Markup Languages

The first thing that needs to be said is that a *markup language* is not a programming language. Programming languages provide instructions that tell a computer how to perform a task (imperative languages, such as Java, Python, and C), or specify the expected outcome of a task (declarative languages, such as Lisp and Haskel).

Markup languages are used to describe and structure data, guiding its presentation, processing, or interpretation. We will go through examples to aid in understanding.

Each cell in the following table contains a file that gives an example of some of todays most common markup languages. The first line of each file is a comment line, unnecessary to the file, that contains the name of the markup language.

/* JSON */
{
  "title": "The Great Gatsby",
  "author": "Fitzgerald",
  "year_published": 1925,
  "isbn": "978-0743273565",
  "price": 10.99
}
	<!-- XML -->
	<?xml version="1.0" encoding="UTF-8"?>
	<book>
	  <title>The Great Gatsby</title>
	  <author>Fitzgerald</author>
	  <year_published>1925</year_published>
	  <isbn>978-0743273565</isbn>
	  <price>10.99</price>
	</book>

# YAML title: "The Great Gatsby" author: "Fitzgerald" year_published: 1925 isbn: "978-0743273565" price: 10.99	**<!-- Markdown -->** # The Great Gatsby **Author**: Fitzgerald **Year Published**: 1925 **ISBN**: 978-0743273565 **Price**: $10.99

<!-- HTML -->
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>The Great Gatsby</title>
</head>
<body>
   The Great Gatsby
  Author: Fitzgerald
  Year Published: 1925
  ISBN: 978-0743273565
  Price: $10.99
</body>
</html>
	% LaTeX
	\documentclass{article}
	\begin{document}
	
	\title{The Great Gatsby}
	\author{Fitzgerald}
	\date{1925}
	\maketitle
	
	\section*{Book Details}
	\begin{itemize}
	  \item \textbf{Author}: Fitzgerald
	  \item \textbf{Year Published}: 1925
	  \item \textbf{ISBN}: 978-0743273565
	  \item \textbf{Price}: \$10.99
	\end{itemize}
	
	\end{document}

Each of these formats have a different primary use cases. The first three – JSON, XML, YAML – are used to structure data. The latter three – Markdown, HTML, and LaTeX – are used for describing how to format the data for display. The uses are described in slightly more detail in the table, below.

	File Format		Primary use case
structure	JSON	JavaScript Object Notation	transmission of structured data between computers
	XML	eXtensible Markup Language	transmission of structured data between computers
	YAML	YAML ain't Markup Language *or* Yet Another Markup Language	software configuration files
format	LaTeX		formatting of academic papers and books because of its strength in describing complex text, including mathematical equations and scientific content
	HTML	HyperText Markup Language	formatting of web pages
		Markdown	formatting plain text for easy conversion to HTML (documentation, blog posts, etc.)

HyperText Markup Language (HTML)

HTML is the foundation of the web. All web pages use HTML to tell the browser how to format the web page. Let's create a simple web page, step-by-step.

First, find any text editor that can save a file in plain text. On Microsoft Windows, you might use NotePad. On a Mac, you can use TextEdit (but you might have to change the preferences such that it saves and displays in "Plain Text" rather than "Rich Text" format). Alternatively, but perhaps when you decide you will learn HTML more seriously, you can download VS Code, which will provide highlighting, syntax checking and other helpful features.

A Basic HTML Code Example

Create a new plain text file, enter contents similar to those in the figure below (but use your own name), and save it with an extension "`.html`". Remember where you save your file! For my file, I'll save it as "`chris.html`".

```
My Web Page!
My name is Chris.
```

Find your file in the file browser. If you double-click on the file, it should open and display in your default web browser. If your file opens in another application, check that your file has the extension "`.html`". As a workaround, you can also try to right-click on the file to select the application that should open the file.

If everything works fine, you can also consider making a copy of the file, renaming it to have the extension "`.txt`", and again open it in your web browser. (With a "`.txt`" extension, you will most likely need to right-click on the file and specify that you want to open it in your browser application). Notice that it will be displayed differently.

As is, our HTML file does not actually contain any "markup" to tell the web browser how to structure and display the text that is in the file. Let's add some. Update your file to add the text highlighted in bold.

```
<h1>My Web Page!</h1>
<p>My name is Chris.<p>
```

HTML markup is accomplished by adding ***tags*** to the file. An HTML tag is formed by enclosing the tag name within ***angle brackets*** (`< >`). Most tags have an opening tag (<tag>) and a closing tag (</tag>), with the ***content*** of the tag placed between them. An HTML ***element*** consists of the opening tag, the content between the tags, and the closing tag. In our file, we have enclosed the content "`My Web`

Page!" within an opening and a closing <h1> tag – the tag for a level one heading. This entire structure, <h1>My Web Page!</h1>, is an HTML *element*.

The second line has been enclosed within an opening and a closing <p> tag – the paragraph tag. Open your file in your browser again and not how the formatting of the text has changed.

