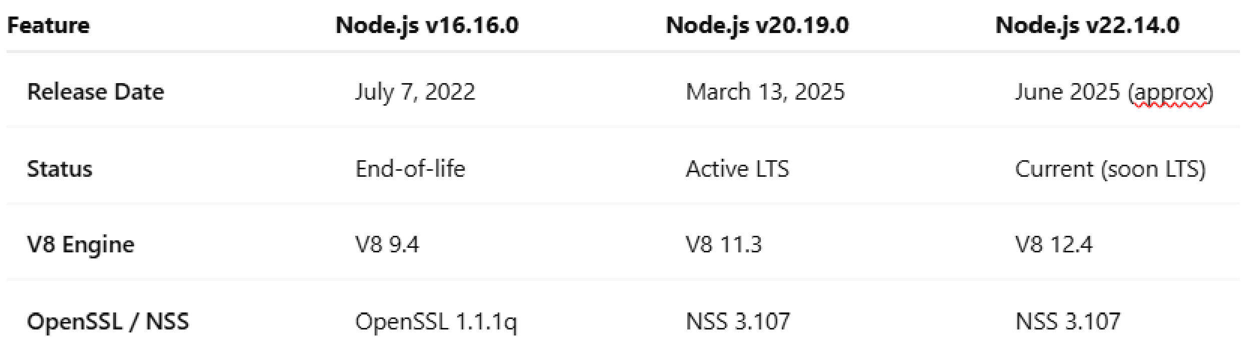
Node.js Version Comparison

This document compares three major Node.js versions 16.16.0, 20.19.0, and 22.14.0

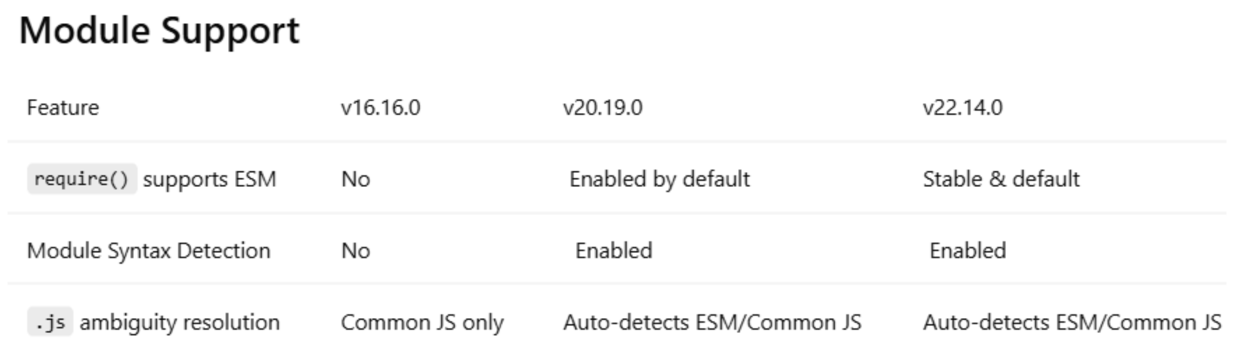
Node.js v16.16.0, once stable and widely adopted, is now at the end of its life cycle, meaning it no longer receives security updates.

Node.js 20.19.0 is an active LTS version and remains a reliable choice for production environments.

Node.js 22.14.0, the latest version, pushes boundaries with cutting-edge features, including advanced module support and extended TypeScript capabilities.



Feature	Node.js v16.16.0	Node.js v20.19.0	Node.js v22.14.0
Release Date	July 7, 2022	March 13, 2025	June 2025 (approx)
Status	End-of-life	Active LTS	Current (soon LTS)
V8 Engine	V8 9.4	V8 11.3	V8 12.4
OpenSSL / NSS	OpenSSL 1.1.1q	NSS 3.107	NSS 3.107



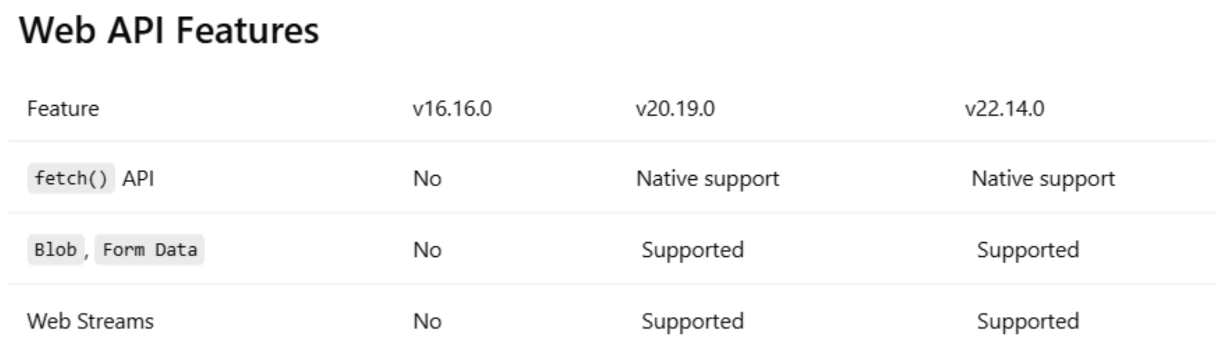
Module Support

Feature	v16.16.0	v20.19.0	v22.14.0
`require()` supports ESM	No	Enabled by default	Stable & default
Module Syntax Detection	No	Enabled	Enabled
`.js` ambiguity resolution	Common JS only	Auto-detects ESM/Common JS	Auto-detects ESM/Common JS

The biggest leap forward in module support has occurred between Node 16 and Node 20+.

Developers no longer need workarounds to use ES Modules with require() and ambiguous .js files are now resolved based on syntax rather than assumptions.

- Node 16 had limited support for ECMAScript Modules (ESM); developers had to use .mjs extensions or configure workarounds.
- Node 20 simplified ESM usage by auto-detecting module types and enabling better syntax detection.
- Node 22 builds on this by stabilizing ESM support and allowing seamless interoperability between CommonJS and ESM code.

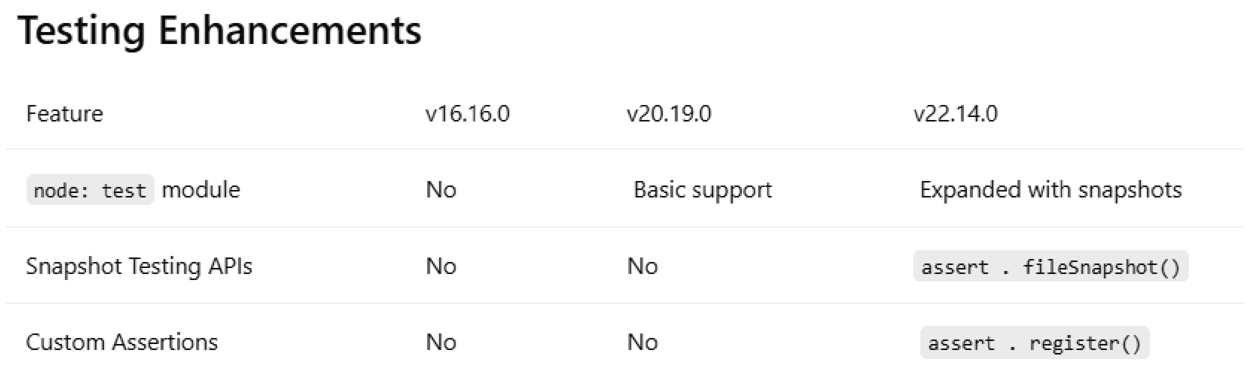


Web API Features

Feature	v16.16.0	v20.19.0	v22.14.0
`fetch()` API	No	Native support	Native support
`Blob`, `Form Data`	No	Supported	Supported
Web Streams	No	Supported	Supported

By Node 20, web APIs like fetch, Blob, and FormData became natively available, reducing the reliance on external libraries.

- Node 16 did not support these modern browser-like APIs, requiring polyfills or external libraries.
- Node 20 introduced native support for, fetch, Blob, and FormData , aligning more with browser environments.
- Node 22 continues this trend with complete support, reducing third-party dependency overhead.

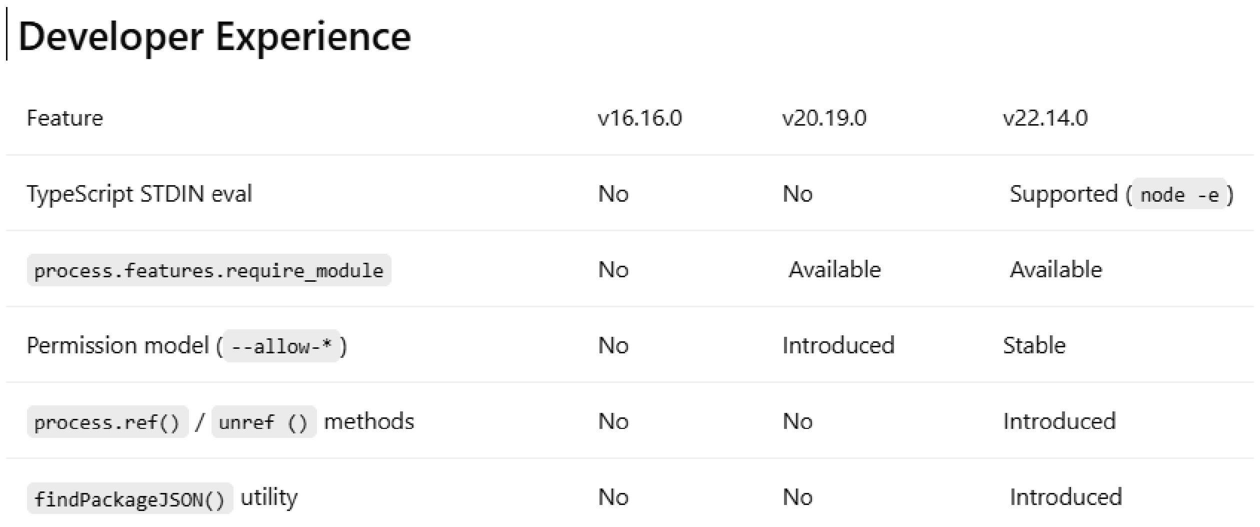


Testing Enhancements

Feature	v16.16.0	v20.19.0	v22.14.0
`node: test` module	No	Basic support	Expanded with snapshots
Snapshot Testing APIs	No	No	`assert . fileSnapshot()`
Custom Assertions	No	No	`assert . register()`

Testing in Node.js has seen major improvements since v20 making it easier for developers to write reliable and maintainable tests without third-party libraries.

- Node 16 lacked built-in testing features, pushing developers to use tools like Jest or Mocha.
- Node 20 introduced the node: test module, offering basic testing capabilities natively.
- Node 22 enhances this with snapshot testing and custom assertion registration, reducing the need for third-party test frameworks.

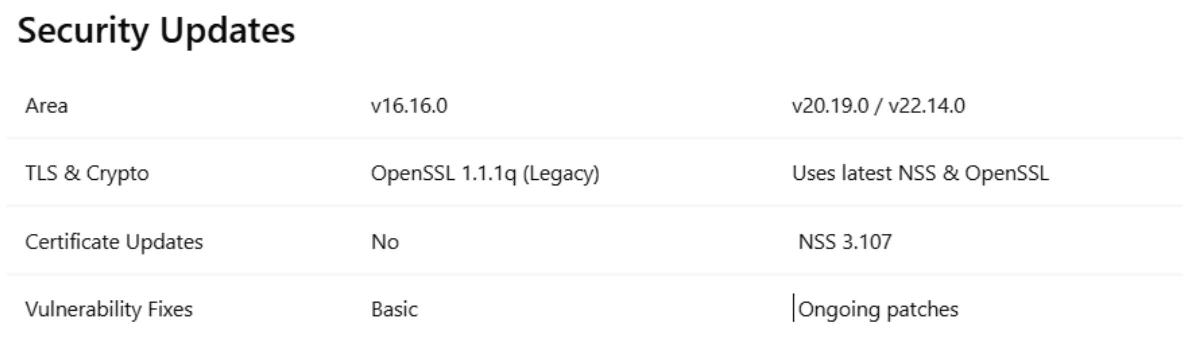


Developer Experience

Feature	v16.16.0	v20.19.0	v22.14.0
TypeScript STDIN eval	No	No	Supported (`node -e`)
`process.features.require_module`	No	Available	Available
Permission model (`--allow-*`)	No	Introduced	Stable
`process.ref()` / `unref ()` methods	No	No	Introduced
`findPackageJSON()` utility	No	No	Introduced

The developer experience in Node 22 is more refined.

