * give partner schools a discount**

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

What should the code not do?

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

What should the code not do?

It shouldn't

*** give anyone else a discount!**

Testing Strategy

What should the code do?

It should

- * give nephew a discount**
- * give partner schools a discount**

What should the code NOT do?

It shouldn't

- * give anyone else a discount!**

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

Testing

How - Example

It should

- * give nephew a discount
- * give partner schools a discount

It shouldn't

- * give anyone else a discount!

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

Testing How - Example

It should
* give nephew a discount
* give partner schools a discount

It shouldn't
* give anyone else a discount!

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);
```

Testing How - Example

It should
~~* give nephew a discount~~
* give partner schools a discount

It shouldn't
* give anyone else a discount!

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);
```

Testing How - Example

It should
~~* give nephew a discount~~
*** give partner schools a discount**

It shouldn't
*** give anyone else a discount!**

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);
```

Testing How - Example

It should
~~* give nephew a discount~~
~~* give partner schools a discount~~

It shouldn't
*** give anyone else a discount!**

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);
```

Testing How - Example

It should
~~* give nephew a discount~~
~~* give partner schools a discount~~

It shouldn't
*** give anyone else a discount!**

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);

# Ensure non-partners don't get discount
console.assert(shouldOfferDiscount("nathan@gmail.com") === false);
console.assert(shouldOfferDiscount("john@hotmail.com") === false);
console.assert(shouldOfferDiscount("elsie@yahoo.com") === false);
```

Testing How - Example

It should
~~* give nephew a discount~~
~~* give partner schools a discount~~

It shouldn't
~~* give anyone else a discount!~~

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

pricing.test.js

```
# Ensure boss' nephew gets discount
console.assert(shouldOfferDiscount("boss-nephew@gmail.com") === true);

# Partners should get discount
console.assert(shouldOfferDiscount("jim@nyu.edu") === true);
console.assert(shouldOfferDiscount("bob@barnard.edu") === true);
console.assert(shouldOfferDiscount("egbert@columbia.edu") === true);

# Ensure non-partners don't get discount
console.assert(shouldOfferDiscount("nathan@gmail.com") === false);
console.assert(shouldOfferDiscount("john@hotmail.com") === false);
console.assert(shouldOfferDiscount("elsie@yahoo.com") === false);
```

Testing

How - Example

pricing.js

```
let shouldOfferDiscount = (shopper) => {
  if (shopper === 'boss-nephew@gmail.com') {
    return true;
  }

  const partners = [
    'nyu.edu',
    'barnard.edu',
    'columbia.edu'
  ]

  let match = false;
  for (partner of partners) {
    if (shopper.indexOf(partner) > -1) {
      match = true;
      break;
    }
  }
  return match;
}
```

Article Talk

Read

Edit

View history

Search Wikipedia



Edge case

From Wikipedia, the free encyclopedia



This article **needs additional citations for verification**. Please help **improve this article** by **adding citations to reliable sources**. Unsourced material may be challenged and removed.

Find sources: "Edge case" – news · newspapers · books · scholar · JSTOR (January 2014) (Learn how and when to remove this template message)

An **edge case** is a problem or situation that occurs only at an extreme (maximum or minimum) operating **parameter**. For example, a stereo speaker might noticeably distort audio when played at maximum volume, even in the absence of any other extreme setting or condition.

An edge case can be expected or unexpected. In engineering, the process of planning for and gracefully addressing edge cases can be a significant task, and yet this task may be overlooked or underestimated.

Testing Terminology

Tests

Test Cases

Edge Cases

TLD

Testing Terminology

Tests

Test Cases

Edge Cases

TLD

TDD

Testing

How

Test Driven Development (TDD)

1. Write tests first
2. Write code later

Testing How

Test Driven Development (TDD)

Example

Write a user password validator (learning-tdd)

Module 9

Testing with Jest

Testing - Jest

Why

Why use anything other than console.assert?

Testing - Jest

Why

	console.assert	Jest
Tests in production	Yes	No
Separation of code	No	Yes
Abstraction level	Low	High

Testing - Jest

How

Example
With create-react-app!

Testing - Jest

Tips

- Always name test files in the format “*<name of file being tested>.test.js*”
 - Ex. App.js => App.test.js, Utils.js => Utils.js
- Test files belong in the same folder as the code

Testing - Jest

Tips

- Wrap each block of test cases in a “describe”
- Wrap test case(s) in a “it” (or “test”)

Testing - Jest Example

Example

Username / password validation with Jest (jest-tdd)

Testing - Jest

Walkthrough

- Find the file that contains the code you want to test
- Create a test file (“pricing.test.js”)
- Start a “describe” block with the name of the function to test
- Start an “it” block with a description of what the tests will do
- Add test statements!

