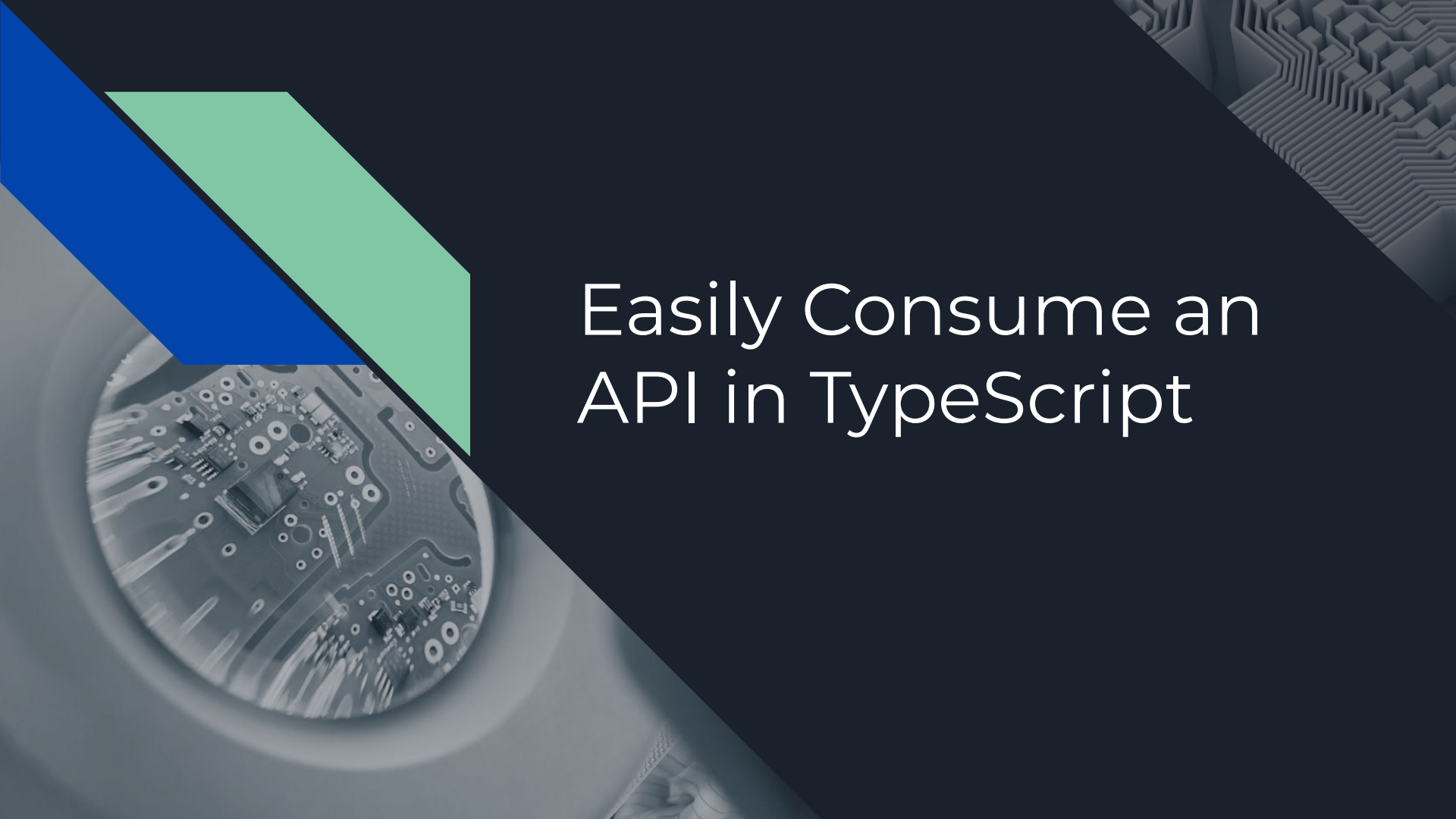