- Node 16 offers a traditional, minimal experience.
- Node 20 introduced the experimental permission model (`--allow-*`), offering more control in security-sensitive applications.
- Node 22 polished the developer experience with utilities like `findPackageJSON()`, and built-in TypeScript support for quick evals — useful in CLI tooling.



Security Updates

Area	v16.16.0	v20.19.0 / v22.14.0
TLS & Crypto	OpenSSL 1.1.1q (Legacy)	Uses latest NSS & OpenSSL
Certificate Updates	No	NSS 3.107
Vulnerability Fixes	Basic	Ongoing patches

Node 16 relies on outdated cryptographic libraries, while Node 20 and 22 integrate with modern ones like NSS, ensuring better protection out of the box.

- Node 16 used older cryptographic libraries (OpenSSL 1.1.1q), which may have known vulnerabilities.
- Node 20 and 22 use NSS 3.107 and updated OpenSSL versions, ensuring better encryption, modern TLS support, and ongoing security patches.

Changes in Inbuilt Library

Area	v16.16.0	v20.19.0	v22.14.0
`fs` Promises	Limited	Stable & Extended	Enhanced with async hooks
`url`, `util`, `tty`	Legacy	Improved APIs	Further streamlined
`timers/promises`	No	Available	Expanded
`crypto`	Basic	Modern, WebCrypto APIs	WebCrypto+NSS

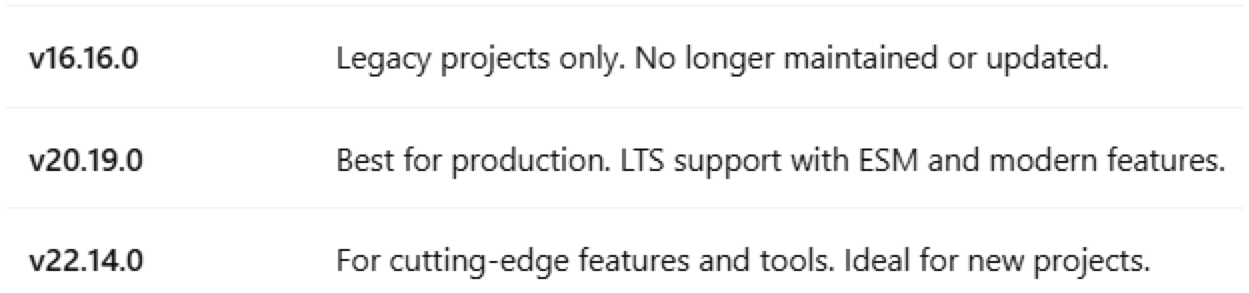
- Node 22 continues to modernize standard libraries to align with web standards.
- The `crypto` library integrates NSS for stronger cryptography.
- Libraries like `fs` and `timers/promises` have matured significantly.

Updates in HTTP Library and Protocol

Feature	v16.16.0	v20.19.0	v22.14.0
HTTP Keep-Alive improvements	No	Introduced	Optimized
Undici (`fetch()` backend)	No	Included	Default HTTP client
HTTP/2 stability	Partial	Stable	Improved diagnostics
HTTP/3 experimental	No	Experimental	More stable

- Node 22 treats `undici` as the default HTTP engine, enhancing performance.
- HTTP/3 support has seen better integration in Node 22.
- HTTP client libraries have evolved to support modern internet protocols.

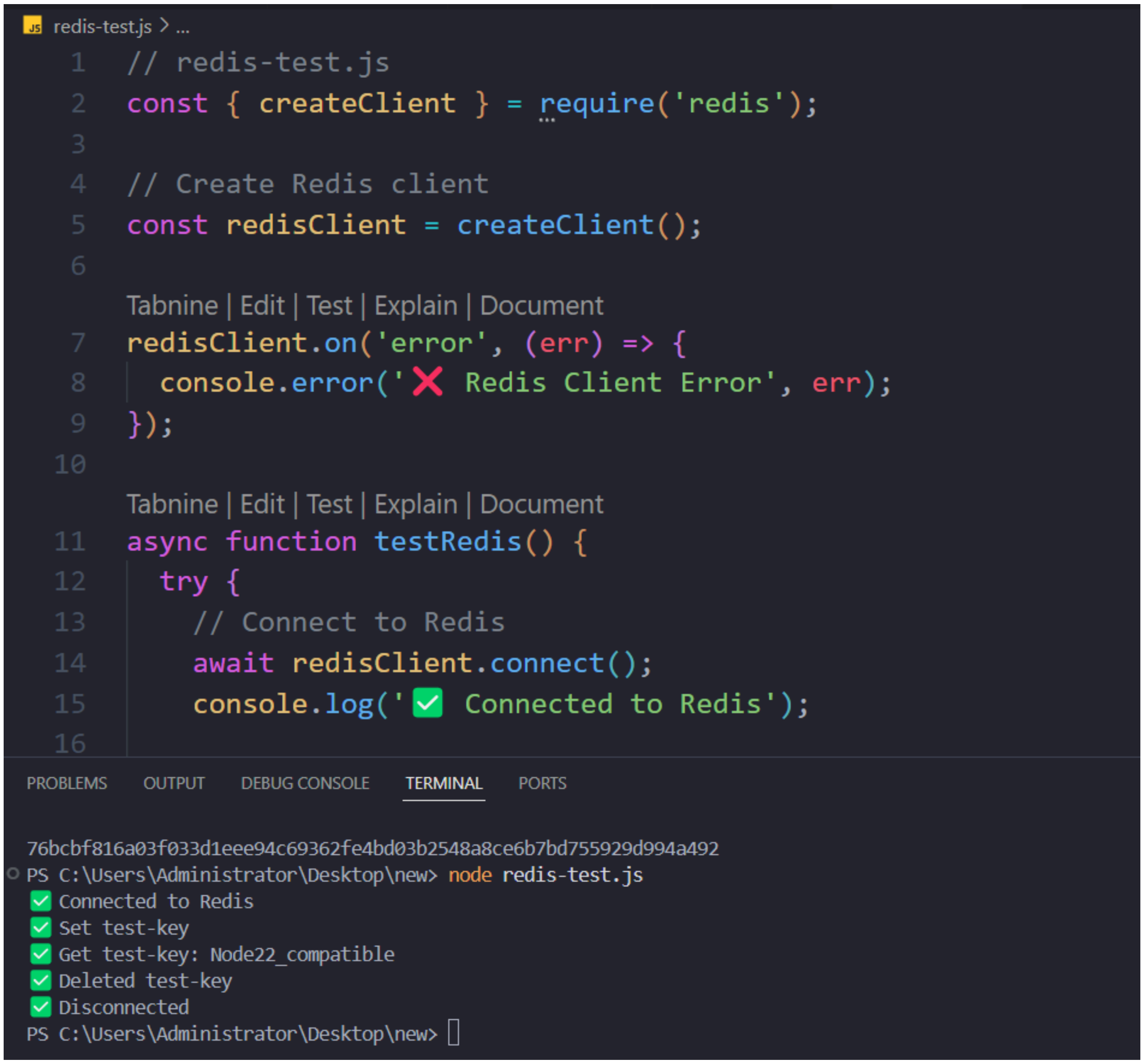
As an overview ,



v16.16.0	Legacy projects only. No longer maintained or updated.
v20.19.0	Best for production. LTS support with ESM and modern features.
v22.14.0	For cutting-edge features and tools. Ideal for new projects.

NODE V22

testing Redis with the official `redis@^4.6.7` package, which is compatible with **Node.js v22**.



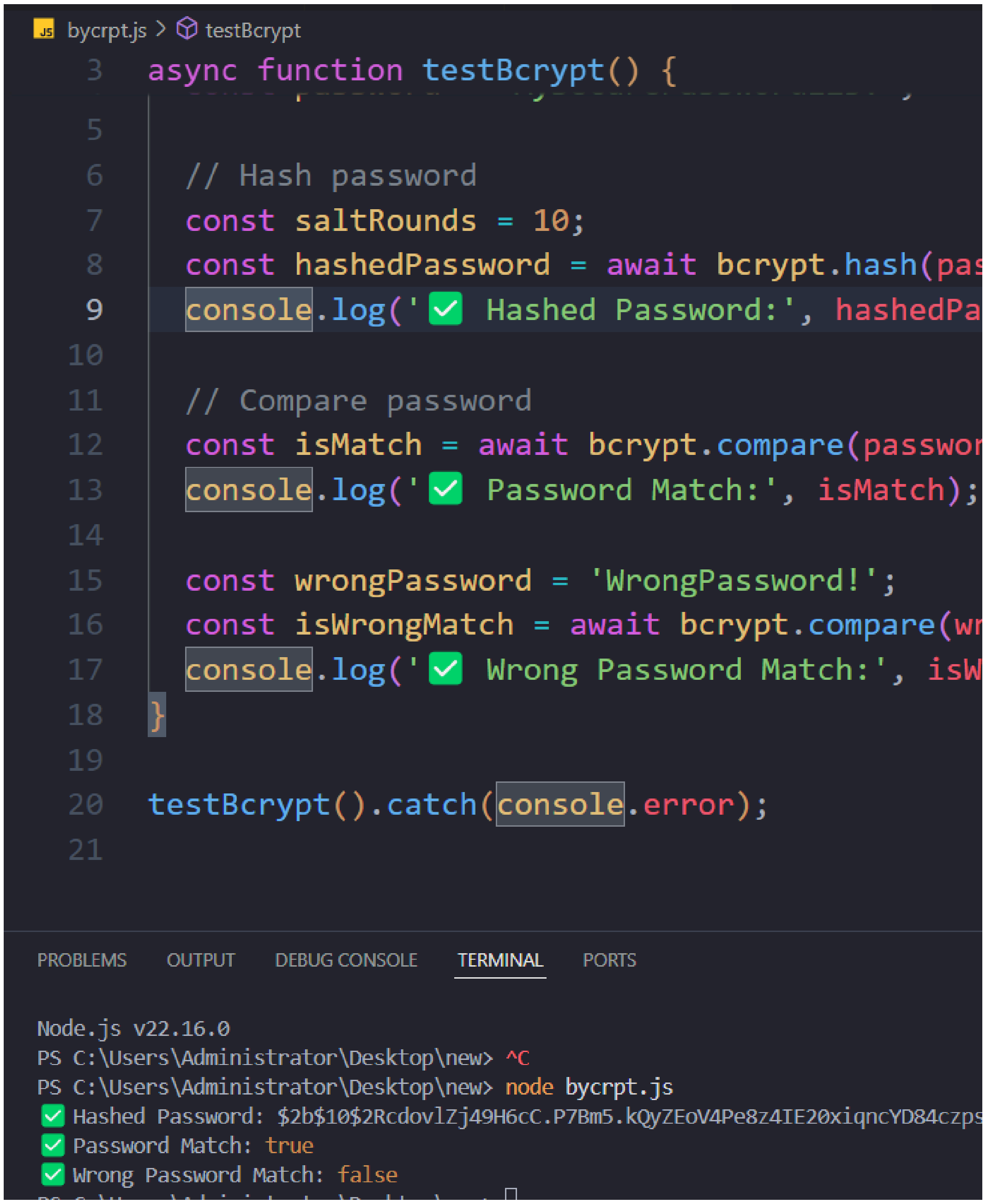
redis-test.js > ...

```
// redis-test.js
const { createClient } = require('redis');

// Create Redis client
const redisClient = createClient();

Tabnine | Edit | Test | Explain | Document
redisClient.on('error', (err) => {
  console.error('❌ Redis Client Error', err);
});

Tabnine | Edit | Test | Explain | Document
async function testRedis() {
  try {
    // Connect to Redis
    await redisClient.connect();
    console.log('✅ Connected to Redis');

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
76bcbf816a03f033d1eee94c69362fe4bd03b2548a8ce6b7bd755929d994a492
PS C:\Users\Administrator\Desktop\new> node redis-test.js
✅ Connected to Redis
✅ Set test-key
✅ Get test-key: Node22_compatible
✅ Deleted test-key
✅ Disconnected
PS C:\Users\Administrator\Desktop\new> 
```

migration from `bcrypt-nodejs` to `bcrypt`, which is secure, native, and actively maintained (and fully compatible with **Node.js v22**).



