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

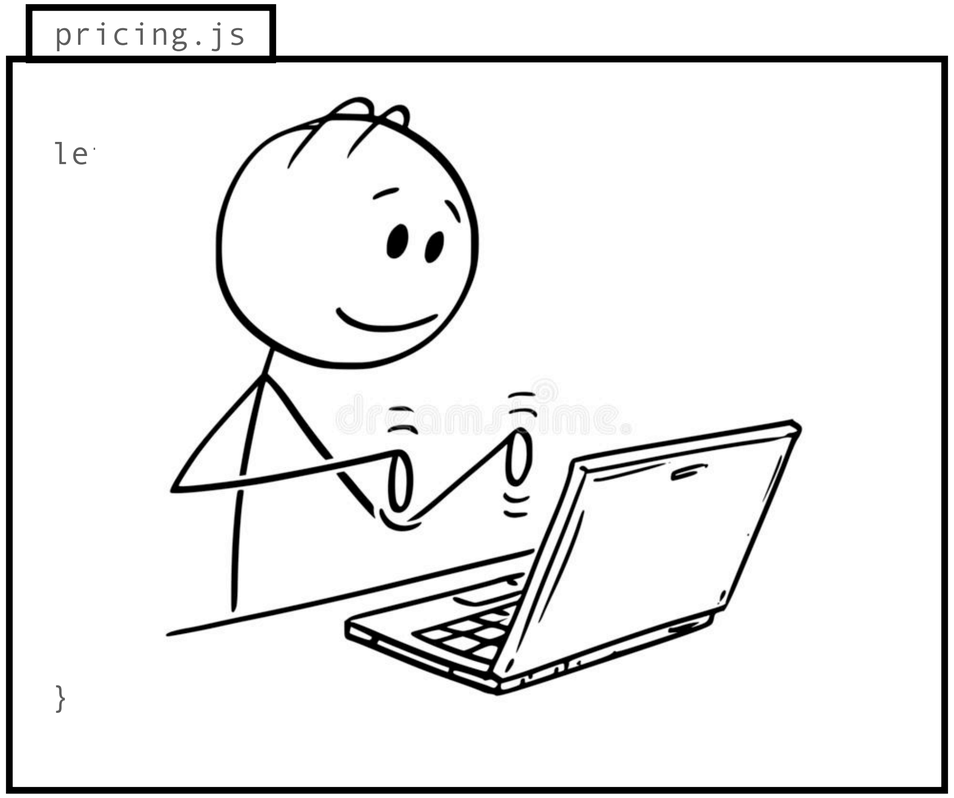
pricing.js

```
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;
}
```