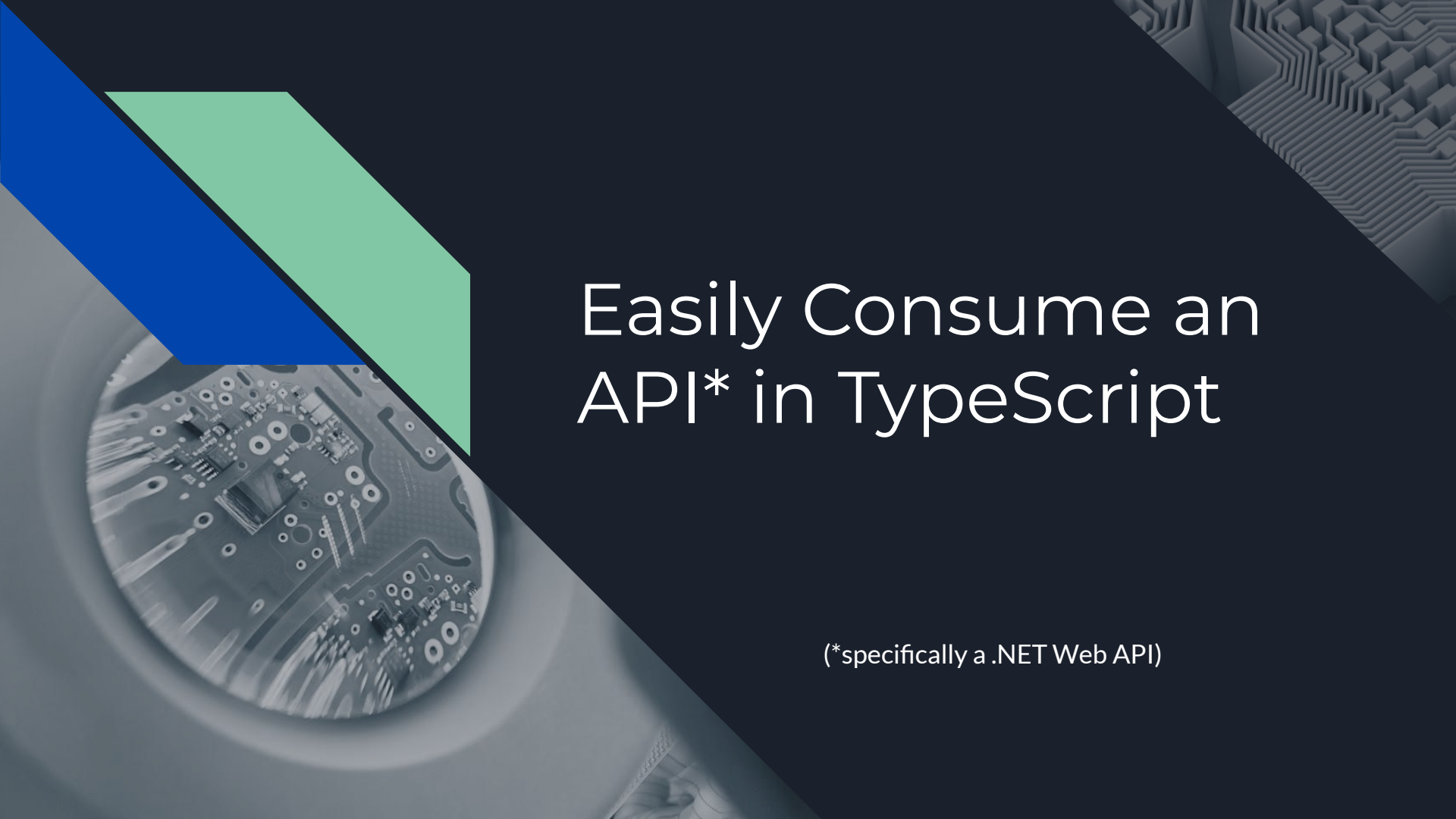