```
expect(getEmailDomain("n@gmail.com")).toEqual("gmail.com")
```

```
expect(add(1, 2, 3)).toEqual(6)
```

Testing - Jest

Walkthrough

Pull down the latest from [careers-in-code-classroom-codebase](#)

Go into **cohort-2/module-9/day-3/additional-jest-example**

Run **npm install**

Run **npm start**

Testing - Jest

Mocking

Why?

- * unit tests shouldn't depend on external services
- * sometimes external services will return unreliable data

How?

- * 3rd party package? jest.mock
- * internal function? jest.spyOn

Testing - Jest

Mocking

WARNING

Watch out for mocks creeping into other tests!

`spy.mockRestore` when your test is done!

Testing - Jest

Jest + Enzyme

- For more info, read <insert enzyme here>
- Otherwise, we'll use cypress for this

Module 9

Testing with Cypress

Testing - Cypress

Why

Isn't Jest good enough?

Testing - Cypress

Why

	console.assert	Jest	Cypress
Tests in production	Yes	No	No
Separation of code	No	Yes	Yes
Abstraction level	Low	High	High
Type of tests	Unit / Integration	Unit / Integration	E2e (functional) / Integration

Testing - Cypress

Unit vs Integration vs End-to-End

Unit testing

- * test how **one** function behaves

Testing - Cypress

Unit vs Integration vs End-to-End

Unit testing

- * test how **one** function behaves

Integration testing

- * test how **multiple** functions behave

Testing - Cypress

Unit vs Integration vs End-to-End

Unit testing

- * test how **one** function behaves

Integration testing

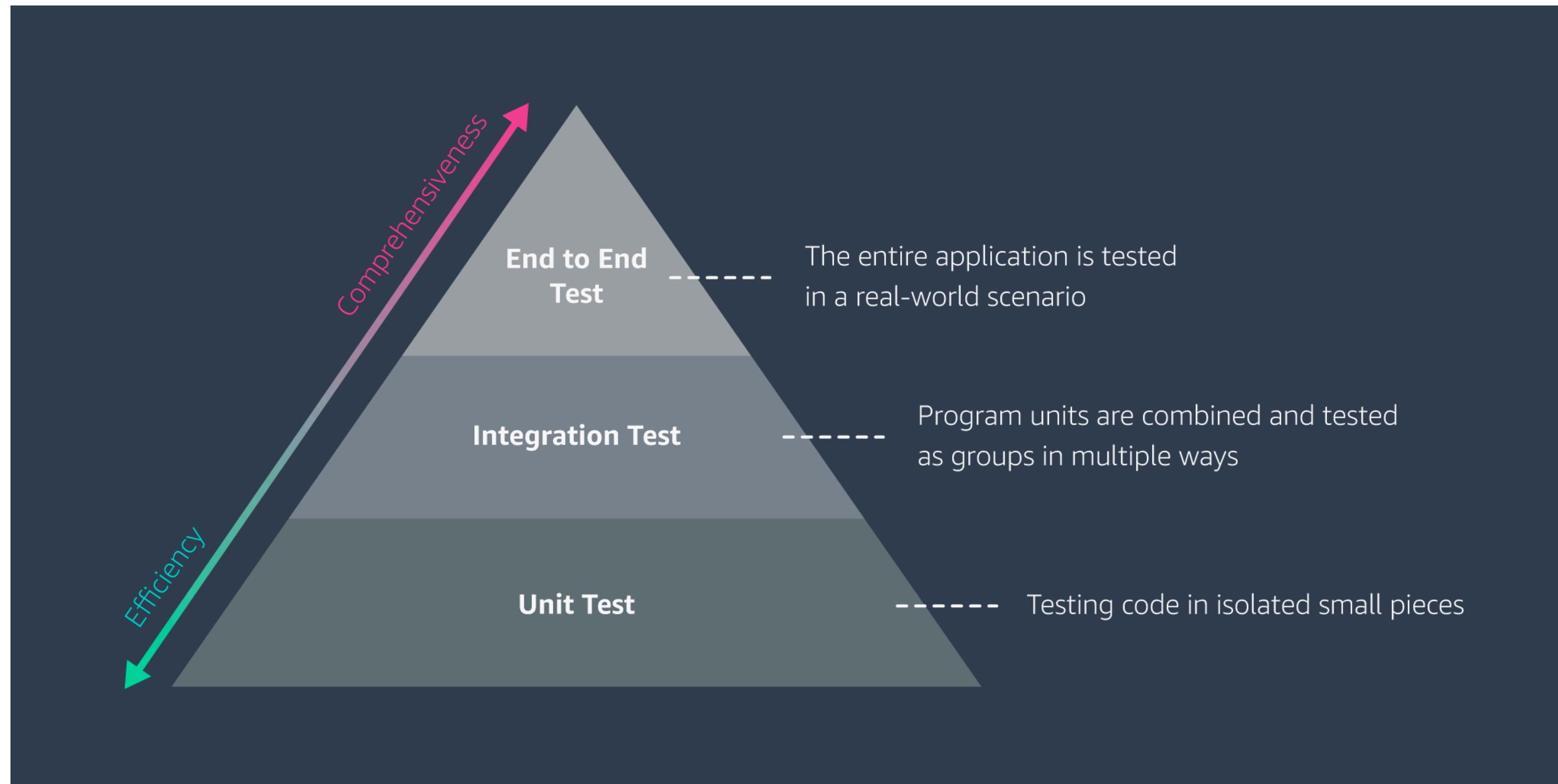
- * test how **multiple** functions behave

End to End testing (functional)

- * test how the **application** behaves

Testing - Cypress

Unit vs Integration vs End-to-End



Testing - Cypress

How



Testing - Cypress

How



Instructions

Testing - Cypress

How



Instructions

1. Open localhost:3000 in a browser
2. Login
3. Add a couple items to the shopping cart
4. Verify the items in the shopping cart are added correctly

Testing - Cypress

How

Human Instructions

1. Open localhost:3000 in a browser
2. Login
3. Add a couple items to the shopping cart
4. Verify the items in the shopping cart are added correctly

Cypress Instructions

1. cy.visit(...)
2. cy.get(...).type(...)
3. cy.get(...).click()
4. cy.get(...).contains(...)

Testing - Cypress

How

