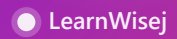


Theme Images, URLs, SVG Color and the Designer Image Picker

Lesson Guide · Lab / Exam Guide



Lesson Guide

Image sources that are not files you ship next to the page: named theme images loaded from the client theme cache, relative and absolute URLs, base64 data embedded in the theme, the Visual Studio Image Selector that writes the source string for you, and the `?color=` suffix that recolors a suitable monochrome SVG with a theme color name. What this means for a developer who knows basic C# and has done some Wisej.NET control and designer work — and how to read this module.



Production focus

Image sources that are not files you ship next to the page: named theme images loaded from the client theme cache, relative and absolute URLs, base64 data embedded in the theme, the Visual Studio Image Selector that writes the source string for you, and the `?color=` suffix that recolors a suitable monochrome SVG with a theme color name.

1. The production problem

IconDesk now has a command panel and an image lab, and every picture in them is a shipped file plus a path somebody typed. That holds until the application is re-themed, deployed under a different virtual directory, or asked to show the same symbol in a warning colour on one screen and a neutral one on another. This module moves the origin of an image out of the project folder and into the theme, a URL or the designer, so the string a control carries still means something after the artwork, the theme or the server changes.

2. Key concepts

Concept	Why it matters
named theme image	A semantic name such as <code>icon-print</code> that the active theme maps to artwork, so a different theme can supply a different picture with no C# change.
client theme cache	Theme images travel with the theme and are already in the browser when a control asks for one, which makes a theme the right home for a small set of command symbols and the wrong home for thousands.
relative URL source	An image-source string resolved against the application: the simplest form for a logo deployed with the app.
absolute URL source	A complete external address that turns the image into a dependency on another system, with availability, privacy, CORS and Content Security Policy consequences.
embedded base64 image	Image bytes written into the theme definition itself, so the artwork needs no second request, at the cost of a larger theme.
Image Selector	The Visual Studio design-time picker that browses project images, theme images and referenced icon-pack resources, then serializes the chosen source string into <code>Designer.cs</code> .
?color= suffix	A suffix on an image source that asks Wisej.NET to recolour a compatible monochrome SVG with a CSS colour or a theme colour name.

3. Lesson narrative

Four forms an image can take inside a theme

A Wisej.NET theme is not only colours and borders; it also carries image definitions that the framework and your application can both use. The Theme Builder documentation describes four useful forms for those definitions. A **relative URL** is resolved against the application. An **absolute URL** names a complete external address. **Embedded base64** data puts the image bytes inside the theme definition, so nothing else has to be fetched. A **named image** points at an image definition, which adds a layer of indirection: many appearances can reference the same name, and changing the definition once changes every use.

The indirection is the interesting part for IconDesk. When the gallery asks for a name rather than a file, the decision about which artwork that name means belongs to the theme, and the theme can be swapped at deployment time or per customer. When the page asks for a URL, the decision belongs to whoever typed the URL, and switching themes changes nothing. The lab puts both behaviours on one screen so the difference is not a matter of belief.

Absolute URLs deserve a deliberate decision rather than a habit. An image fetched from another host is a runtime dependency: it can be unavailable, it can require authentication the browser does not have, it can be blocked by a Content Security Policy, and its cache policy is not yours. An icon that must always be present is better deployed with the application or packaged as a resource, which is where Module 5 goes.

A theme image name is a key, not a path

Assigning a theme image is the shortest line in the course: set an `ImageSource` property to the name, and Wisej.NET loads it from the client-side theme cache. Nothing on the server opens a file and no `System.Drawing.Image` is created, so this is the second pipeline from Module 1, not the first. The string is a theme key. If another theme maps that key to different artwork, the application changes appearance without a single edit in C#, which is why semantic names such as `icon-print`, `icon-save` or `icon-warning` beat a pasted path.

That preloading behaviour is also a sizing rule. Theme images arrive with the theme, so a theme suits a small, stable set of framework-level command symbols. Icon-pack resources behave the other way round: each icon is requested individually and then cached by the browser, which scales to very large catalogues but makes the first use of each icon cost a request. Knowing which of the two you are using tells you what the network panel should look like, and that is the evidence Module 4 asks you to collect.

```
// IconGallery: three source strings, three different resolutions.
buttonPrint.ImageSource = "icon-print";           // theme key
logo.ImageSource        = "Images/company-logo.svg"; // relative URL
avatar.ImageSource       = "https://cdn.example.com/users/42.jpg"; // absolute

// Runtime choices stay semantic, never assembled from user text.
buttonStatus.ImageSource = isDanger
    ? "icon-warning?color=error"
    : "icon-info?color=activeText";
```

Let the Image Selector write the source string

The Wisej.NET designer in Visual Studio offers an image selection experience for source properties. It browses the images in the project, the images the current theme defines, and, when icon-pack assemblies are referenced, the packs it discovers automatically. The icon-pack explorer shows a list or grid with a preview, and a monochrome icon can be previewed in a chosen colour before you commit. When you pick one, the designer serializes a concrete source such as a `resource.wx/...` URL into `Designer.cs`.

This is more than typing assistance. Reading the generated line teaches the runtime representation: choosing a pack icon does **not** copy the SVG into the form. The form stores a source string that points back at the pack assembly, which is why five forms using one icon still ship a single copy of the artwork. The lab asks you to open `Designer.cs` and read those strings for exactly that reason.

- Project images: pick with the selector so no path is typed twice.
- Theme images: use the theme list and preview so you see the current theme's artwork.
- Icon-pack resources: search and preview rather than memorising file names.

Recolouring a monochrome SVG with ?color=

Wisej.NET can recolour a compatible monochrome SVG by adding `?color=` to the image source. The value can be a CSS-style colour such as `red` or `#3078d4`, or the name of a theme colour such as `error` or `activeText`. A theme colour is the stronger choice because it keeps making sense when the theme changes: a hard-coded dark grey can vanish against a dark theme, while a semantic theme colour is

designed to adapt with it. The same suffix works on a theme image name and on a `resource.wx` resource source, because it is applied to the source string rather than to a particular storage mechanism.

Recolouring is not universal, and the failure is quiet. The documentation warns that it may fail when the SVG itself hard-codes `fill` or `stroke` values, because those values win. For an icon meant to be coloured by Wisej.NET, remove the conflicting hard-coded colours where that is appropriate; for a multi-colour logo, leave it alone. IconDesk's gallery should hold one of each so the lab note can say which artwork accepted the suffix and which ignored it. Keep runtime source strings semantic, and never build a resource path out of untrusted user text.



Name the mechanism first

Before you assign any image, say which mechanism you are using: a `System.Drawing.Image` object the server renders as PNG, an image-source string the browser resolves, a named theme image, a `resource.wx` embedded resource, or an icon font drawn as HTML. Assign one image property per control so the others cannot quietly reset it, keep the source string meaningful instead of a pasted magic path, prefer SVG for interface icons so they stay sharp at every size and scale, and give every icon that carries meaning a text alternative.



Learning objectives

By the end of this module you'll be able to:

- ✓ Use relative URLs, absolute URLs, embedded base64 data and named images in a Wisej.NET theme.
- ✓ Assign theme image names directly to `ImageSource`.
- ✓ Use the Visual Studio design-time image selector to select local, theme and icon-pack assets.
- ✓ Recolor suitable SVG icons with the Wisej.NET `?color=` syntax and theme color names.

Lab / Exam Guide

What you'll build in the hands-on lab, the suggested approach, and the required deliverables.



Hands-on lab

Fill an IconGallery page in IconDesk using the designer Image Selector instead of typing source strings by hand. Assign images to five controls: one named theme image, one project SVG referenced by relative URL, one absolute URL and at least two icon-pack assets if a pack is installed. Recolor one monochrome SVG with the `?color=` suffix and a theme color name, open the generated Designer.cs and read the image-source strings the picker produced, then switch the application between two Wisej themes and record which assets changed and which stayed exactly the same.

Starting point

Open the IconDesk solution where Module 2 left it: the command panel from Module 1 and the ImageLab page with its size-mode grid, its `LoadAsync` PictureBox and its crisp SVG. Add a new page named `IconGallery` with five empty controls and a theme selector, so every image on it is assigned during this lab and nothing is inherited from an earlier module.

Suggested approach

- ✓ Place the five gallery controls first and give each one a label naming the mechanism you intend to use, before any image is assigned.
- ✓ Assign every image through the designer Image Selector: the theme list for the named theme image, the project tree for the relative SVG, the address field for the absolute URL, and the icon-pack explorer for the pack assets.
- ✓ Add the `?color=` suffix with a theme colour name to one monochrome SVG, and try it on a multi-colour asset as well so you can see the suffix being ignored.
- ✓ Open `Designer.cs`, copy the generated image-source lines into the lab note and mark each one as a theme key, a relative URL, an absolute URL or a resource source.
- ✓ Switch the application between two Wisej themes, capture the gallery in both, and record which controls followed the theme and which did not.

Required deliverables

- ✓ IconGallery page with five controls whose images were assigned through the designer Image Selector
- ✓ One named theme image, one relative project SVG, one absolute URL and at least two icon-pack assets among them
- ✓ Monochrome SVG recoloured through the `?color=` suffix with a theme colour name
- ✓ Designer.cs excerpt showing the image-source strings the picker generated
- ✓ Theme-switch note recording which assets followed the theme and which stayed fixed

Student review questions

- ✓ The gallery renders correctly on your machine and shows nothing for the absolute URL on a colleague's. Which of the five mechanisms could produce that difference, and what would you check first?
- ✓ Two of your icons refuse to change colour when the suffix is added. What is inside those files, and when is editing them the wrong answer?
- ✓ A designer asks for the same symbol in the gallery, a toolbar and a grid cell. Does the theme key, the relative URL or the pack resource cost the least to change later, and why?



Name the mechanism first

Before you assign any image, say which mechanism you are using: a `System.Drawing.Image` object the server renders as PNG, an image-source string the browser resolves, a named theme image, a `resource.wx` embedded resource, or an icon font drawn as HTML. Assign one image property per control so the others cannot quietly reset it, keep the source string meaningful instead of a pasted magic path, prefer SVG for interface icons so they stay sharp at every size and scale, and give every icon that carries meaning a text alternative.