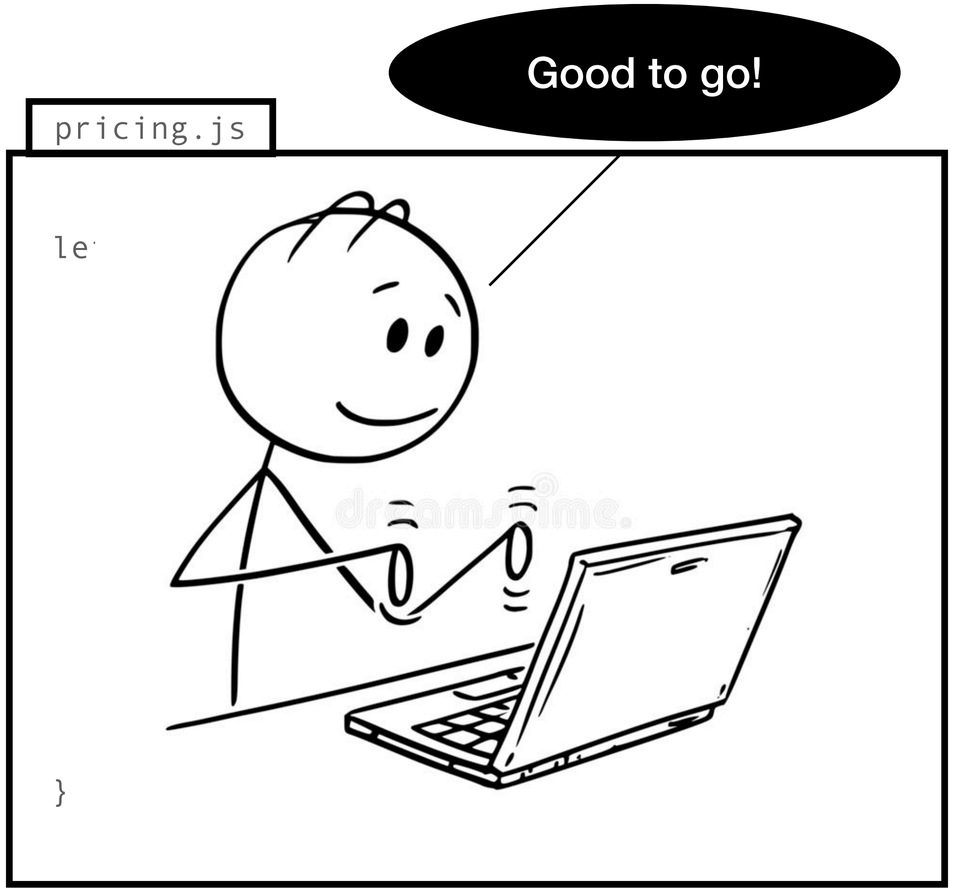
Testing

What - Example



Testing

What - Example



Testing

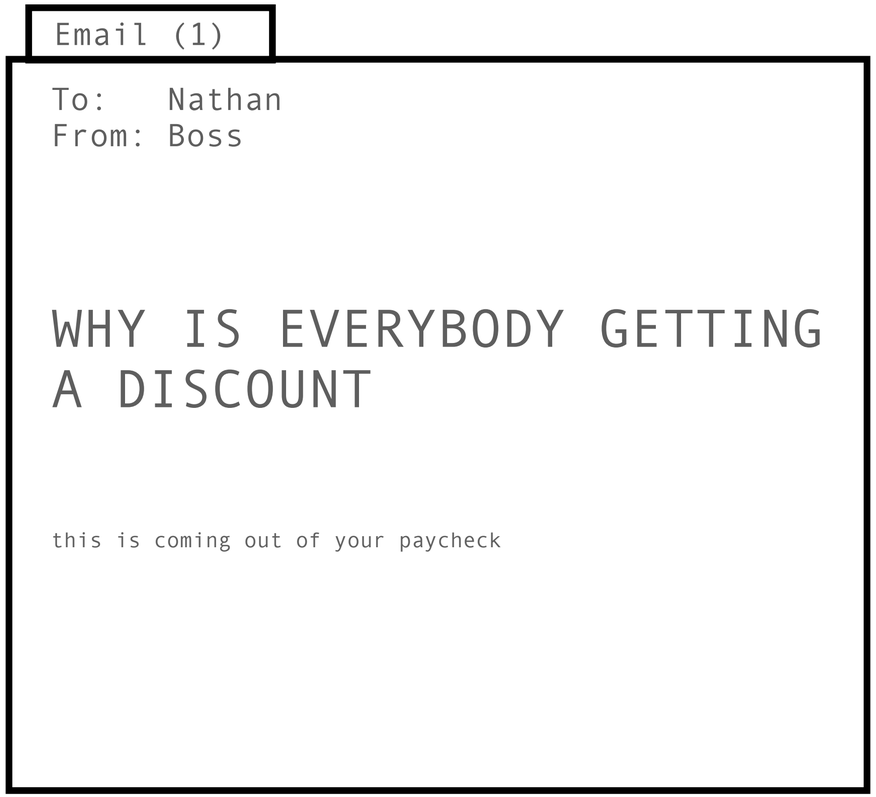
What - Example

Testing

What - Example

Testing

What - Example



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

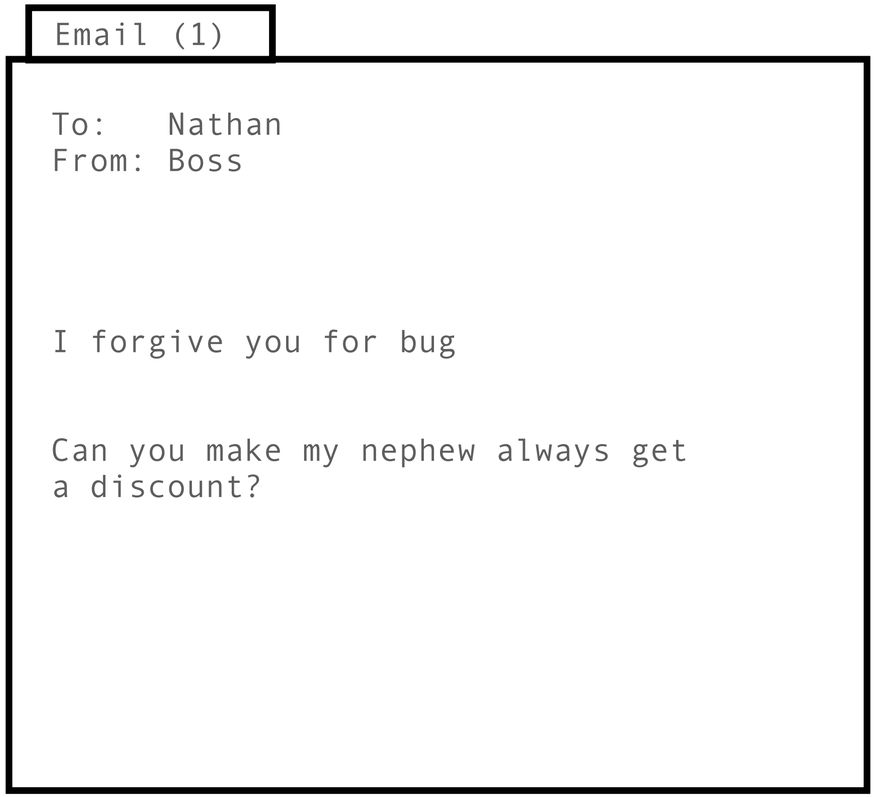
What - Example

Testing

What - Example

Testing

What - Example