bycrpt.js > testBcrypt

```
async function testBcrypt() {
  [illegible]

  // Hash password
  const saltRounds = 10;
  const hashedPassword = await bcrypt.hash(pas
  console.log('✅ Hashed Password:', hashedPa

  // Compare password
  const isMatch = await bcrypt.compare(passwor
  console.log('✅ Password Match:', isMatch);

  const wrongPassword = 'WrongPassword!';
  const isWrongMatch = await bcrypt.compare(wr
  console.log('✅ Wrong Password Match:', isW
}

testBcrypt().catch(console.error);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

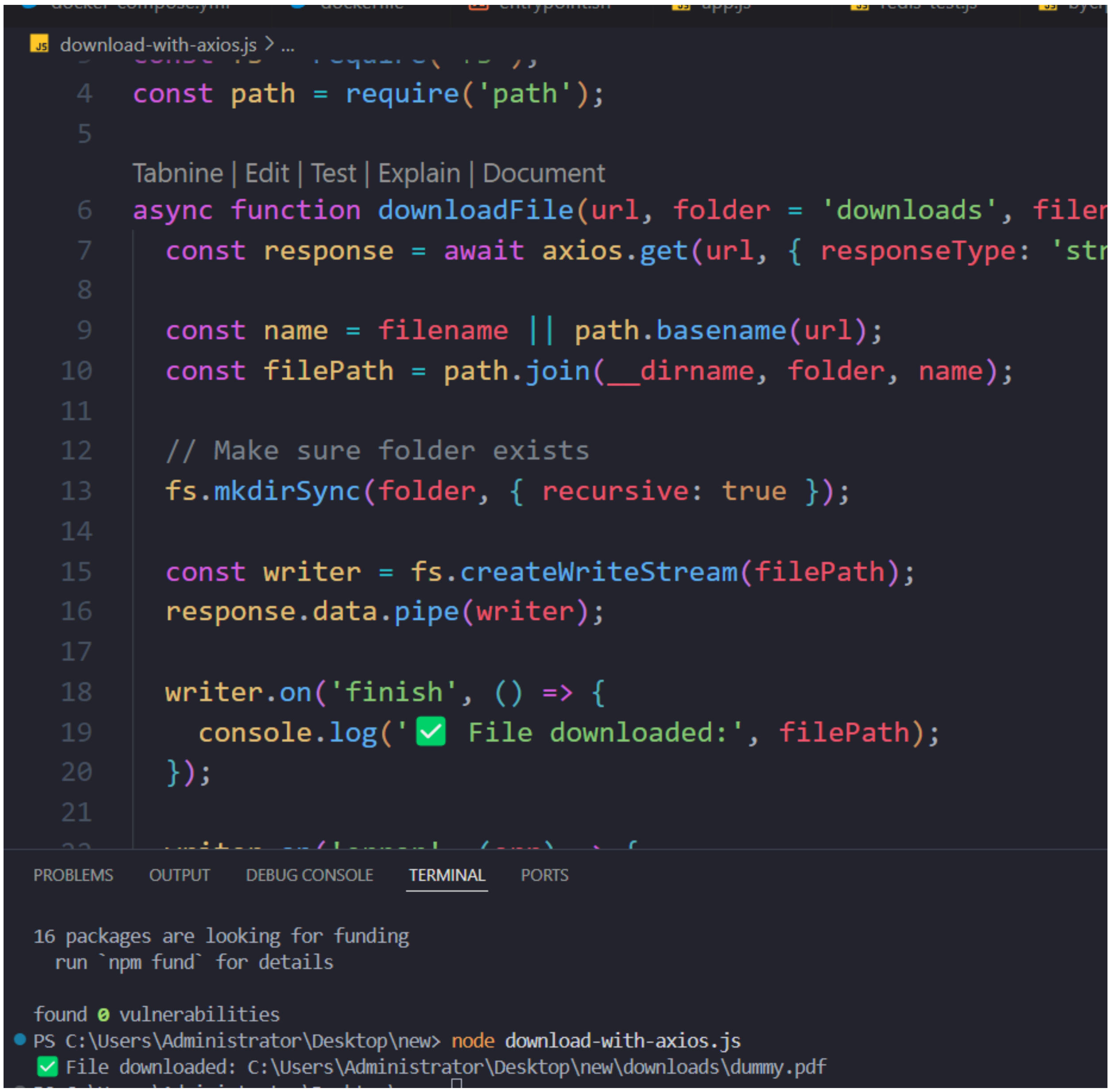
```
Node.js v22.16.0
PS C:\Users\Administrator\Desktop\new> ^C
PS C:\Users\Administrator\Desktop\new> node bycrpt.js
✅ Hashed Password: $2b$10$2RcdovlZj49H6cC.P7Bm5.kQyZEoV4Pe8z4IE20xiqncYD84czps
✅ Password Match: true
✅ Wrong Password Match: false
PS C:\Users\Administrator\Desktop\new>
```

Replaced deprecated `download` package

Used `axios` with `fs.createWriteStream()` to download a file

Saved the file to a local `downloads/` folder

Ran it successfully on **Node.js v22**

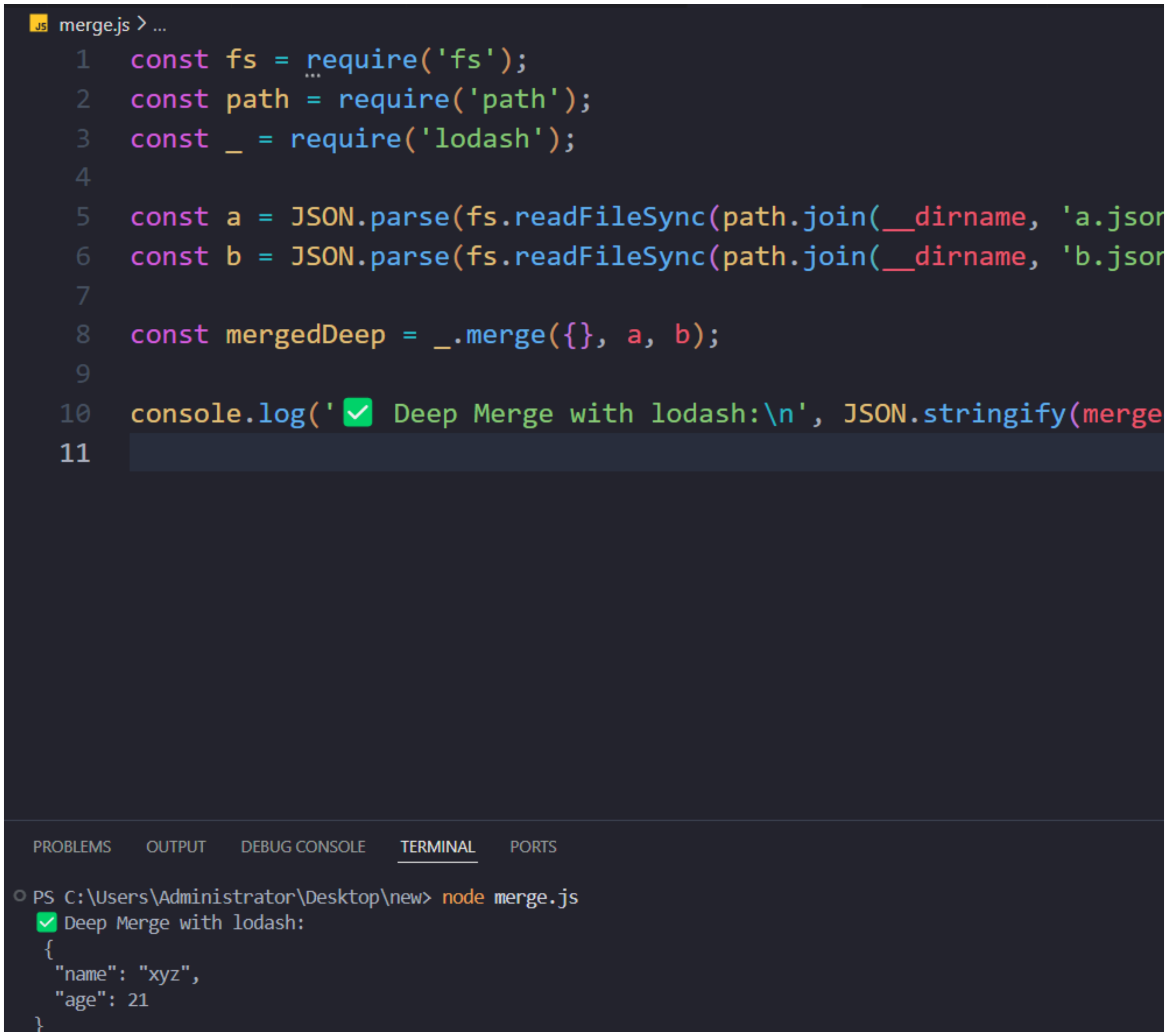


download-with-axios.js > ...

```
const path = require('path');

Tabnine | Edit | Test | Explain | Document
async function downloadFile(url, folder = 'downloads', filen
  const response = await axios.get(url, { responseType: 'str

  const name = filename || path.basename(url);
  const filePath = path.join(__dirname, folder, name);

  // Make sure folder exists
  fs.mkdirSync(folder, { recursive: true });

  const writer = fs.createWriteStream(filePath);
  response.data.pipe(writer);

  writer.on('finish', () => {
    console.log('✅ File downloaded:', filePath);
  });

```

```
16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Administrator\Desktop\new> node download-with-axios.js
✅ File downloaded: C:\Users\Administrator\Desktop\new\downloads\dummy.pdf
```

Use `lodash.merge` or native spread operator

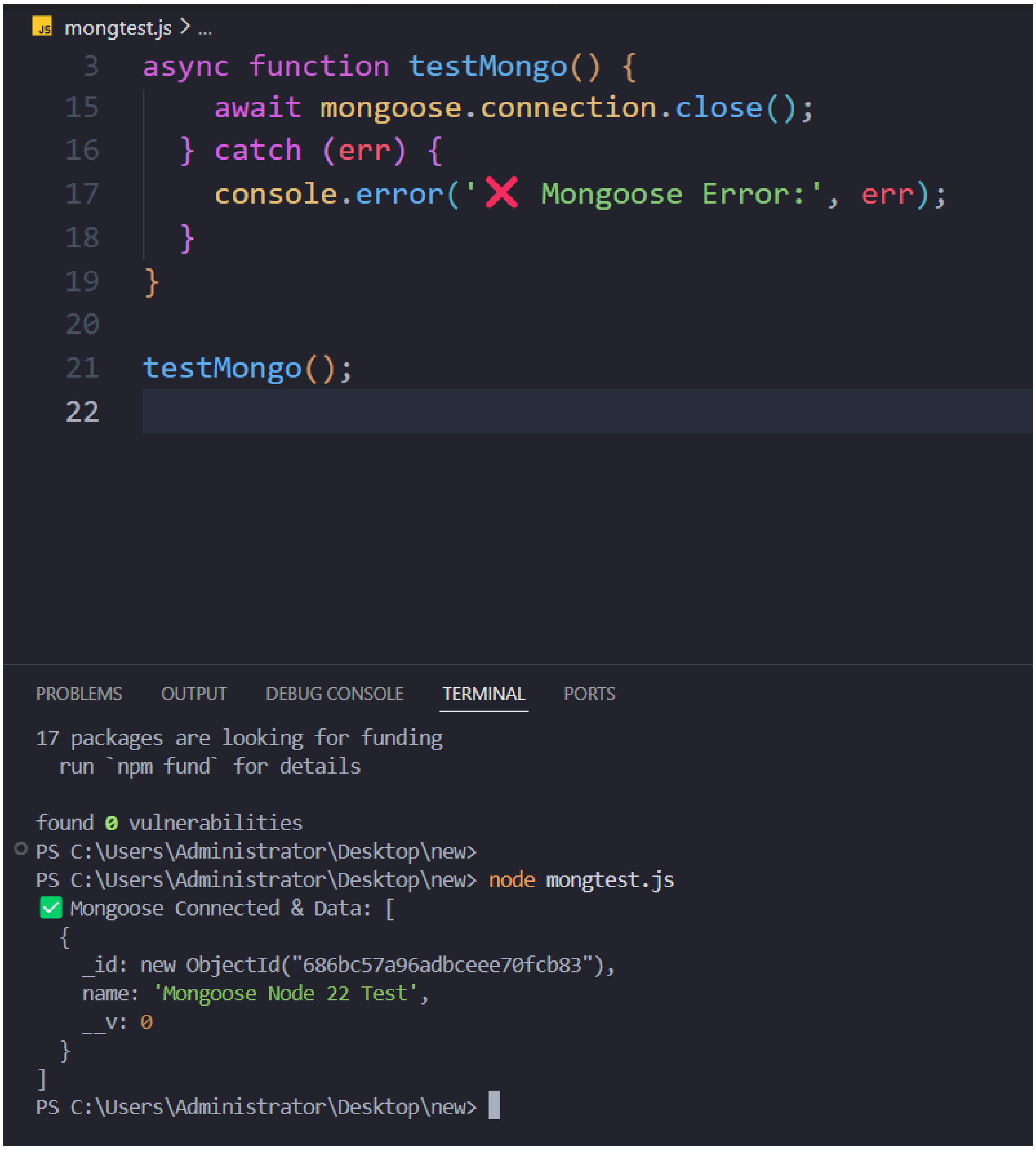


merge.js > ...