```
$ npm i --save-dev cypress  
$ ./node_modules/.bin/cypress open
```

Testing - Cypress

How

Open a website

```
cy.visit("https://google.com")
```

Testing - Cypress

How

Enter text in a box

```
cy.get("input").type("funny cat pictures")
```

Testing - Cypress

How

Enter text in a box and hit enter

```
cy.get("input").type("funny cat pictures").type("{enter}")
```

Testing - Cypress

How

Cheat sheet

Cypress tests go in the cypress/integration folder
Typically named “<something>.spec.js”

```
cy.visit("http://localhost:3000")          // open the browser to localhost:3000
cy.get("p").contains("some text")          // find a <p> element that has the text "some text"
cy.get("button").contains("Logout").click() // find a <button> element with the text "Logout" and click it
cy.get("#password").type("password1234") // find an element with id="password", and type "password1234" into it
cy.get(".inventory button").eq(3).click()  // click the 4th button inside a parent element with class="inventory"
```

Testing - Cypress

How

Resources

<https://example.cypress.io/commands/querying>

<https://docs.cypress.io/api/table-of-contents>

Testing - Backend

What about node.js?

- Backends can be tested w/ Jest too!
- `npm i --save-dev jest supertest`

Testing - Backend

What about nodejs?

Example

Testing a nodes api with jest (jest-express)