Tags may also include ***attributes*** that provide additional information about the element. Attributes are included after the tag name in the opening tag, and are generally name-value pairs, with the name and value separated by an equals sign (=) and the value enclosed in double quotes. Add the attribute align="center" to the <h1> tag of your HTML file and check the results in your web browser.

```
<h1 align="center">My Web Page!</h1>
<p>My name is Chris.<p>
```

Although it is entirely possible to describe document formatting within HTML tags, it is best practice to use describe only the structure of the document in the HTML tags, and use external CSS to describe the formatting of the HTML elements. We won't discuss CSS here. But let's remove the attribute from the <h1> element and instead add another tag that requires use of attributes.

```
<h1>My Web Page!</h1>
<p>My name is Chris.<p>
<input type="text" placeholder="Enter your name">
```

Check out the results in your browser. Notice also that the <input> tag has no closing tag. It is called a ***void*** element.

Making a Proper HTML Document

Although your web browser is able to interpret the HTML code in the file we wrote, it isn't actually a proper HTML file. The following figure takes the HMTL code we created in the previous section and inserts into a basic HTML5 document. As with the Java programming language, most white space in HTML is only to make the code more human-readable.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0">
    <title>Chris' Web Page</title>
  </head>
  <body>
    <h1>My Web Page!</h1>
    <p>My name is Chris.<p>
    <input type="text" placeholder="Enter your name">
  </body>
</html>
```

Notice that we have ***nested*** HTML elements: some HTML elements (<*tag*>*content*</*tag*>) have been placed between the opening and closing tags of another element.

For example, almost the entire document is enclosed as the contents of the pair of <html> tags. Information about the document (the meta data) is contained within the pair of <head> tags, while the displayable contents of the file is placed within the pair of <body> tags. Among others, there must be only one pair of <html> tags, one pair of <head> tags, and one pair of <body> tags in a single HTML document.

Since all the displayable content of an HTML file is enclosed within the <body> element, and our new HTML document contains only the HTML code from the previous section, it will be displayed exactly the

same as that code. Do notice that the web browser tab (or title bar) should contain the HTML document `<title>` element. Using a proper HTML document for our web page will also allow us to incorporate other features in our document.

Scalable Vector Graphics (SVG)

Scalable Vector Graphics (SVG) is another markup language that describes two-dimensional vector images. SVG code is valid XML code. SVG is considered an "application" of XML. This means the formatting follows the XML standard, but the content is restricted to specific items defined in the SVG specification.

Add the code to your web document and view it in your browser.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport"
          content="width=device-width, initial-scale=1.0">
    <title>Chris' Web Page</title>
  </head>
  <body>
    <h1>My Web Page!</h1>
    <p>My name is Chris.<p>
    <p><input type="text" placeholder="Enter your name"></p>
    <svg width="10mm" height="10mm" xmlns="http://www.w3.org/2000/svg">
      <circle cx="5mm" cy="5mm" r="5mm" fill="blue" />
    </svg>
  </body>
</html>
```

SVG defines a number of tags that can be used withing the `<svg>` element. Below is an example SVG file that uses common shapes.

```
<svg width="10mm" height="10mm" viewBox="0 0 10 10"
xmlns="http://www.w3.org/2000/svg">
  <!-- Rectangle -->
  <rect x="1" y="1" width="3" height="2" fill="red" />

  <!-- Ellipse -->
  <ellipse cx="7" cy="3" rx="2" ry="1" fill="green" />

  <!-- Polygon -->
  <polygon points="2,5 4,7 6,5" fill="blue" />

  <!-- Polyline -->
  <polyline points="7,6 8,8 9,6" fill="none" stroke="orange"
stroke-width="0.2" />

  <!-- Path -->
  <path d="M1,9 L3,7 L5,9" fill="none" stroke="purple" stroke-
width="0.2" />
</svg>
```

Add to the SVG code in your HTML document to create an image or pattern. Be creative.

Where are Vector Graphics Commonly Used

Photographs and other images that do not have sharp contrast, or have details that are not expressed simply by mathematical formulas are commonly stored as raster images. Images created by digital cameras commonly use the JPEG format, and images on web pages are commonly JPEG or WebP format.

Diagrams and images with high contrast are ideal for vector graphics formats. A diagram saved as a vector graphics image will almost certainly take much less space than the same diagram saved as a raster image. The vector image also allows very clean scaling of the diagram.

One common use of vector graphics you're looking at right now. Almost all computer fonts are stored as vector graphics images. Have you noticed that it doesn't matter how large you scale the text in a word processor, it always looks very clear and never displays any blockiness?

Why is HTML and SVG Code So Similar?

A markup language named Standard Generalized Markup Language (SGML) was developed in the 1960's and finally standardized in 1986.

HTML was developed as an application of SGML, but it does not strictly follow the SGML standard.

XML is a simplified subset of SGML. SVG is an application of XML, and does strictly follow the XML standard.

Conveniently, HTML5 allows SVG code to be embedded directly into an HTML file using an `<svg>` element, as we have done in the previous section. The code within an `<svg>` element will be interpreted as an SVG image.

Graphical SVG Editors

The application named ***Inkscape*** is a free and open source vector graphics editor. Download and install this editor on your computer.

Copy only the `<svg>` element from your HTML document and save it into its own text file using the extension "`.svg`". This will be a valid SVG file that you can then open using Inkscape. Try this.

When using Inkscape to save an SVG file, if you wish to view the file with a text editor, save the file as a "Plain SVG" rather than an "Inkscape SVG". Saving as an Inkscape SVG file will make the file less human-readable because it results in a lot of additional data being saved to the file.