```
const fs = require('fs');
const path = require('path');
const _ = require('lodash');

const a = JSON.parse(fs.readFileSync(path.join(__dirname, 'a.jsor
const b = JSON.parse(fs.readFileSync(path.join(__dirname, 'b.jsor

const mergedDeep = _.merge({}, a, b);

console.log('✅ Deep Merge with lodash:\n', JSON.stringify(merge
```

```
PS C:\Users\Administrator\Desktop\new> node merge.js
✅ Deep Merge with lodash:
 {
  "name": "xyz",
  "age": 21
}
```

mongoose@^5.13.14 not capable with Node.js 22
So mongoose@^7.6.0.

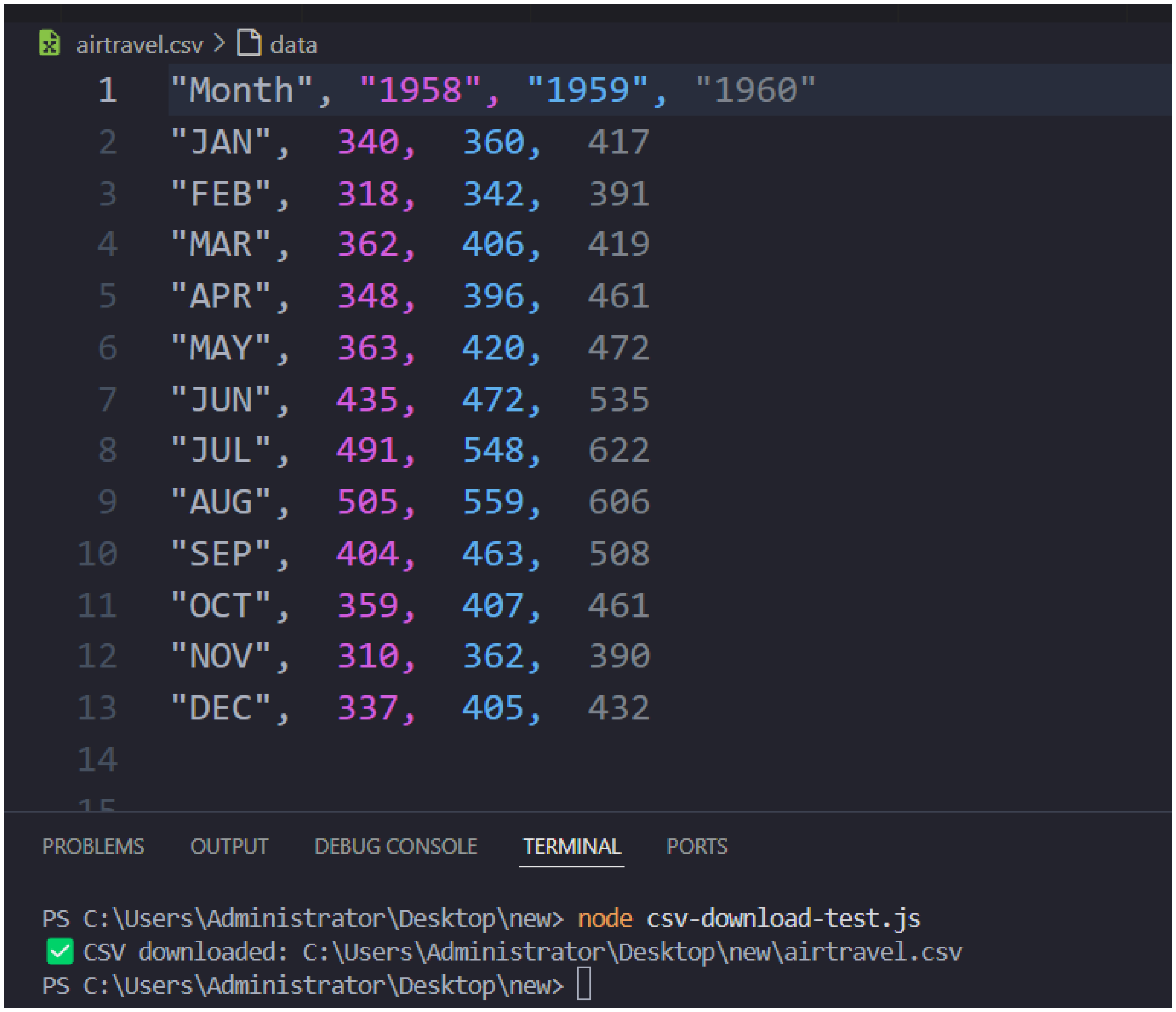

mongtest.js > ...

```
async function testMongo() {
    await mongoose.connection.close();
  } catch (err) {
    console.error('❌ Mongoose Error:', err);
  }
}

testMongo();

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Administrator\Desktop\new>
PS C:\Users\Administrator\Desktop\new> node mongtest.js
✅ Mongoose Connected & Data: [
  {
    _id: new ObjectId("686bc57a96adbceee70fcb83"),
    name: 'Mongoose Node 22 Test',
    __v: 0
  }
]
PS C:\Users\Administrator\Desktop\new>
```

Excel/CSV Download with `axios` (Node.js v22)

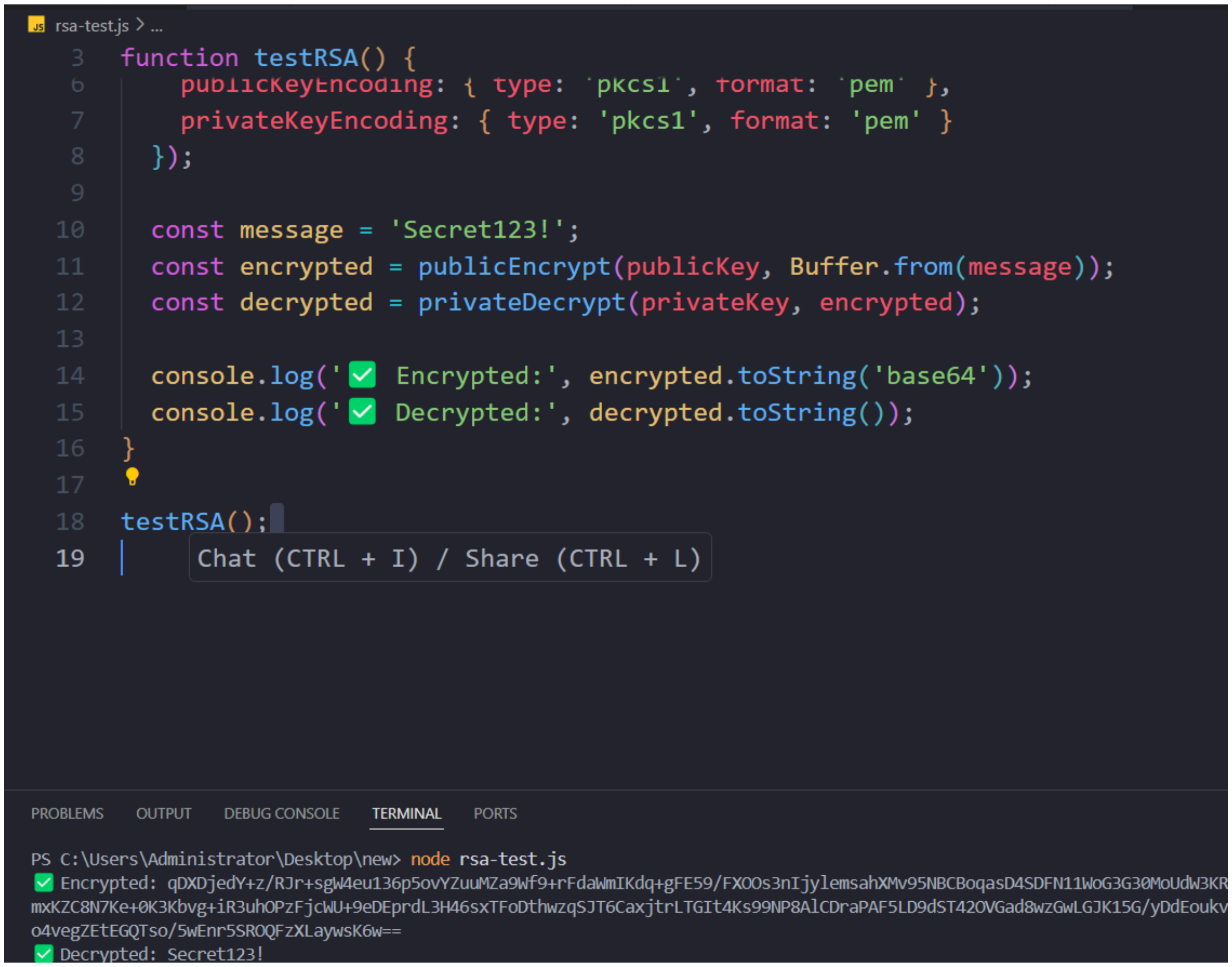


airtravel.csv > data

```
"Month", "1958", "1959", "1960"
"JAN",  340,  360,  417
"FEB",  318,  342,  391
"MAR",  362,  406,  419
"APR",  348,  396,  461
"MAY",  363,  420,  472
"JUN",  435,  472,  535
"JUL",  491,  548,  622
"AUG",  505,  559,  606
"SEP",  404,  463,  508
"OCT",  359,  407,  461
"NOV",  310,  362,  390
"DEC",  337,  405,  432
```

```
PS C:\Users\Administrator\Desktop\new> node csv-download-test.js
✅ CSV downloaded: C:\Users\Administrator\Desktop\new\airtravel.csv
PS C:\Users\Administrator\Desktop\new>
```

To **securely encrypt a message** using RSA (public-private key pair), then decrypt it.
Encrypts password using server's **public key, decrypt with its own private key** so that only the **intended receiver** can read it.

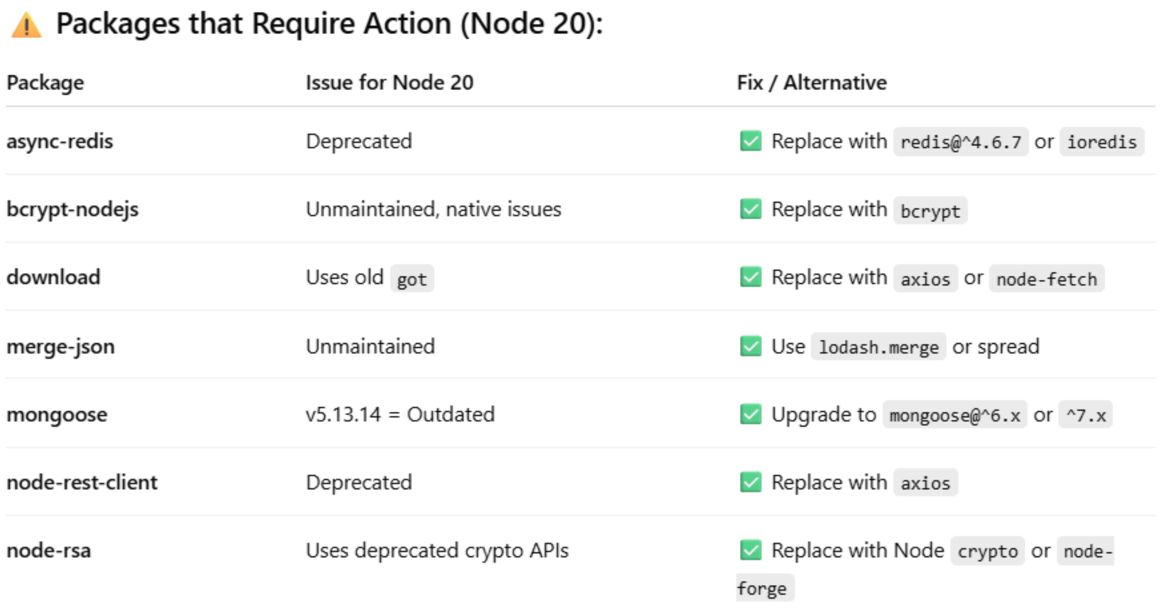


rsa-test.js > ...