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

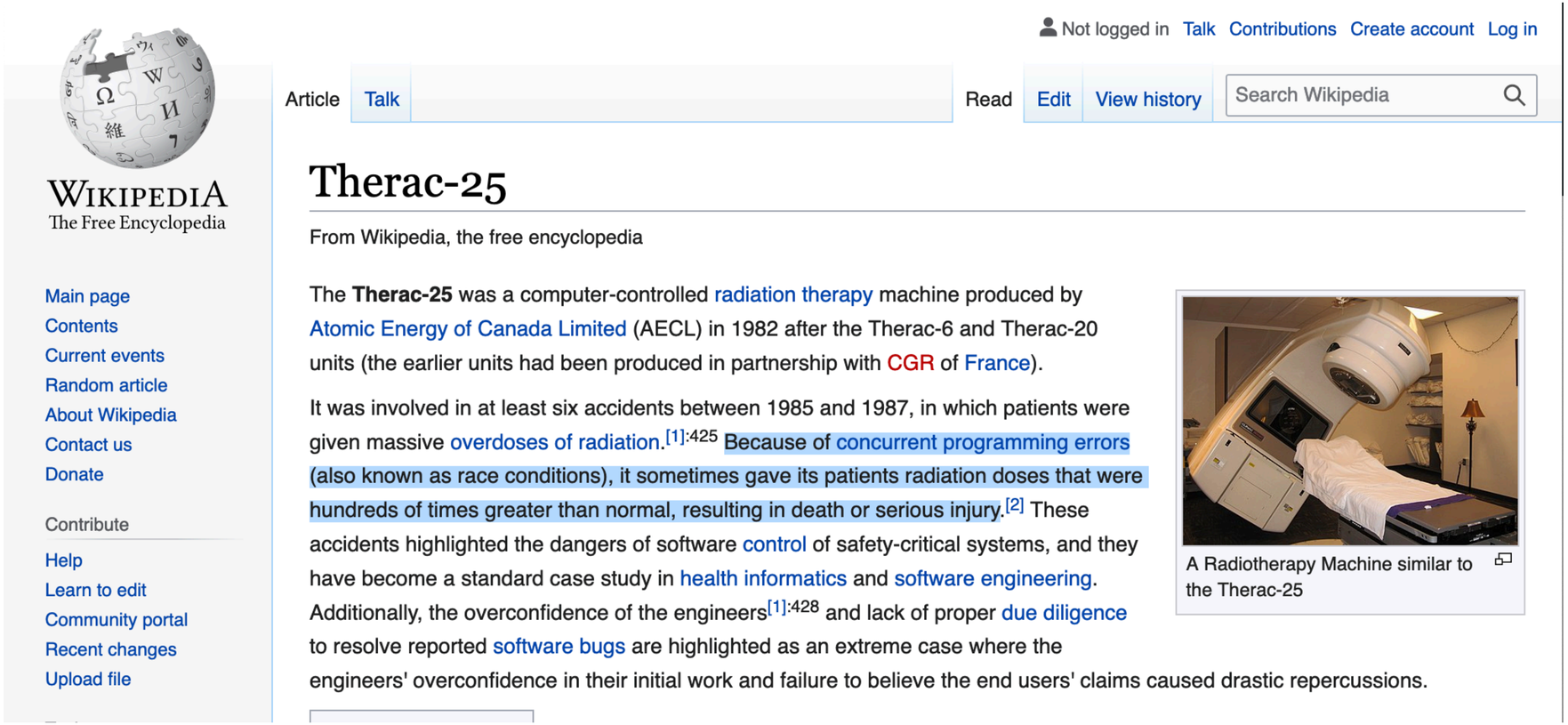
Why

Testing

Why

Testing

Why



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

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

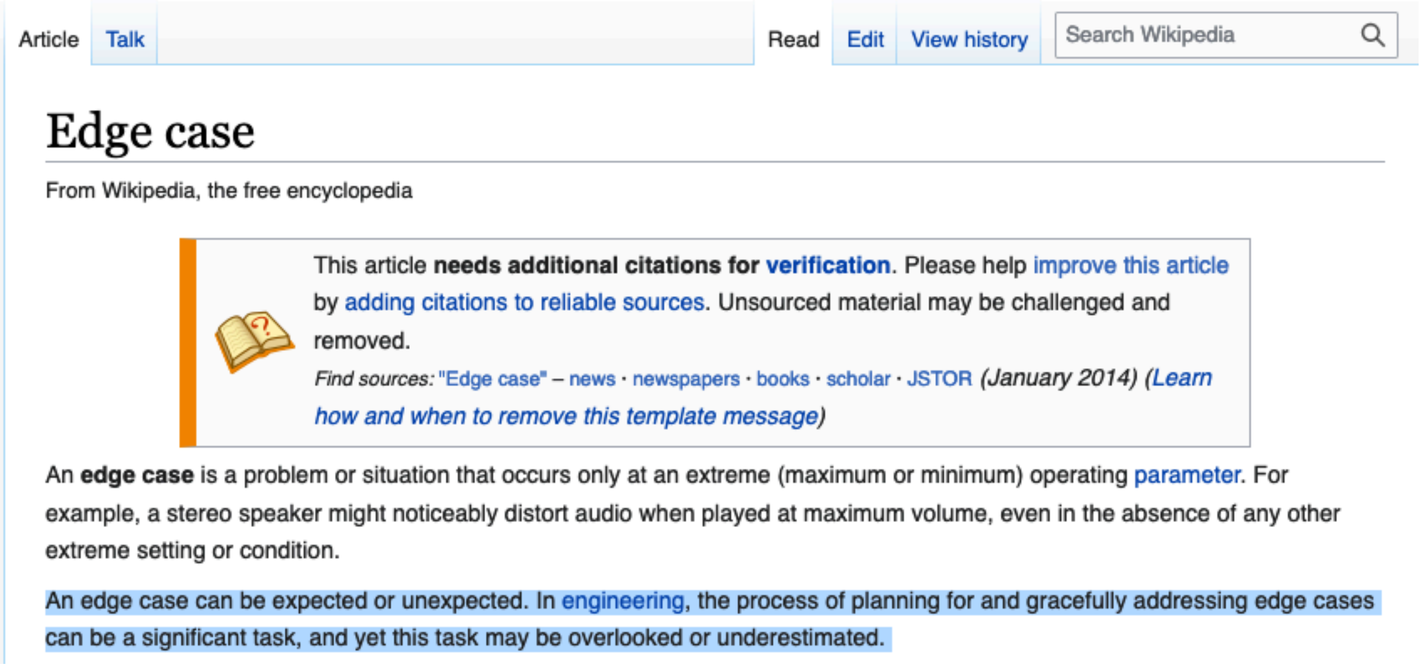
How - Example

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

Edit 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 their testing

Testing - Jest

Tips

- Describe / test
- Wrap each function in a “describe"
- Wrap test case(s) in a “test”

Testing - Jest

Example

Example

Username / password validation with Jest (jest-tdd)

Testing - Jest

Tips

- it.only to only run one test (it.skip to skip a test)
- console.log is your friend!
- Watch out for tests breaking other tests (beforeEach)

Testing - Jest

Mocking

Why?

