

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/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/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/clientAuth.ts

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

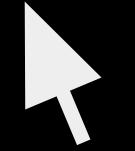
```
class ClientAuth implements Middleware
```

```
{
```

```
  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

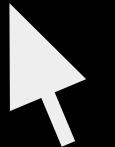
CODE → src/middleware/clientAuth.ts

```
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';
```

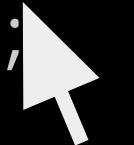
```
class ClientAuth implements Middleware
```

```
{
```

```
  async handle(request: Request, next: NextHandler): Promise<Response>
```

```
{
```

```
    request.setHeader('Authorization', 'Richard');
```



```
    return next();
```

```
}
```

```
}
```

```
export default new ClientAuth();
```

TERMINAL

~ Distributed JavaScript Runtime ~

> gateway

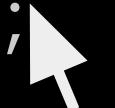
> default

> data

Code Editor

CODE → src/middleware/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'); 
```

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/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