My Journey in Nextjs website creation

# Starting off the project using ‘Build a Duolingo Clone’ by Code With Antonio

## Creating the project

Go to the terminal.

You can’t create a project if you are restricted from running scripts in visual studio.

Run the line: Get-ExecutionPolicy -List

If the scope “Current User” is set to “Undefined”, run the line: Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser .

RemoteSigned means you can run any scripts that are signed for, so it does not mean unlimited script running access.

Next: npx create-next-app@latest, y to install packages, name the project then:

1. A screenshot of a computer program

   Description automatically generatedTypescript = yes
2. ESLint = yes
3. Tailwind CSS = yes
4. ‘src/’ directory = no
5. App Router = yes
6. Custom import alias = no

If there are recommended npm updates, run them.

Now use the file location to move the project to where you would like.

Use Open folder on the project folder and trust the author.

## Using ShadCN

Type npx shadcn@latest init into the terminal to access the large components collection to design your own library of components.

Find all of the components at <https://ui.shadcn.com/docs>. Each component can be installed by typing a certain line into the terminal such as ‘npx shadcn@latest add button’. The website will provide these lines for each component as well as how to import each component to the top of your page scripts.

## Handling Imported Shadcn Components

Look at the website to get codes to copy and paste into the terminal to import them and they will appear in the components/ui file.

Component files have Variants of components and component sizes that you can call from page.tsx. You can add/remove/customize whatever sizes and variants.

# Creating Pages and Layouts

## Page.tsx Files

Create a folder in the ‘app’ folder. Name it whatever you like (Ex: “home”), then make a file in that folder called ‘page.tsx’. To create a page without error that appears on localhost:3000/home, write this:

const variablePageName = () => { return ( PUT PAGE CONTENT HERE ); };

export default variablePageName;

Ex: const HomePage = () => { return ( <p> paragraph text </p>); };

export default variablePageName;

Each folder under app containing a page.tsx file is a url that can be accessed.

Ex: app->page.tsx can be accessed through [http://localhost:3000](http://localhost:3000/) and app->buttons->page.tsx can be accessed through <http://localhost:3000/buttons>.

Creating a page.tsx in a folder under app where the name is enclosed in () will make that page show in <http://localhost:3000> so it is good for organising.

## Layout.tsx Files

Layout files are page files that take priority so export defaulting the same as a page.tsx will overwrite a page file. However, if you include {children} in the const variable it will read the page.tsx code.

### Example

import React from "react";

type Props = {

    children: React.ReactNode;

};

const MarketingLayout = ({ children }: Props) => {

    return (

        <div className="flex flex-col snap-center p-5 space-y-5 max-w-[200px] bg-cyan-500 tracking-wide">

            <p>aaaaa</p>

            {children}

        </div>

    )

}

export default MarketingLayout;

This layout will affect any child folder, so you can create a base layout for a lot of your pages then add them

>:3

## Alternate File References

If you want to reuse code, make a tsx file (example.tsx), write

export const Example = () => {

    return (

        Your code here

    );

};

Then do import { Example } from “./example”; and place <Example/> inside your code to call that code.

# Important Code Information

## Links to help

Using fonts - <https://nextjs.org/docs/pages/building-your-application/optimizing/fonts>

Tailwindcss - <https://tailwindcss.com/docs/installation>

Tailwind uses - <https://tailwindcss.com/docs/utility-first> , check out customization/Base styles/layout etc…

## Class Summaries

<div>: Used to group elements and apply styles to that group.

<div/>: Absolutely useless term for div, but means that it is referencing something without encapsulating anything in that page.

<main>: dominant content of the <body> of a document. Ex:  className="flex flex-col gap-8 row-start-2 items-center sm:items-start"

<footer>: Footer info. Ex: className="row-start-3 flex gap-6 flex-wrap items-center justify-center"

<a>: Creates hyperlink to web page/file/email address/location in same page/any url address. Has className, href, target, rel.

<ol>: defines an ordered list, numerical or alphabetical.

<li>: defines each item of an ordered list (<ol>).

<h1> -> <h5>: represents headings from 1 (most important/biggest) to 5 (lowest level/smallest).

<p>: Defines a paragraph of text, good for breaking up large chunks of text.

<meta/>: Info about the page, place in the header.

<image>: Image, can be placed inside brackets of other <>. Src=image source location. Alt = description of image, width, height, aria-hidden=?.

<image … />: image doesn’t need to encapsulate any code.

<code>: Still unsure, but seems to highlight/change color of text. A black screen with orange text

Description automatically generated



## Important Class Code

href="…": added to icon/button/text etc... to take user to specified link

alt= “…”: alternative text if image not found

## ClassName Terms

className is a variable inside <> brackets that you can add terms to create effects such as className=“text-cyan-500 tracking-wide”.

Note: some numbers can be replaced by “[“+number+“px]” (Ex:h-10 -> h-[10px]). This measures in pixels instead.

### Text

“text-{…}”: Text options (Ex: alignment, size, color, background)

“text-{colorName}-{number)}”: 500 is pure color, 0 is blackest vrsn and 1000 is whitest vrsn.

“text-{fontSize}”: text size. (Ex: 20xl,xl,sm etc…)

“ text-primary-foreground”: White?

“uppercase”: It makes it lowercase 🥴

“tracking-wide”: spaces out letters

“underline-offset-{number}”: underline text where ‘number’ is space between underline and text.

“font-{fontName}”: set font type (Ex: bold, semibold, italic etc…)

“font-{fontType}”: set font style (Ex: bold,italic,opacity, etc…)

“px-{number}”: Padding (spacing) to left and right. (recommend number=1)

“py-{number}”: Padding (spacing) to top and bottom. (recommend number=0.5)

“bg-{colorName}-500”: Set background color of text.

“bg-black/[0.05]”: set grayscale background colour between 0 & 1.

“rounded/rounded-{roundedType}”: round corners of text background.

### Columns

“flex”: Keeps things the same size in columns such as 2 buttons with differing texts.

“flex-col”: Keeps things ordered inside a column.

“space-y-{number}”: space between buttons.

“p-{number}”: Padding between column and what’s inside the columns.

“max-w-{number}”: Maximum width of column. Screen size can change.

“flex”

“border-b-[{number}px]": border of bottom of column extension.

### Buttons

“shadow”: A cartoon character of a black and red hedgehog

Description automatically generated

“h-{number}”: height of button(?).

“w-{number}”: width of button(?).

“px-{number}” & “py-{number}”: padding.

“rounded”: options for rounding button corners.

“hover:…”: button options for when you hover the mouse on the button

“border-{letter}-[{number}px]": border of bottom of column extension.

Border letters: b=bottom, r=right, t=top, e=right?, s=left?, x=sides, y=top/bottom

## Keywords

Metadata: Data that describes other data such as the content, format or attributes (organizational info).

## Additional Information

Looking up things you’re unsure about: specify that the code you’re looking for comes from html.

To see your website in action, type ‘npm run dev’ and go to <http://localhost:3000/> in your internet url.

Check imported components for cool className terms.