Exception filters

The only difference between the HTTP exception filter layer and the corresponding web sockets layer is that instead of throwing `HttpException`, you should use `WsException`.

```
throw new WsException('Invalid credentials.');
```

> info **Hint** The `WsException` class is imported from the `@nestjs/websockets` package.

With the sample above, Nest will handle the thrown exception and emit the `exception` message with the following structure:

```
{
  status: 'error',
  message: 'Invalid credentials.'
}
```

Filters

Web sockets exception filters behave equivalently to HTTP exception filters. The following example uses a manually instantiated method-scoped filter. Just as with HTTP based applications, you can also use gateway-scoped filters (i.e., prefix the gateway class with a `@UseFilters()` decorator).

```
@UseFilters(new WsExceptionFilter())
@SubscribeMessage('events')
onEvent(client, data: any): WsResponse<any> {
  const event = 'events';
  return { event, data };
}
```

Inheritance

Typically, you'll create fully customized exception filters crafted to fulfill your application requirements. However, there might be use-cases when you would like to simply extend the **core exception filter**, and override the behavior based on certain factors.

In order to delegate exception processing to the base filter, you need to extend `BaseWsExceptionFilter` and call the inherited `catch()` method.

```
@@filename()
import { Catch, ArgumentsHost } from '@nestjs/common';
import { BaseWsExceptionFilter } from '@nestjs/websockets';
```

```
@Catch()
export class AllExceptionsFilter extends BaseWsExceptionFilter {
  catch(exception: unknown, host: ArgumentsHost) {
    super.catch(exception, host);
  }
}
@@switch
import { Catch } from '@nestjs/common';
import { BaseWsExceptionFilter } from '@nestjs/websockets';

@Catch()
export class AllExceptionsFilter extends BaseWsExceptionFilter {
  catch(exception, host) {
    super.catch(exception, host);
  }
}
```

The above implementation is just a shell demonstrating the approach. Your implementation of the extended exception filter would include your tailored **business logic** (e.g., handling various conditions).