```javascript
function testRSA() {
    publicKeyEncoding: { type: 'pkcs1', format: 'pem' },
    privateKeyEncoding: { type: 'pkcs1', format: 'pem' }
  });

  const message = 'Secret123!';
  const encrypted = publicEncrypt(publicKey, Buffer.from(message));
  const decrypted = privateDecrypt(privateKey, encrypted);

  console.log('✅ Encrypted:', encrypted.toString('base64'));
  console.log('✅ Decrypted:', decrypted.toString());
}

testRSA();
```

```
PS C:\Users\Administrator\Desktop\new> node rsa-test.js
✅ Encrypted: qDXDjedY+z/RJr+sgW4eu136p5ovYZuuMZa9Wf9+rFdaWmIKdq+gFE59/FXOOs3nIjylemsahXMv95NBCBoqasD4SDFN11WoG3G30MoUdW3KR
mxKZC8N7Ke+0K3Kbvg+iR3uhOPzFjcWU+9eDEprdL3H46sxTFoDthwzqSJT6CaxjtrLTGIt4Ks99NP8AlCDraPAF5LD9dST42OVGad8wzGwLGJK15G/yDdEoukv
o4vegZEtEGQTso/5wEnr5SROQFzXLaywsK6w==
✅ Decrypted: Secret123!
```

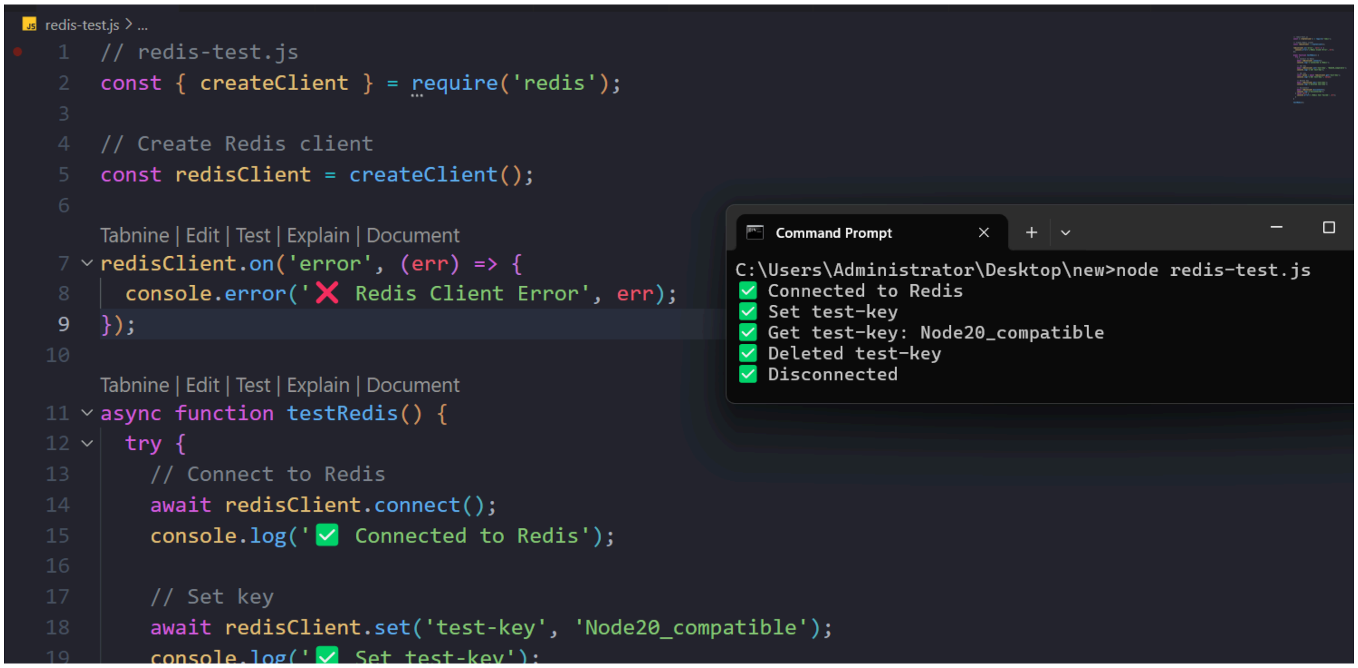
For node V20



⚠️ Packages that Require Action (Node 20):

Package	Issue for Node 20	Fix / Alternative
async-redis	Deprecated	✅ Replace with `redis@^4.6.7` or `ioredis`
bcrypt-nodejs	Unmaintained, native issues	✅ Replace with `bcrypt`
download	Uses old `got`	✅ Replace with `axios` or `node-fetch`
merge-json	Unmaintained	✅ Use `lodash.merge` or spread
mongoose	v5.13.14 = Outdated	✅ Upgrade to `mongoose@^6.x` or `^7.x`
node-rest-client	Deprecated	✅ Replace with `axios`
node-rsa	Uses deprecated crypto APIs	✅ Replace with Node `crypto` or `node-forge`

Replace `async-redis` → use `redis@^4.6.7`



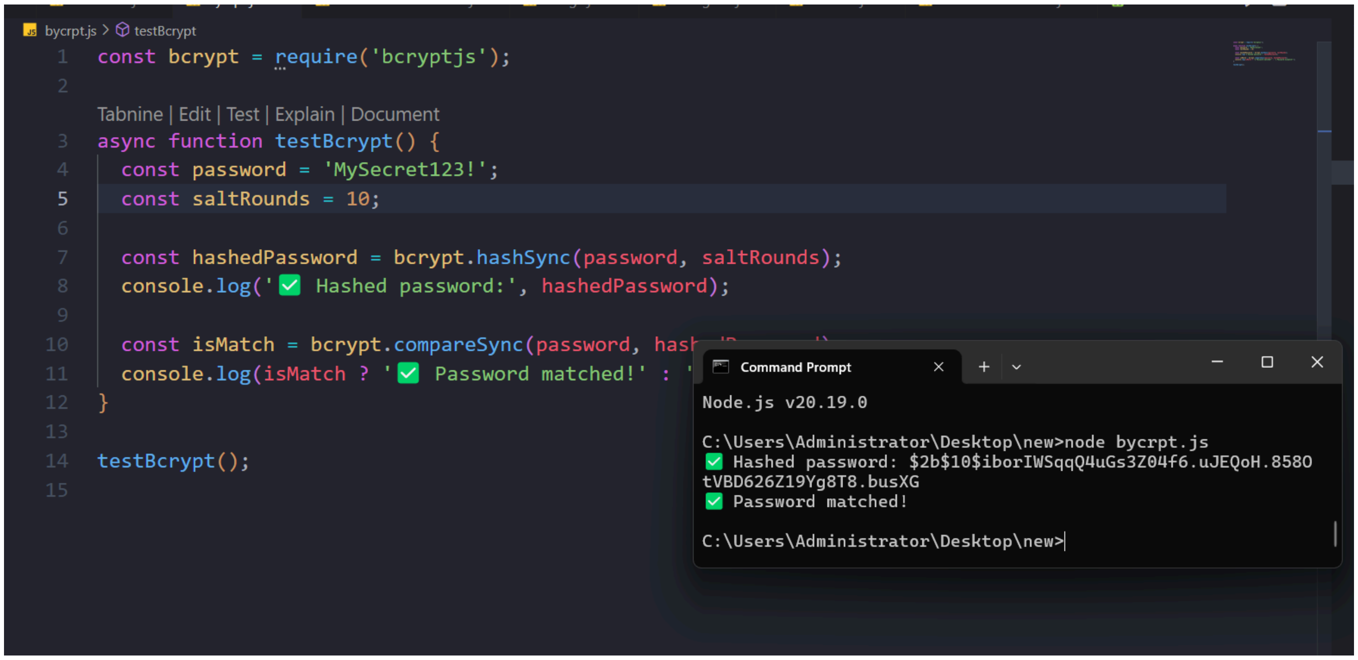
redis-test.js > ...

```
// redis-test.js
const { createClient } = require('redis');

// Create Redis client
const redisClient = createClient();

Tabnine | Edit | Test | Explain | Document
redisClient.on('error', (err) => {
  console.error('❌ Redis Client Error', err);
});

Tabnine | Edit | Test | Explain | Document
async function testRedis() {
  try {
    // Connect to Redis
    await redisClient.connect();
    console.log('✅ Connected to Redis');

    // Set key
    await redisClient.set('test-key', 'Node20_compatible');
    console.log('✅ Set test-key');
```

Command Prompt

```
C:\Users\Administrator\Desktop\new>node redis-test.js
✅ Connected to Redis
✅ Set test-key
✅ Get test-key: Node20_compatible
✅ Deleted test-key
✅ Disconnected
```

Replace `bcrypt-nodejs` → Use `bcrypt`



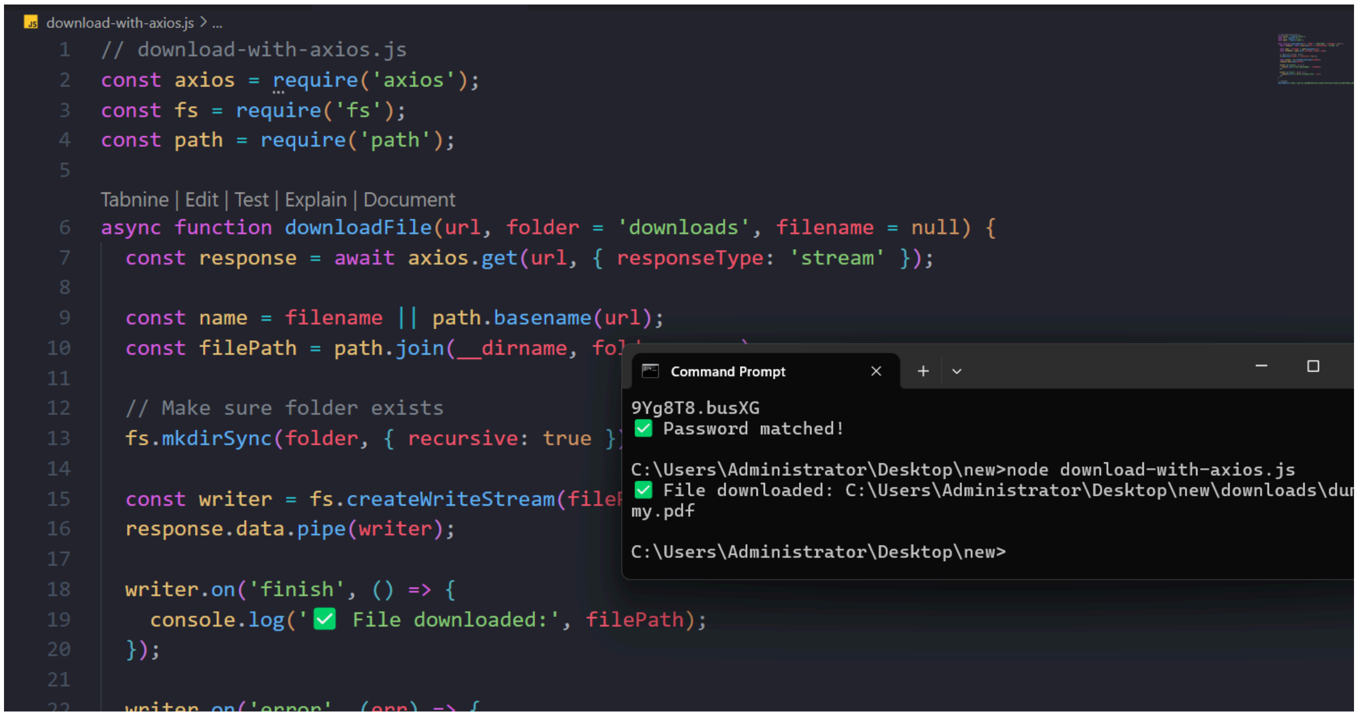
bycrpt.js > testBcrypt

```javascript
const bcrypt = require('bcryptjs');

Tabnine | Edit | Test | Explain | Document
async function testBcrypt() {
  const password = 'MySecret123!';
  const saltRounds = 10;

  const hashedPassword = bcrypt.hashSync(password, saltRounds);
  console.log('✅ Hashed password:', hashedPassword);

  const isMatch = bcrypt.compareSync(password, hash
  console.log(isMatch ? '✅ Password matched!' : '
}

testBcrypt();
```

Command Prompt

```
Node.js v20.19.0

C:\Users\Administrator\Desktop\new>node bycrpt.js
✅ Hashed password: $2b$10$iborIWSqqQ4uGs3Z04f6.uJEQoH.8580
tVBD626Z19Yg8T8.busXG
✅ Password matched!