Easily Consume an
API in TypeScript



Easily Consume an API* in TypeScript

(*specifically a .NET Web API)


msi


QuadTech


FCx
performance


THOUGHTFOCUS


KOHLER
GENERATORS


MUTH


STRATTEC

Honor Flight NETWORK

STARS AND STRIPES HONOR FLIGHT WISCONSIN

TOC

Overview

Generally web applications rely on some sort of backend data source and most common today is a web-based API that exchanges data as JSON. These API's can follow an RPC-style or a "REST-ful" style that generally involves using HTTP methods to provide CRUD access to the data.

While this style can be more approachable by developers than binary or XML-based API's, there still are factors to consider:

- Documentation & discoverability
- Request/Response schema a & data types
- Authentication
- Validation and error codes
- Changes over time

Origins of Swagger (OpenAPI)

2011 Started by Tony Tam to automate documentation and SDK generation. Maintained by SmartBear. Open-sourced with support for Node.js and Ruby on Rails

2016 With help from Google, IBM and Microsoft the specification was separated from the tooling and renamed OpenAPI Specification

The tooling that was separated included a validator, a code generator, and tools for editing the Swagger/OpenAPI JSON documents using JavaScript or other languages. There is also a web UI to render the API documents and allow interaction from a browser.

SmartBear has added to these tools and created an ecosystem that includes free and paid services and integrated them with various development tools/platforms.

Swashbuckle, NSwag, and others

These projects bring some of the same functionality to ASP .NET Core web applications.

(Note: Microsoft has their own tool for reading existing OpenAPI specification documents and creating stub code/controllers in a .NET project for implementing an API.)

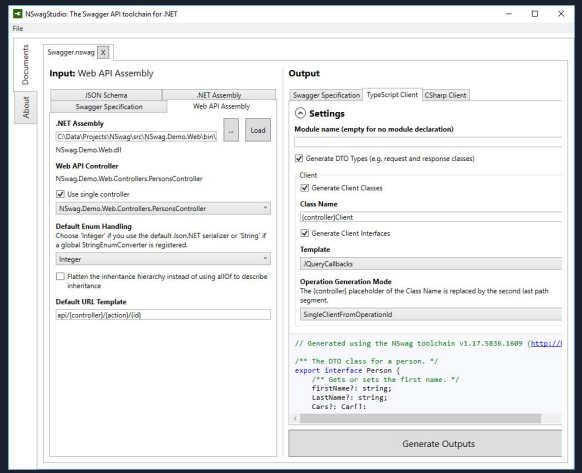
Swashbuckle	Add the SwagerGen library to a Web API project as a NuGet package and it will emit an OpenAPI JSON document for the existing API at an endpoint you specify. The SwaggerUI package will present a web page for reading and interacting with the API.
NSwag	Adding this package gets you everything above, plus the ability to add build targets that create client code for various platforms.
WebApiClientGen	Generates stricter types than possible with Swagger. Main focus is C# client but can emit forms of TypeScript. (Have not used it - YMMV)

NSwag Outputs

- JQuery with Callbacks, JQueryCallbacks
- JQuery with promises JQueryPromises
- AngularJS using $http, AngularJS
- Angular (v2+) using the http service, Angular
- window.fetch API and ES6 promises, Fetch (use this template in your React/Redux app)
- Aurelia using the HttpClient from aurelia-fetch-client, Aurelia (based on the Fetch template)
- Axios (preview)

NSwagStudio

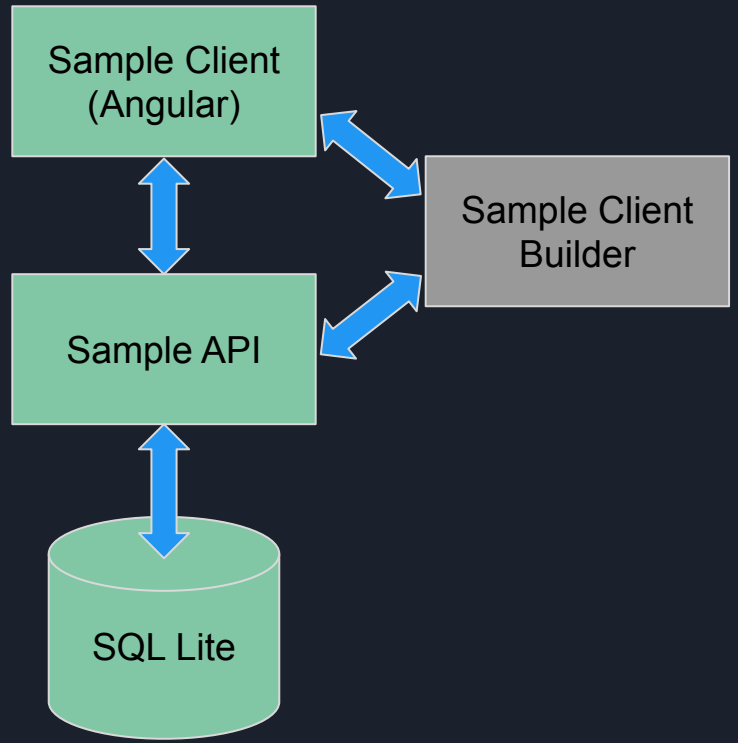
- Windows Desktop UI to construct build a client based on an OpenAPI document.



