Templates

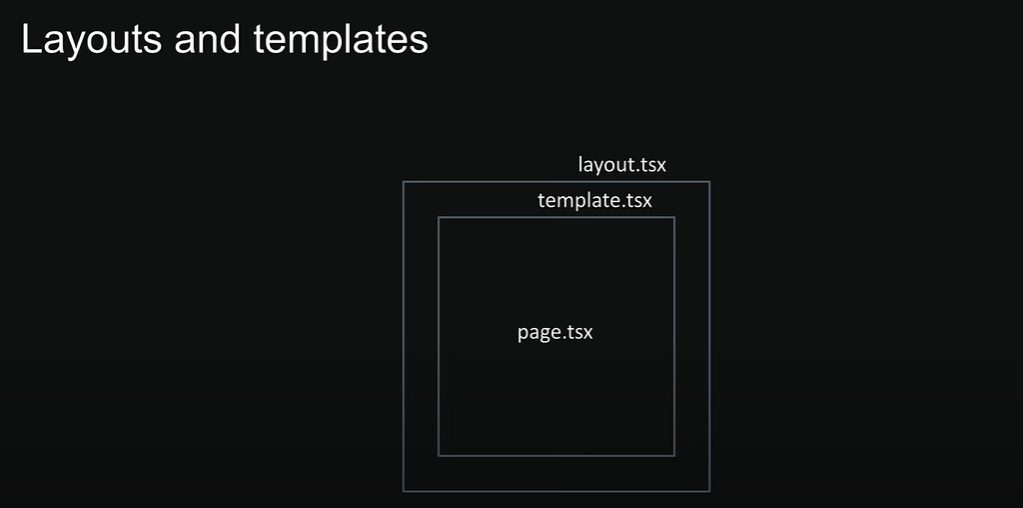
Teamplates are similar to layouts in that they are also UI shared between multiple pages in your app

Whenever a user navigates between routes sharing a template, you get a completely fresh start

* A new template component instance is mounted
* DOM elememts are recreated
* State is cleared
* Effects are re-synchronized

Create a template by exporting a default React component from a template.js or template.tsx file

Like layouts, templates need to accept a children prop to render the nested route segment



Loading UI



**loading.tsx**

This file helps us create loading states that users see while waiting for content to load in a specific route segment

The loading states appear instantly when navigating, letting users know that the application is reponsive and actively loading content

**loading.tsx benefits**

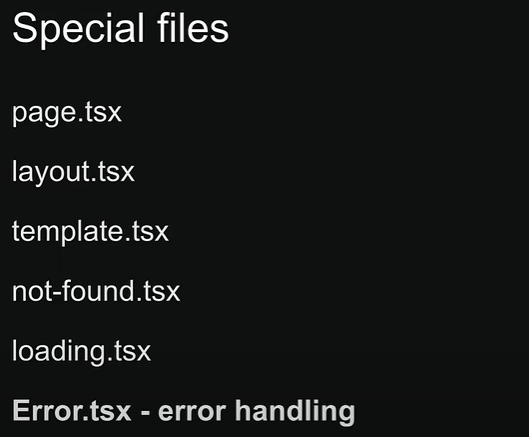
1. It gives users immediate feeback when they navigate somewhere new

This makes your app feel snappy and responsive, and users know their click actually did something

1. Next.js keeps shared layouts interactive while new content loads

Users can still use things like navigation menus or sidebars even if the main content isn’t ready yet

**Error handing**



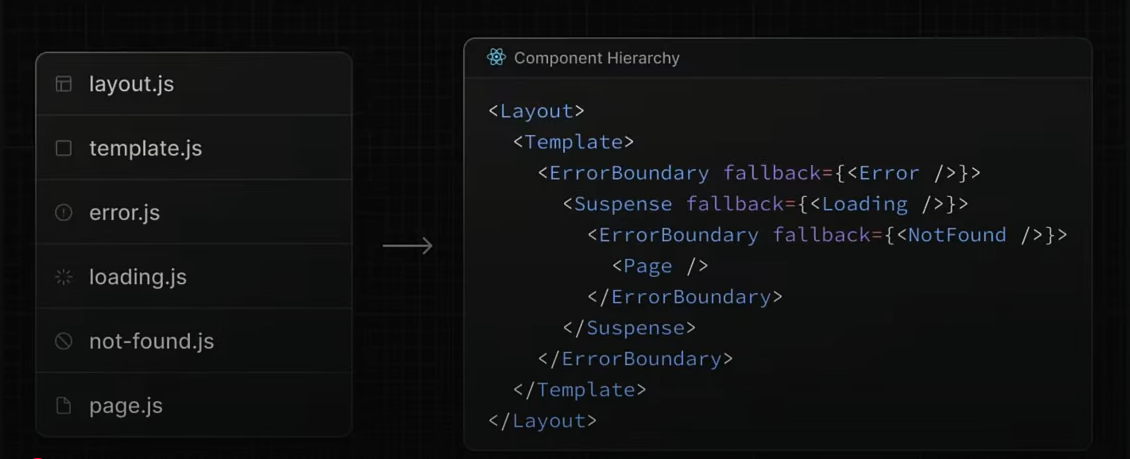
**Handing errors in nested routes**

Errors always bubble up to find the closest parent error boundary  
An error.tsx file handles errors not just for its own folder, but for all the nested child segments below it too  
By strategically placing error.tsx files at different levels in your routes folders, you can control exactly how detailed your error handling gets

Where you put your error.tsx file makes a huge difference – it determines exactly which parts of your UI get affected when things go wrong

**Handing errors in layouts**

The error boundary won’t catch errors thrown in layout.tsx within the same segment because of how the component hierarchy works  
The layout actually sits above the error boundary in the component tree



**Handing global errors**

If an error boundary can’t catch errors in the layout.tsx file from the same segment, what about errors in the root layout?  
It doesn’t have a parent segment – how do we handle those errors?  
Next.js provides a special file called *global-error.tsx* that goes in your root app directory  
This is your last line of defense when something goes catastrophically wrong at the highest level of your app

* Works only in production mode
* Requires html and body tags to be rendered