C:\Users\Administrator\Desktop\new>
```

Replace `download` → with `axios`



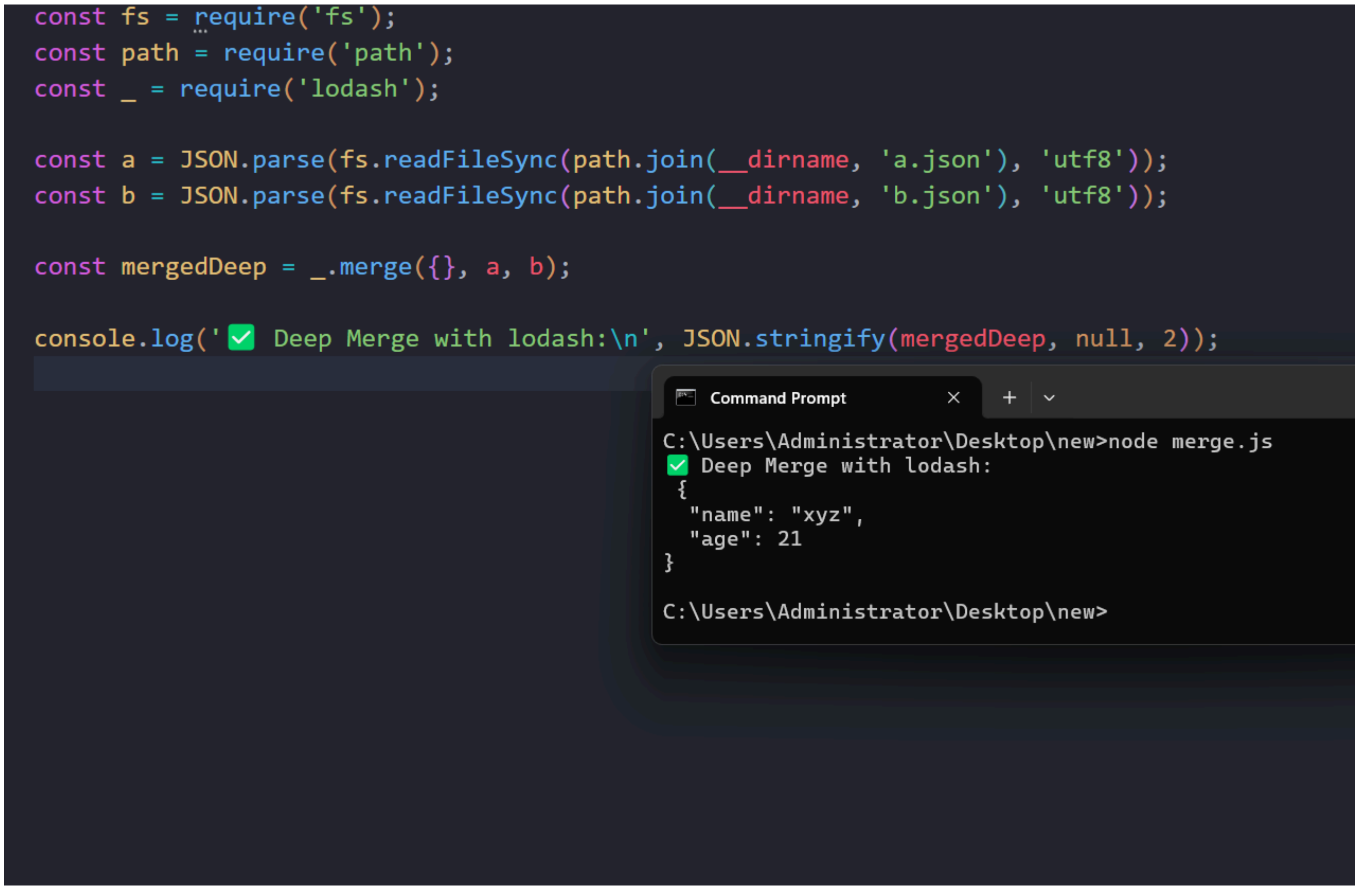
download-with-axios.js > ...

```
// download-with-axios.js
const axios = require('axios');
const fs = require('fs');
const path = require('path');

Tabnine | Edit | Test | Explain | Document
async function downloadFile(url, folder = 'downloads', filename = null) {
  const response = await axios.get(url, { responseType: 'stream' });

  const name = filename || path.basename(url);
  const filePath = path.join(__dirname, fol

  // Make sure folder exists
  fs.mkdirSync(folder, { recursive: true }

  const writer = fs.createWriteStream(fileP
  response.data.pipe(writer);

  writer.on('finish', () => {
    console.log('✅ File downloaded:', filePath);
  });

  writer.on('error', (err) => {
```

Command Prompt

```
9Yg8T8.busXG
✅ Password matched!

C:\Users\Administrator\Desktop\new>node download-with-axios.js
✅ File downloaded: C:\Users\Administrator\Desktop\new\downloads\dur
my.pdf

C:\Users\Administrator\Desktop\new>
```

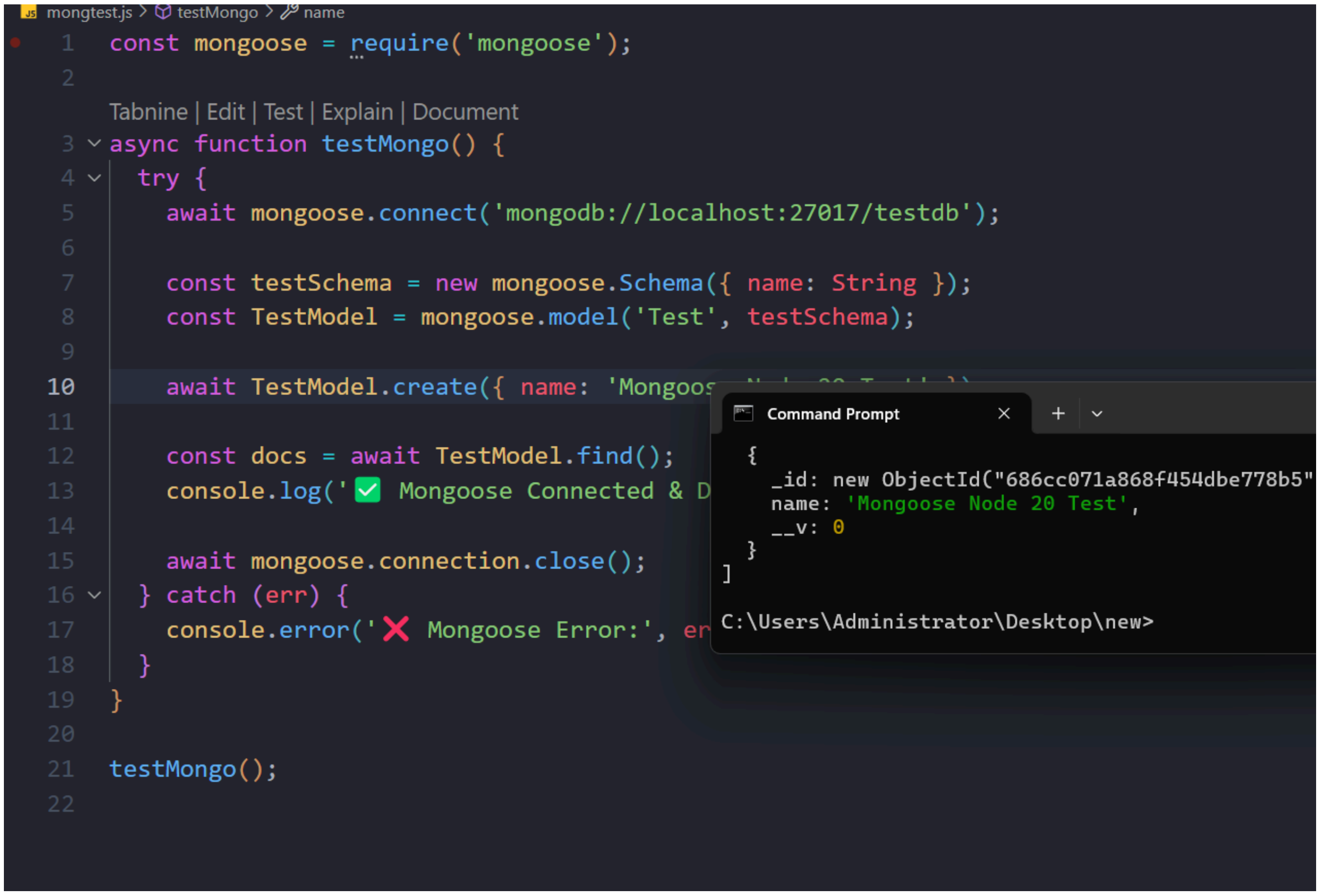
merge-json → Use lodash.merge



```
const fs = require('fs');
const path = require('path');
const _ = require('lodash');

const a = JSON.parse(fs.readFileSync(path.join(__dirname, 'a.json'), 'utf8'));
const b = JSON.parse(fs.readFileSync(path.join(__dirname, 'b.json'), 'utf8'));

const mergedDeep = _.merge({}, a, b);

console.log('✅ Deep Merge with lodash:\n', JSON.stringify(mergedDeep, null, 2));
```

```
C:\Users\Administrator\Desktop\new>node merge.js
✅ Deep Merge with lodash:
 {
  "name": "xyz",
  "age": 21
}

C:\Users\Administrator\Desktop\new>
```

the compatibility of **Mongoose v7** with **Node.js v20.19.0**.



mongtest.js > testMongo > name

```javascript
const mongoose = require('mongoose');

Tabnine | Edit | Test | Explain | Document
async function testMongo() {
  try {
    await mongoose.connect('mongodb://localhost:27017/testdb');

    const testSchema = new mongoose.Schema({ name: String });
    const TestModel = mongoose.model('Test', testSchema);

    await TestModel.create({ name: 'Mongoos

    const docs = await TestModel.find();
    console.log('✅ Mongoose Connected & D

    await mongoose.connection.close();
  } catch (err) {
    console.error('❌ Mongoose Error:', er
  }
}

testMongo();
```

Command Prompt

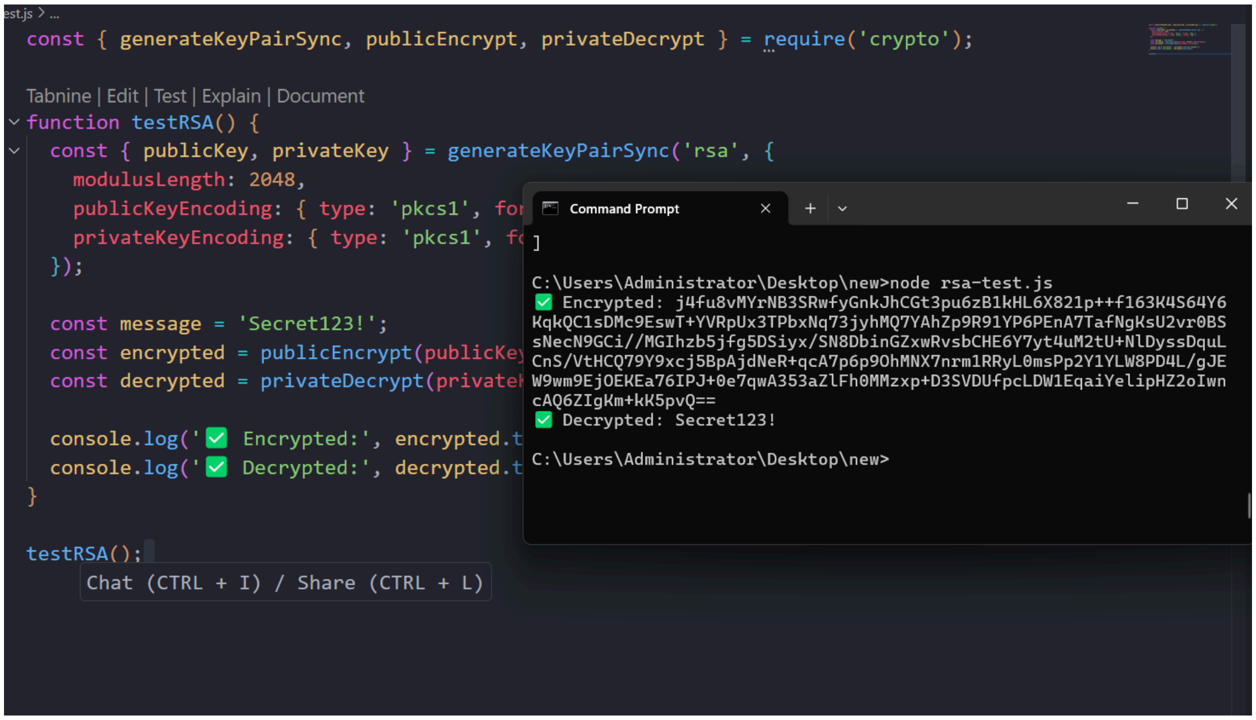
```
  {
    _id: new ObjectId("686cc071a868f454dbe778b5"
    name: 'Mongoose Node 20 Test',
    __v: 0
  }
]