Sample Stack

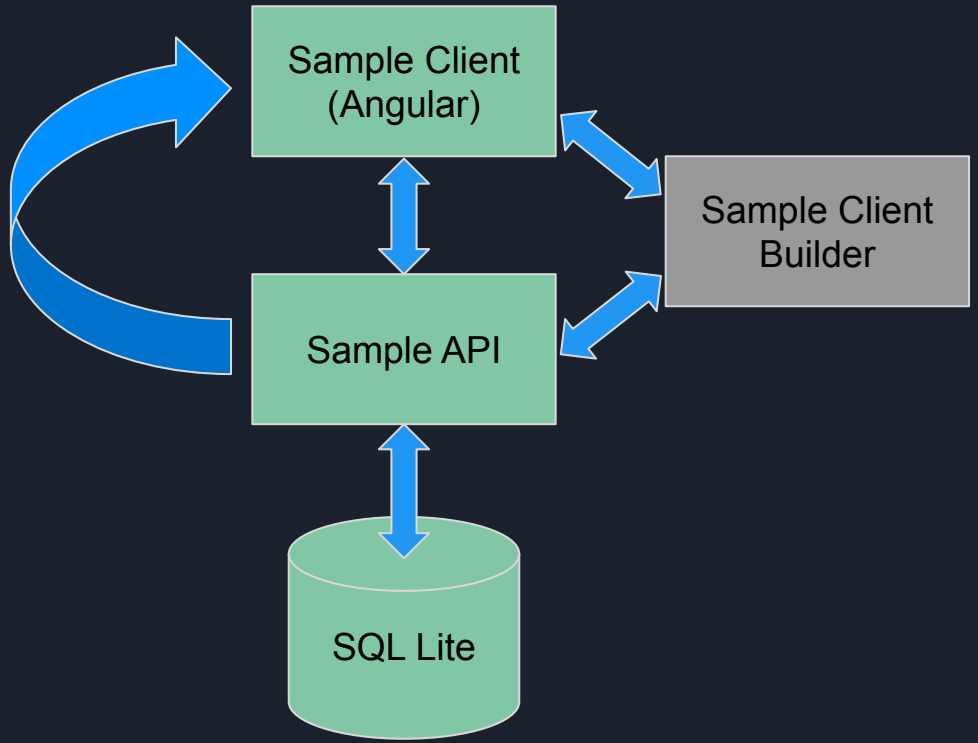
Need to think about how you will deliver the client definitions:

- Private NPM packages (NuGet Packages for C#)
- Simple build output between projects
- An application in the CI/CD pipeline



