# Code Highlighting Test

## JavaScript Example

**JAVASCRIPT**

1 | // This is a comment

2 | function fibonacci(n) {

3 | if (n <= 1) return n;

4 | return fibonacci(n - 1) + fibonacci(n - 2);

5 | }

6 |

7 | const result = fibonacci(10);

8 | console.log(`Fibonacci of 10 is: ${result}`);

9 |

10 | class Calculator {

11 | constructor() {

12 | this.value = 0;

13 | }

14 |

15 | add(x) {

16 | this.value += x;

17 | return this;

18 | }

19 |

20 | multiply(x) {

21 | this.value \*= x;

22 | return this;

23 | }

24 |

25 | getValue() {

26 | return this.value;

27 | }

28 | }

29 |

30 | const calc = new Calculator();

31 | const finalValue = calc.add(5).multiply(3).getValue();

## TypeScript Example

**TYPESCRIPT**

1 | interface User {

2 | id: number;

3 | name: string;

4 | email?: string;

5 | }

6 |

7 | type UserRole = 'admin' | 'user' | 'guest';

8 |

9 | class UserService {

10 | private users: Map<number, User> = new Map();

11 |

12 | async getUser(id: number): Promise<User | null> {

13 | return this.users.get(id) || null;

14 | }

15 |

16 | async createUser(user: User): Promise<void> {

17 | this.users.set(user.id, user);

18 | }

19 | }

20 |

21 | const service = new UserService();

22 | await service.createUser({ id: 1, name: 'John Doe' });

## Python Example

**PYTHON**

1 | import asyncio

2 | from typing import List, Optional

3 |

4 | class DataProcessor:

5 | """A class to process data asynchronously"""

6 |

7 | def \_\_init\_\_(self, batch\_size: int = 100):

8 | self.batch\_size = batch\_size

9 | self.data: List[dict] = []

10 |

11 | async def process\_batch(self, items: List[dict]) -> List[dict]:

12 | """Process a batch of items"""

13 | results = []

14 | for item in items:

15 | # Simulate async processing

16 | await asyncio.sleep(0.01)

17 | results.append({

18 | \*\*item,

19 | 'processed': True,

20 | 'timestamp': datetime.now()

21 | })

22 | return results

23 |

24 | async def run(self):

25 | """Main processing loop"""

26 | tasks = []

27 | for i in range(0, len(self.data), self.batch\_size):

28 | batch = self.data[i:i + self.batch\_size]

29 | task = asyncio.create\_task(self.process\_batch(batch))

30 | tasks.append(task)

31 |

32 | results = await asyncio.gather(\*tasks)

33 | return [item for batch in results for item in batch]

34 |

35 | # Usage

36 | processor = DataProcessor(batch\_size=50)

37 | processed\_data = await processor.run()

## SQL Example

**SQL**

1 | -- Create users table

2 | CREATE TABLE users (

3 | id SERIAL PRIMARY KEY,

4 | username VARCHAR(50) UNIQUE NOT NULL,

5 | email VARCHAR(100) UNIQUE NOT NULL,

6 | created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

7 | );

8 |

9 | -- Complex query with JOIN

10 | SELECT

11 | u.username,

12 | u.email,

13 | COUNT(o.id) as order\_count,

14 | SUM(o.total\_amount) as total\_spent

15 | FROM users u

16 | LEFT JOIN orders o ON u.id = o.user\_id

17 | WHERE u.created\_at >= '2024-01-01'

18 | GROUP BY u.id, u.username, u.email

19 | HAVING COUNT(o.id) > 0

20 | ORDER BY total\_spent DESC

21 | LIMIT 10;

## JSON Example

**JSON**

1 | {

2 | "name": "markdown-docx",

3 | "version": "1.2.0",

4 | "description": "Convert Markdown to DOCX with syntax highlighting",

5 | "scripts": {

6 | "build": "tsup",

7 | "test": "vitest"

8 | },

9 | "dependencies": {

10 | "shiki": "^3.12.2",

11 | "marked": "^15.0.8",

12 | "docx": "^9.3.0"

13 | }

14 | }

## Bash Example

**BASH**

1 | #!/bin/bash

2 |

3 | # Function to check if command exists

4 | command\_exists() {

5 | command -v "$1" >/dev/null 2>&1

6 | }

7 |

8 | # Install dependencies

9 | if command\_exists npm; then

10 | echo "Installing with npm..."

11 | npm install

12 | elif command\_exists yarn; then

13 | echo "Installing with yarn..."

14 | yarn install

15 | else

16 | echo "No package manager found!"

17 | exit 1

18 | fi

19 |

20 | # Build the project

21 | npm run build

22 |

23 | # Run tests

24 | npm test -- --coverage

## Plain Text (No Language Specified)

This is plain text without syntax highlighting.

It should still be displayed in a code block format.

But without any color coding.

## Inline Code

Here's some inline code: const x = 42 and function() { return true; }.