C:\Users\Administrator\Desktop\new>
```

-

-

Replace `node-rsa` → Use Node.js built-in `crypto` module

.

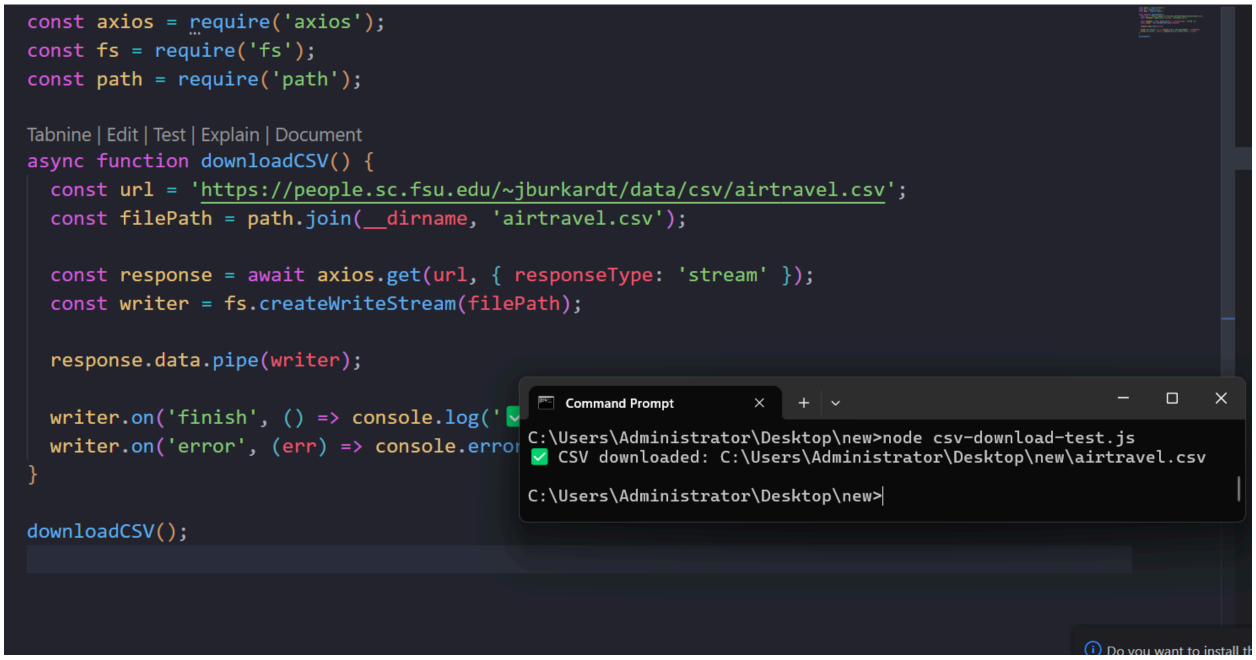


est.js > ...
Tabnine | Edit | Test | Explain | Document
Chat (CTRL + I) / Share (CTRL + L)
Command Prompt

```
const { generateKeyPairSync, publicEncrypt, privateDecrypt } = require('crypto');

function testRSA() {
  const { publicKey, privateKey } = generateKeyPairSync('rsa', {
    modulusLength: 2048,
    publicKeyEncoding: { type: 'pkcs1', for
    privateKeyEncoding: { type: 'pkcs1', fo
  });

  const message = 'Secret123!';
  const encrypted = publicEncrypt(publicKey
  const decrypted = privateDecrypt(privateK

  console.log('✅ Encrypted:', encrypted.t
  console.log('✅ Decrypted:', decrypted.t
}

testRSA();
```

```
]

C:\Users\Administrator\Desktop\new>node rsa-test.js
✅ Encrypted: j4fu8vMYrNB3SRwfyGnkJhCGt3pu6zB1kHL6X821p++f163K4S64Y6
KqkQC1sDMc9EswT+YVRpUx3TPbxNq73jyhMQ7YAhZp9R91YP6PEnA7TafNgKsU2vr0BS
sNecN9GCi//MGIhzb5jfg5DSiyx/SN8DbinGZxwRvsbCHE6Y7yt4uM2tU+NlDyssDquL
CnS/VtHCQ79Y9xcj5BpAjdNeR+qcA7p6p9OhMNX7nrm1RRyL0msPp2Y1YLW8PD4L/gJE
W9wm9EjOEKEa76IPJ+0e7qwA353aZlFh0MMzxp+D3SVDUfpcLDW1EqaiYelipHZ2oIwn
cAQ6ZIgKm+kK5pvQ==
✅ Decrypted: Secret123!

C:\Users\Administrator\Desktop\new>
```

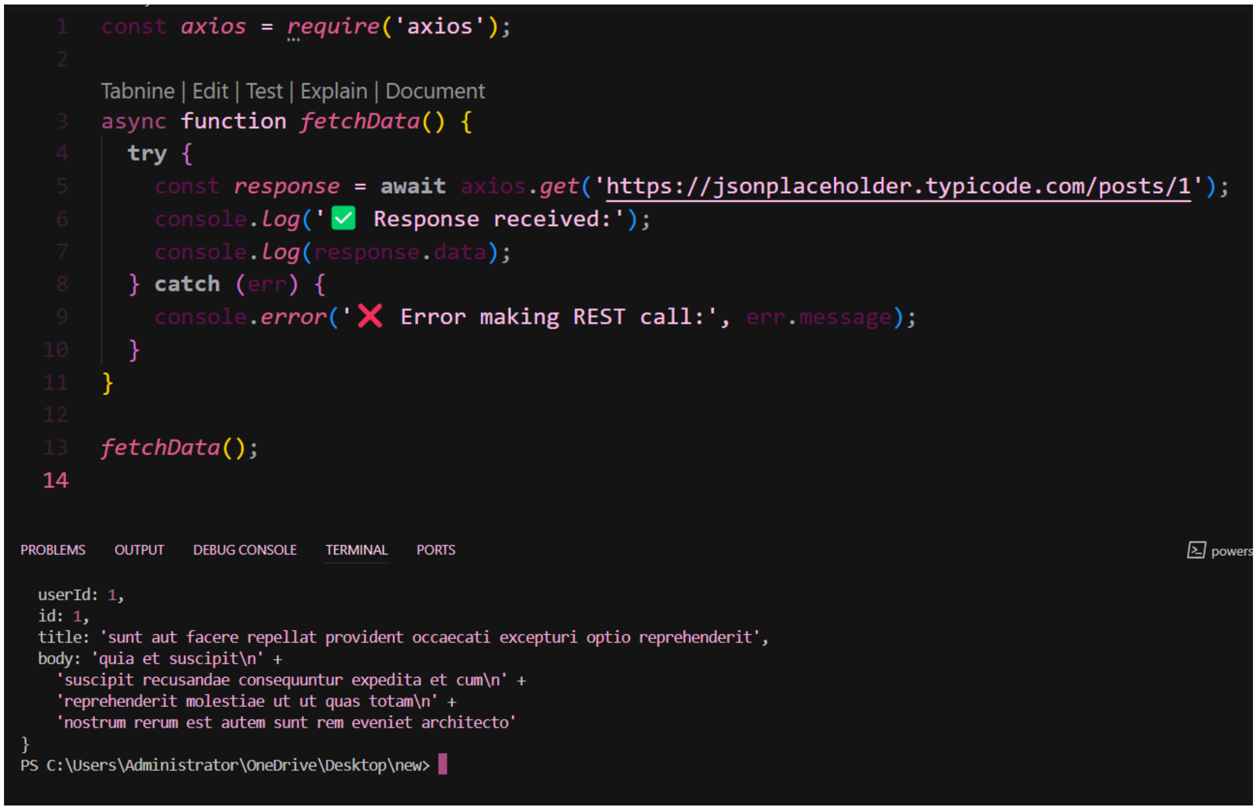
Excel/CSV Downloads – Test Compatibility on Node.js v20.19.0



```
const axios = require('axios');
const fs = require('fs');
const path = require('path');

Tabnine | Edit | Test | Explain | Document
async function downloadCSV() {
  const url = 'https://people.sc.fsu.edu/~jburkardt/data/csv/airtravel.csv';
  const filePath = path.join(__dirname, 'airtravel.csv');

  const response = await axios.get(url, { responseType: 'stream' });
  const writer = fs.createWriteStream(filePath);

  response.data.pipe(writer);

  writer.on('finish', () => console.log('✅
  writer.on('error', (err) => console.error
}

downloadCSV();
```

```
C:\Users\Administrator\Desktop\new>node csv-download-test.js
✅ CSV downloaded: C:\Users\Administrator\Desktop\new\airtravel.csv

C:\Users\Administrator\Desktop\new>
```

node-rest-client → Use axios



```
const axios = require('axios');

Tabnine | Edit | Test | Explain | Document
async function fetchData() {
  try {
    const response = await axios.get('https://jsonplaceholder.typicode.com/posts/1');
    console.log('✅ Response received:');
    console.log(response.data);
  } catch (err) {
    console.error('❌ Error making REST call:', err.message);
  }
}