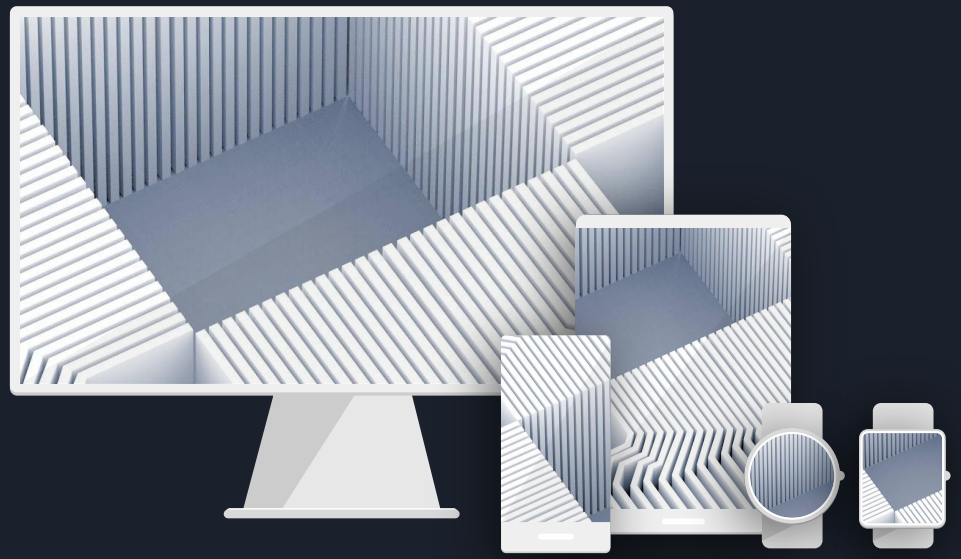
Sample Stack

MS Build



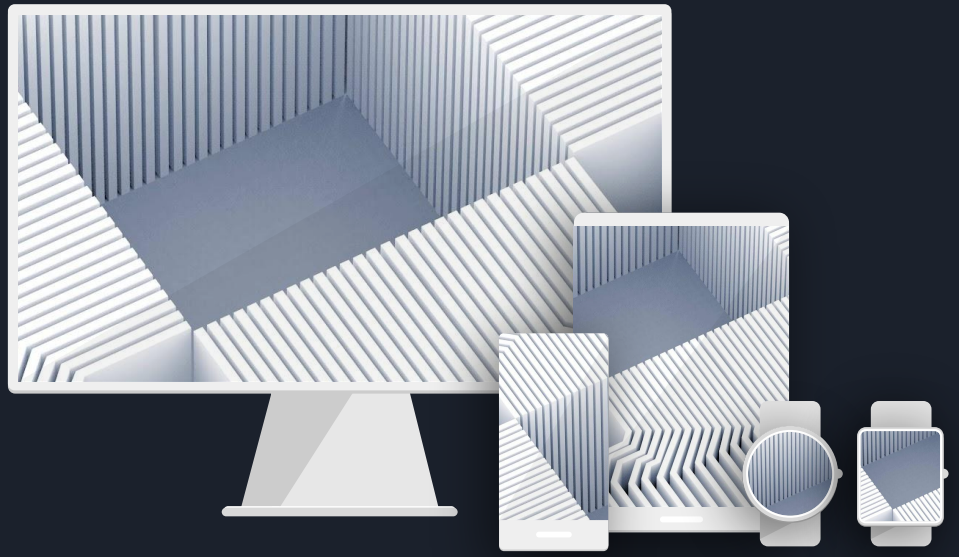
API Demo

Let’s look at some Swagger and NSwag output:



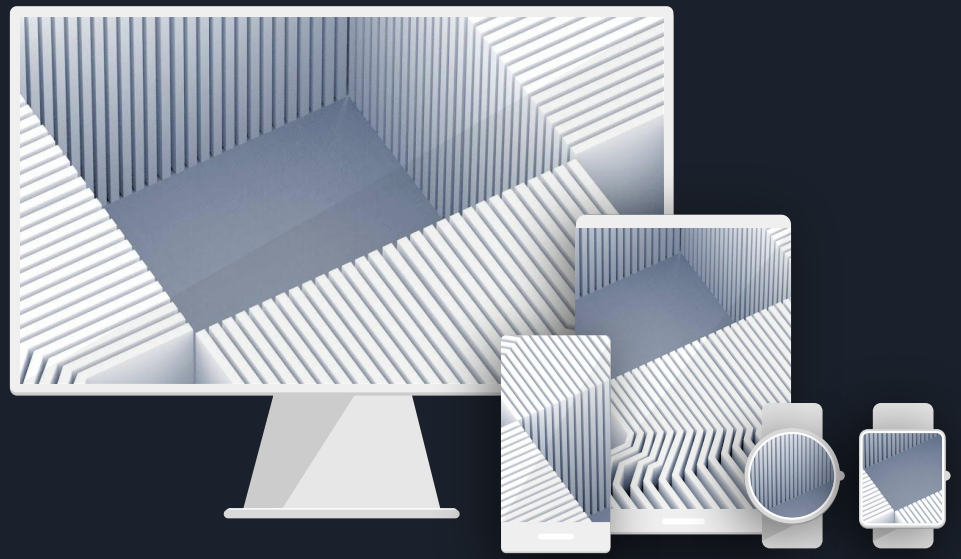
Client Demo

Let’s look at what we get in our web app:

