

Code Editor

CODE → src/middleware/clientAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ClientAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    request.setHeader('Authorization', 'Richard');

    return next();
  }
}

export default new ClientAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/clientAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ClientAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    request.setHeader('Authorization', 'Richard');

    return next();
  }
}

export default new ClientAuth();
```



TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/clientAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ClientAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    request.setHeader('Authorization', 'Richard');

    return next();
  }
}

export default new ClientAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

TERMINAL

~ Distributed JavaScript Runtime ~

>_ gateway
>_ default
>_ data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```



TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```

TERMINAL

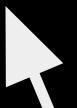
~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name); 
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data

Code Editor

CODE → src/middleware/serverAuth.ts

```
import type { Middleware, NextHandler, Request, Response } from 'jitar';

class ServerAuth implements Middleware
{
  async handle(request: Request, next: NextHandler): Promise<Response>
  {
    const name = request.getHeader('Authorization');
    request.setArgument('name', name);

    return next();
  }
}

export default new ServerAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway
> default
> data