fetchData();
```

```
  userId: 1,
  id: 1,
  title: 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit',
  body: 'quia et suscipit\n' +
    'suscipit recusandae consequuntur expedita et cum\n' +
    'reprehenderit molestiae ut ut quas totam\n' +
    'nostrum rerum est autem sunt rem eveniet architecto'
}
PS C:\Users\Administrator\OneDrive\Desktop\new>
```

Redis (node-redis v5)

quit() → close() and disconnect() → destroy()

From the official source.md documentation:

> The QUIT command has been deprecated in Redis 7.2...
> client.QUIT/quit() is replaced by client.close() and client.disconnect() has been renamed to client.destroy()

https://raw.githubusercontent.com/redis/node-redis/master/README.md

https://www.npmjs.com/package/redis

Misbehavior of `disconnect()`

GitHub issue #2719 explains how disconnect() currently flushes pending commands, contrary to docs:

> *"…disconnect should close the connection without flushing and quit close with flushing… Current implementation is not in sync…"*

https://github.com/redis/node-redis/issues/2719

Mongoose (v7+ → v8)

1. Deprecation of `rawResult` → `includeResultMetadata`

> *"Removed rawResult option … replaced by includeResultMetadata"*

https://mongoosejs.com/docs/migrating_to_8.html

2. Removal of `findOneAndRemove()` and `findByIdAndRemove()`

Migration guide states:

> *"Removed findOneAndRemove(). … Please use findOneAndDelete() instead."*

https://mongoosejs.com/docs/migrating_to_8.html

3. Removal of `count()` in favor of `countDocuments()`/`estimatedDocumentCount()`

Migration docs:

> *"Removed count() … Use Model.countDocuments() and Query.prototype.countDocuments()"*

https://mongoosejs.com/docs/migrating_to_8.html

4. **Deprecation warning for `findOneAndUpdate()` without `useFindAndModify`**

🔧 GitHub issue details:

> *“DeprecationWarning: Mongoose: findOneAndUpdate() and findOneAndDelete() without the useFindAndModify option set to false are deprecated.”*

https://github.com/Automattic/mongoose/issues/9550

Changed Feature	Original API	New API
`rawResult` option	`{ rawResult: true }`	`{ includeResultMetadata: true }`
`findOneAndRemove()`	`Model.findOneAndRemove()`	`Model.findOneAndDelete()`
`findByIdAndRemove()`	`Model.findByIdAndRemove()`	`Model.findByIdAndDelete()`
`count()`	`Model.count()`	`Model.countDocuments()`
`findOneAndUpdate()` warning	no `useFindAndModify:false`	add global setting or use new APIs

`Bcrypt`

1.Native `bcrypt` install errors

Developers frequently face build failures with `bcrypt`, especially on Windows or environments lacking C++ compilers.

https://stackoverflow.com/questions/34546272/cannot-find-module-bcrypt

2. GitHub Issue on `bcrypt.js`: Pure JS alternative

> *“ used 'bcryptjs' module instead of 'bcrypt' module”*
>
> https://github.com/dcodeIO/bcrypt.js/issues/112

Lodash Merge

`lodash.merge` remains fully compatible and reliable for use with Node.js v20

https://github.com/lodash/lodash/wiki/Changelog

No breaking changes or deprecations impacting Node.js v20

Axios

Widely adopted for REST calls/files downloads without any breaking changes

https://github.com/redis/ioredis

`merge-json@0.1.0-b.3`

Latest published in 2015, **no active maintenance or migration** for Node 20

https://www.npmjs.com/package/json-merger

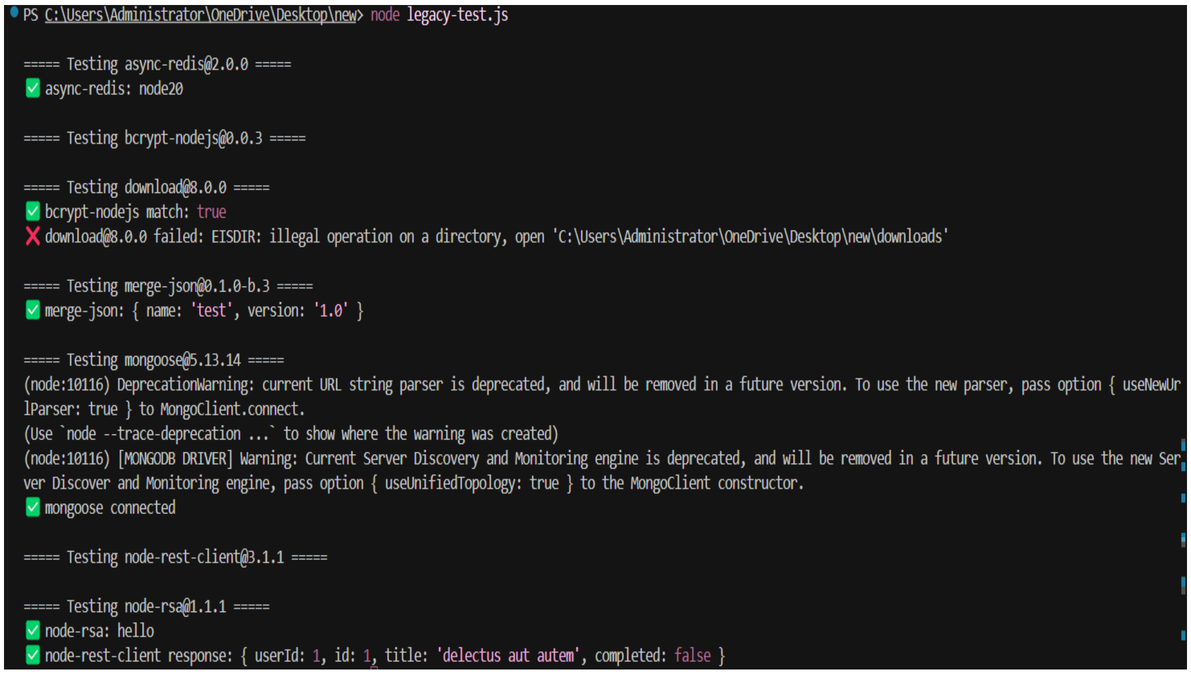
https://github.com/Khezen/mergejson

Stable Package Versions for Node.js v20

Package Name	Recommended Version	Replaces / Reason
redis	^4.6.7	Replaces async-redis (deprecated)
bcrypt	^5.1.1	Replaces bcrypt-nodejs (unmaintained)
axios	^1.6.8	Replaces download, node-rest-client
node-fetch *(opt)*	^3.3.2	Alternative to axios
lodash.merge	^4.6.2	Replaces merge-json
mongoose	^7.6.2 or ^8.0.0	Upgrade from v5.13.14 (outdated)
node-forge *(opt)*	^1.3.1	Replaces node-rsa (deprecated crypto API)
node:crypto	*Built-in* (Node v20+)	Native alternative to node-rsa

Working Without Issues:

- async-redis@2.0.0 → No error
- bcrypt-nodejs@0.0.3 → Works fine, deprecated but functional
- merge-json@0.1.0-b.3 → Works as expected
- node-rest-client@3.1.1 → Works, but deprecated
- node-rsa@1.1.1 → Works



```
PS C:\Users\Administrator\OneDrive\Desktop\new> node legacy-test.js

===== Testing async-redis@2.0.0 =====
✅ async-redis: node20

===== Testing bcrypt-nodejs@0.0.3 =====

===== Testing download@8.0.0 =====
✅ bcrypt-nodejs match: true
❌ download@8.0.0 failed: EISDIR: illegal operation on a directory, open 'C:\Users\Administrator\OneDrive\Desktop\new\downloads'

===== Testing merge-json@0.1.0-b.3 =====
✅ merge-json: { name: 'test', version: '1.0' }

===== Testing mongoose@5.13.14 =====
(node:10116) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUr
lParser: true } to MongoClient.connect.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:10116) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Ser
ver Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
✅ mongoose connected

===== Testing node-rest-client@3.1.1 =====

===== Testing node-rsa@1.1.1 =====
✅ node-rsa: hello
✅ node-rest-client response: { userId: 1, id: 1, title: 'delectus aut autem', completed: false }
```

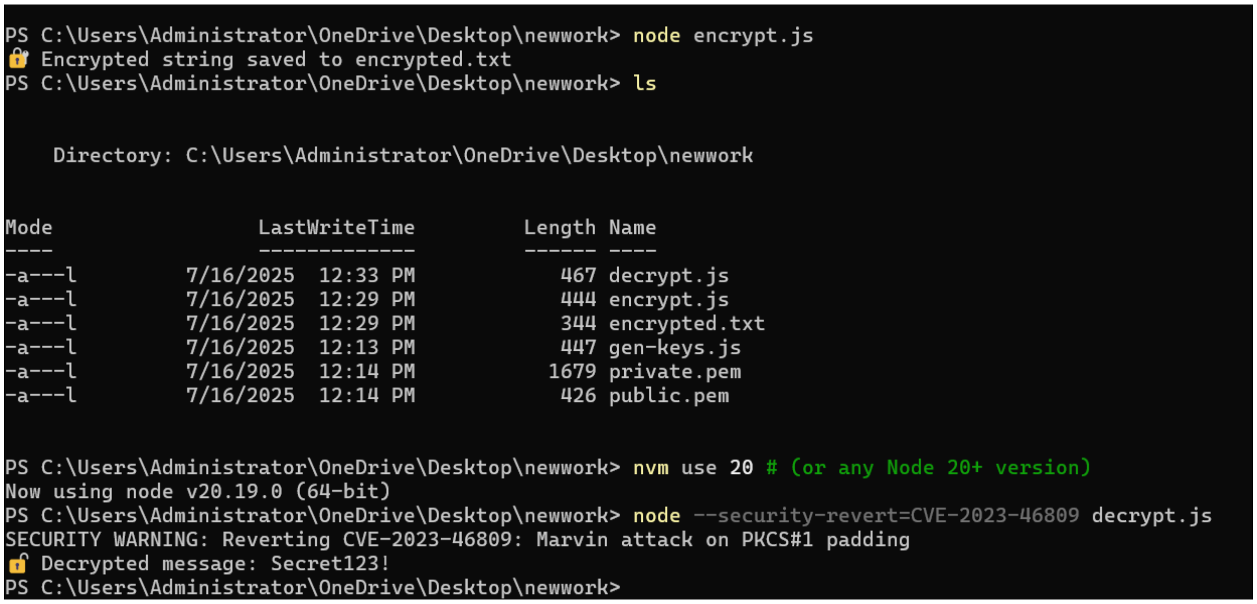
Step 1
Generate RSA Key Pair
Step 2
Encrypt Using Node.js v16
Step 3
Decrypt Using Node.js v20+ (With Revert Flag)
nvm use 20.19.0
node --security-revert=CVE-2023-46809 decrypt.js

Here , I performed secure RSA encryption using Node.js v16 and then decrypted it using Node.js v20. Due to Node's recent security upgrade blocking legacy PKCS#1 v1.5 padding (RSA_PKCS1_PADDING), I used the officially supported flag --security-revert=CVE-2023-46809 to temporarily allow decryption
https://stackoverflow.com/questions/78306265/encryption-decryption-with-rsa-pkcs1-padding-algorithm-in-express-js

https://nodejs.org/en/blog/release/v21.6.2

https://gitea.szsolutions.ch/Mirror/node-ebics-client/commit/aa86eaaffe3ed265307ac5170530601508ac8740

Long-term solution:
Migrate to **`RSA_PKCS1_OAEP_PADDING`** which remains secure and fully supported.



```
PS C:\Users\Administrator\OneDrive\Desktop\newwork> node encrypt.js
🔐 Encrypted string saved to encrypted.txt
PS C:\Users\Administrator\OneDrive\Desktop\newwork> ls


    Directory: C:\Users\Administrator\OneDrive\Desktop\newwork


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a---l         7/16/2025  12:33 PM            467 decrypt.js
-a---l         7/16/2025  12:29 PM            444 encrypt.js
-a---l         7/16/2025  12:29 PM            344 encrypted.txt
-a---l         7/16/2025  12:13 PM            447 gen-keys.js
-a---l         7/16/2025  12:14 PM           1679 private.pem
-a---l         7/16/2025  12:14 PM            426 public.pem


PS C:\Users\Administrator\OneDrive\Desktop\newwork> nvm use 20 # (or any Node 20+ version)
Now using node v20.19.0 (64-bit)
PS C:\Users\Administrator\OneDrive\Desktop\newwork> node --security-revert=CVE-2023-46809 decrypt.js
SECURITY WARNING: Reverting CVE-2023-46809: Marvin attack on PKCS#1 padding
🔓 Decrypted message: Secret123!
PS C:\Users\Administrator\OneDrive\Desktop\newwork>
```

The Marvin Attack is a **timing-based variant** of the classic Bleichenbacher RSA padding oracle attack.

What it exploits:
 If your app uses RSA decryption with PKCS#1 v1.5 padding, and returns **different timing/error messages** depending on **padding validity**, attackers can send many modified ciphertexts and **infer the plaintext** byte by byte.

What makes Marvin unique:
 It performs **statistical timing analysis** using small padding variations and Wilcoxon signed-rank tests to identify decryptability — even when error messages are hidden.

What can the attackers gain?
For a vulnerable implementation the attacker is able to decrypt RSA ciphertexts and forge signatures.

For a TLS server that defaults to RSA encryption key exchanges, that means the attacker can record a session and decrypt it later.

For TLS hosts that use forward secure ciphersuites, the attacker would have to perform a massively parallel attack to forge a server signature before a client would time out during the connection attempt. That makes the attack hard, but not impossible.

The attack is also applicable to other interfaces that perform RSA decryption in automated manner but don't provide the attacker ability to perform arbitrary operations with the private key. Examples include S/MIME, JSON web tokens, or hardware tokens (HSMs, smartcards, etc.).

Node.js is **vulnerable to the Marvin Attack** when **all of the following are true**:

Condition	Status
🔧 **OpenSSL is unpatched**	✅ Node is using an OpenSSL version **before** the Marvin fix
🔑 **PKCS#1 v1.5 padding is used**	✅ You use `RSA_PKCS1_PADDING` (default in `crypto.privateDecrypt`)
🧾 **Untrusted ciphertexts are decrypted**	✅ Input comes from the client (e.g., JWE, encrypted API data)
🧠 **No side-channel mitigation**	✅ Your app leaks timing differences (no `timingSafeEqual`, etc.)