every

August 17, 2026

every

Kind: Function

Source: `src/middleware/combine/index.ts`

Part of: Middleware

Create a composed middleware that runs all middleware and throws an error if any of them fail.

`every` creates a single middleware from multiple middleware functions. It runs each middleware and propagates an error when any middleware fails, allowing the caller to handle the failure through the normal middleware error path.

Signature

```
function every(middleware: (MiddlewareHandler | Condition)[]): MiddlewareHandler
```

Parameters

Name	Type
`middleware`	`(MiddlewareHandler

Returns: `MiddlewareHandler`

Diagram

```
graph LR
  A[Request context] --> B[every middleware]
  B --> C[Middleware A]
  B --> D[Middleware B]
  B --> E[Middleware C]
  C --> F{Any middleware fails?}
```

```
  D --> F
  E --> F
  F -->|Yes| G[Throw error]
  F -->|No| H[Continue request flow]
```

Usage

```ts
import { every } from "./middleware/combine";

const requireUser = async (context: RequestContext) => {
  if (!context.user) {
    throw new Error("Authentication required");
  }
};

const requireProject = async (context: RequestContext) => {
  if (!context.params.projectId) {
    throw new Error("Project ID is required");
  }
};

const validateRequest = every(requireUser, requireProject);

await validateRequest(context);
```

AI Coding Instructions

- Pass middleware that follows the project's middleware function signature.
- Let middleware throw errors for failed checks; `every` propagates those errors.
- Keep each middleware focused on one request concern, such as authentication or input validation.
- Add shared middleware through `every` where a route or handler requires all checks to pass.