* unit tests shouldn’t depend on external services

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 Functional

Unit testing

* test how **one** function behaves

Testing - Cypress

Unit vs Integration vs Functional

Unit testing

* test how **one** function behaves

Integration testing

* test how **many** functions behave

Testing - Cypress

Unit vs Integration vs Functional

Unit testing

* test how **one** function behaves

Integration testing

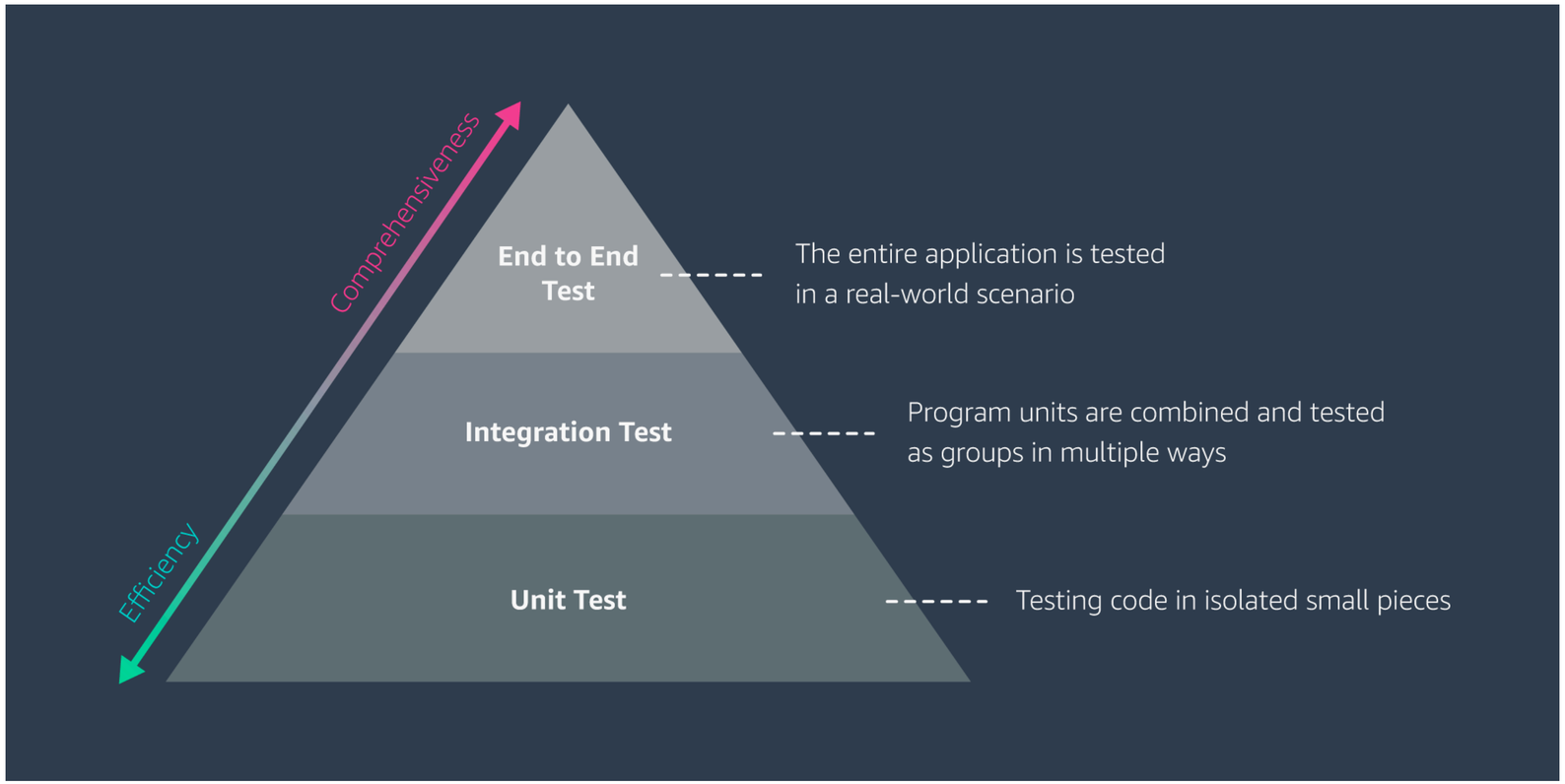
* test how **many** functions behave

End to End testing (functional)

* test how the **whole thing** behaves

Testing - Cypress

Unit vs Integration vs Functional



Testing - Cypress

How

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

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)