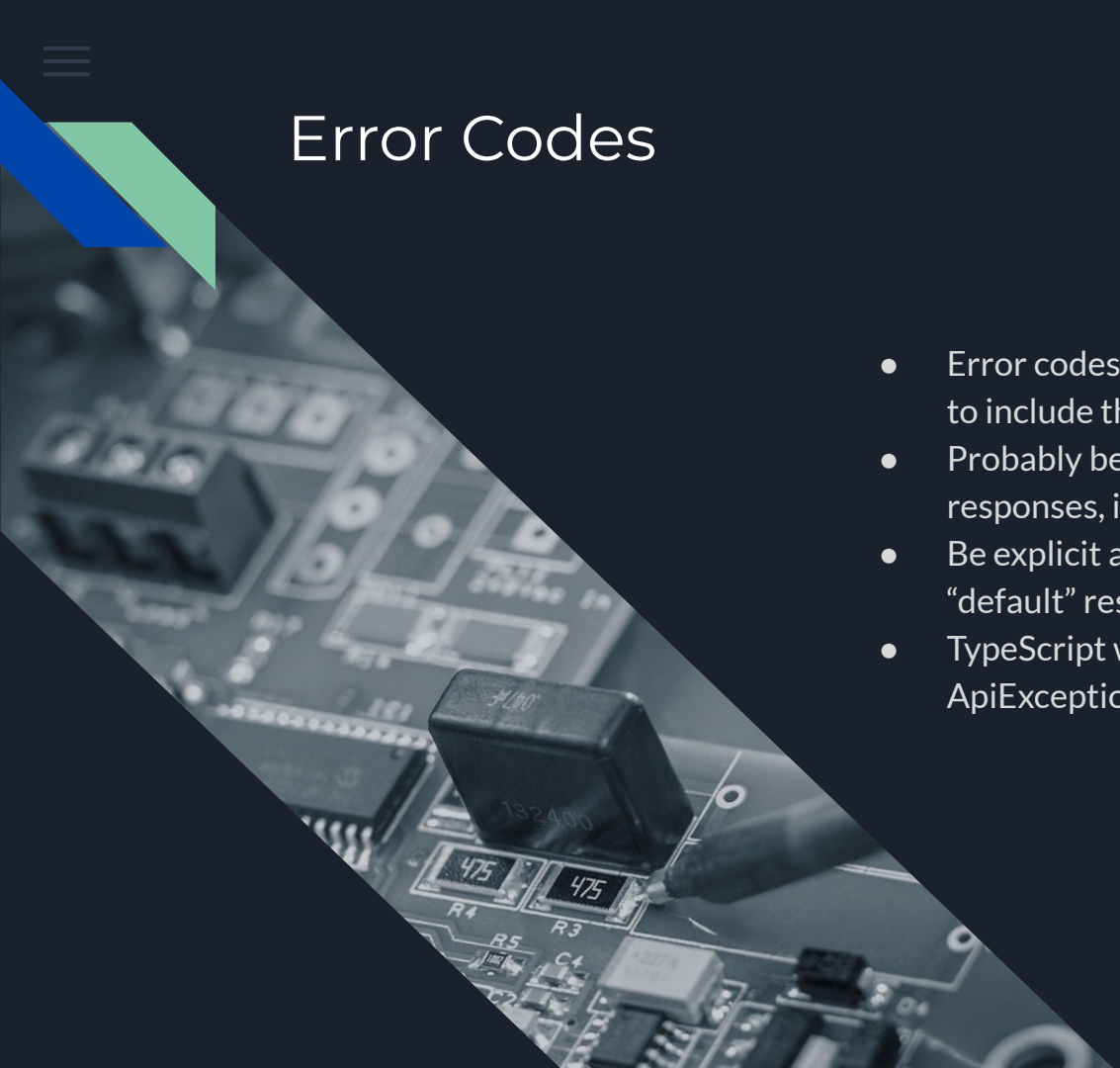


Builder Demo

Let’s look at some client output



Error Codes



- Error codes can be extended in OpenApi/Swagger to include their own data models if desired.
- Probably better to stick to the default HTTP responses, if possible.
- Be explicit about the ones you expect and use the “default” response for everything else.
- TypeScript wraps exceptions in a class called ApiException that extends Error

Authentication

- Various Schemes Swagger can document:
  - http – for Basic, Bearer and other HTTP authentications schemes
  - apiKey – for API keys and cookie authentication
  - oauth2 – for OAuth 2
  - openIdConnect – for OpenID Connect Discovery
- Generated client code may not be involved in auth
- May have the client define an endpoint to perform the authentication and get a bearer token.

Versioning

- WebApi can support multiple API versions in a single ASP.NET Core web service.
- Swagger/Swashbuckle can support multiple versions of an API.
- NSwag can publish those version in a single client output or you can filter to separate versions.
- Use multiple configs or filters to create separate clients.

Gotcha’s

- Inline builds can begin to get slow when API’s become complex.
- May want a way to get the definitions to the client projects without going thru source control.
- Using the example setup, a build will complete successfully even when a generation error occurs. Your build pipeline may need to know the difference.
- For TS, names must differ regardless of parameters
- Different generators - YMMV

Questions?