# Day 8 - Angular & Asp in 21 days Handouts

Status

## Day 8: HTTP Response Status Codes

A front-end developer should handle all possible consequences of sending a request to a URL.

HTTP specifies some response status codes to categorize response types. All possible status codes are best to be documented for each action to make it predictable for front-end developers.

To return different status codes, the return type of the action should be `ActionResult`. Not void nor custom read model types.

```
public ActionResult<ReadModel> Find(Guid id)
...
```

Instead of returning raw data include the success status code like this:

```
public ActionResult<ReadModel> Find(Guid id)
=>Ok(new ReadModel(...));
```

To document the status code to be readable by Swagger you can add the `ProducesResponseType` on top of the action.

This is how you can document the success of a get request for example:

```
[ProducesResponseType(typeof(ReadModel), 200)]
public ActionResult<ReadModel> Find(Guid id)
=> Ok(new ReadModel(...));
```

If the requested resource is not found you can `return NotFound()` from the action.

```
[ProducesResponseType(400)]
[ProducesResponseType(typeof(ReadModel), 200)]
```

```
public ActionResult<ReadModel> Find(Guid id)
```

You may want to read the HTTP 2 RFC to learn more about HTTP status codes.

To handle responses from an angular

```
this.customerService
      .findCustomer({ id: this.customerId })
      .subscribe(flight => this.customer = customer
        , this.handleError)
```

Define `handleError` as an arrow function.

```
private handleError = (err: any) => {
  ...
}
```

So you can access status codes from the error object and handle them individually.

```
private handleError = (err: any) => {
  console.log("Response Error. Status:", err.status)
  console.log("Response Error. Status Text:", err.statusText)
  console.error(err)
  if (err.status == 404) {
    ...
  }
}
```

There are 5 categories of HTTP Status codes: