**What is an SVG?**

**What is an SVG?**

To understand what an SVG is, we'll dive right into a helpful tutorial from our friends at [W3Schools](https://www.w3schools.com/graphics/svg_intro.asp). SVG stands for Scalable Vector Graphics. In 'simple' terms, SVG is a way to define images in a two-dimensional space using XML coded tags with specific parameters.

**SVG Example:**

<html>

<body>

<h1>My first SVG</h1>

<svg width="100" height="100" xmlns="http://www.w3.org/2000/svg">

<circle

cx="50"

cy="50"

r="40"

stroke="green"

stroke-width="4"

fill="yellow"

/>

</svg>

</body>

</html>

SVGs are awesome because they maintain their quality, no matter what size you make them. If you stretch a traditional image file like a .jpg or .png, they become pixelated and lose clarity. SVGs don’t suffer from this issue because they’re scalable. They’re defined within an exact parameter, thus maintaining their quality regardless of scale.

I encourage you to play with editing the parameters in the [**W3Schools SVG Demo**](https://www.w3schools.com/graphics/tryit.asp?filename=trysvg_myfirst). Experiment with how the different parameters can change your image! There's lots of documentation on that website detailing all the tags and features one could add to an SVG.

**Creating Your Own SVG**

Let's look at how we can create our own simple SVG, right in our IDE. Create the file img/example.svg. We can use the <svg> tag to define what our simple image will look like.

<svg xmlns="http://www.w3.org/2000/svg" width="500" height="500">

<text x="200" y="250" fill="white">

Hi! You decoded this!{" "}

</text>

</svg>

❗ **IMPORTANT** You will likely need to download a SVG preview extension to view the SVG in your IDE. I recommend trying [**SVG Preview**](https://marketplace.visualstudio.com/items?itemName=SimonSiefke.svg-preview).

Importantly, this SVG code ***is not*** a URI, but we can convert this into a URI that a browser can understand by passing all the necessary data through the URL of our browser.

**Converting to a URI**

In your terminal, enter the command base64 --help to determine if you have base64 installed, this isn't compatible with all computers, so if you don't have it available, you can copy the encoding I've provided below.

For those with base64 installed, first switch to your img directory.

cd img

You should get an output like this (those without base64 can copy and paste this value):

PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZHRoPSI1MDAiIGhlaWdodD0iNTAwIj4KPHRleHQgeD0iMjAwIiB5PSIyNTAiIGZpbGw9IndoaXRlIj5IaSEgWW91IGRlY29kZWQgdGhpcyEgPC90ZXh0Pgo8L3N2Zz4=

This weird output is the base64 encoded example.svg. We can now add a prefix which tells our browser what type of data this is, 

This same process can be applied to our SVG images for our NFTs. You can navigate to the [**GitHub Repo**](https://github.com/Cyfrin/foundry-nft-f23/blob/main/images/dynamicNft/happy.svg?short_path=224d82e) to see the code which represents our happy and sad SVGs.

**Happy.svg**

<svg

viewBox="0 0 200 200"

width="400"

height="400"

xmlns="http://www.w3.org/2000/svg"

>

<circle

cx="100"

cy="100"

fill="yellow"

r="78"

stroke="black"

stroke-width="3"

/>

<g class="eyes">

<circle cx="61" cy="82" r="12" />

<circle cx="127" cy="82" r="12" />

</g>

<path

d="m136.81 116.53c.69 26.17-64.11 42-81.52-.73"

style="fill:none; stroke: black; stroke-width: 3;"

/>

</svg>

❗ **PROTIP** If you don't have happy.svg and sad.svg images within your img folder, now would be a great time to create them! Copy the SVG code from the [**GitHub Repo**](https://github.com/Cyfrin/foundry-nft-f23/tree/main/images/dynamicNft)!

Once we have both of these images in our workspace, we can run our base64 commands to encode them (those without base64, feel free to grab the encodings below):

base64 -i happy.svg



base64 -i sad.svg



Append our prefix data:image/svg+xml;base64, and view the images in your browser to assure things are working as expected!

**Wrap Up**

This is more like what we'd expect to see in a TokenURI, and it's small enough that we can reasonable store this data directly on-chain! As an additional bonus to storing this data on chain and SVGs basically being code is ... we can interact with this data and change it programmatically! It just keep getting cooler.

In the next lesson we'll go over creating an NFT, leveraging on-chain data allowing it to be *dynamic*.

Let's go!