DIVS, SPANS, AND GROUPING THINGS

INTRODUCTION

Always remember, HTML is used to get our **content** onto the page in a somewhat logical order. If we need text, we use a tag for showing text (p, h1, h2, etc.) If we need images we use the img tag, and so on.

As we will start to learn, every HTML tag has default **css** styling. We will learn how to customize this stuff later, but for now I think you can recognize the difference between an **h1** tag and a **p** tag. One is large bold text while the other is smaller bland text.

Why am I talking about this default CSS? Because it is important to remember this when learning about the basic layout tags like **div** and **span.** These are the tags we will use to group our content together on the page!

DIVS & SPANS

Let's introduce the most basic layout/grouping tags HTML has to offer. The **div** and the **span** tags.

The div is a tag is one you will use most often when trying to group content together. In simple terms, if you ever have a few tags that make up an actually *thing* on your page, you probably want to group them using a div.

For example, if you have a basic website you might break it into 3 logical groupings:

1. The header
2. The content
3. The footer

Here it would make sense to throw these sections into their own div tags. This means all the tags for the header would be nested in a div tag designed to group together the header content.

DIVS & SPANS CONT.

An interesting thing to note about the div/span is that they don’t actually contain any content meant to be seen by the user. You can really think of a div/span as a box you **put around content tags.**

This allows you as a developer to keep yourself organized and keep things grouped together that should be grouped together and things that are not related not in the same grouping.

As we will see, if we do this correctly this also allows us to easily style our HTML code as it becomes easier to target the correct HTML with our style choices.

DIVS & SPANS CONT.

The div tag is considered a **block** element (which we will talk about later in the slides) but for now, just assume that a block tag is designed to fill as much horizontal space as possible.

You have all actually seen this in action on your sites up until this point. In your HTML, go and put three p tags one after the other. Notice how they all go on a new line by default? Now go and put 3 a tags one after the other. Notice how they all stay on the same line by default (if there is enough space)?

An inline element is not as greedy when it comes to space. It simply hugs any content within the span as tightly as possible. This is useful if you want a logical grouping to not go onto new lines.

INLINE VS BLOCK

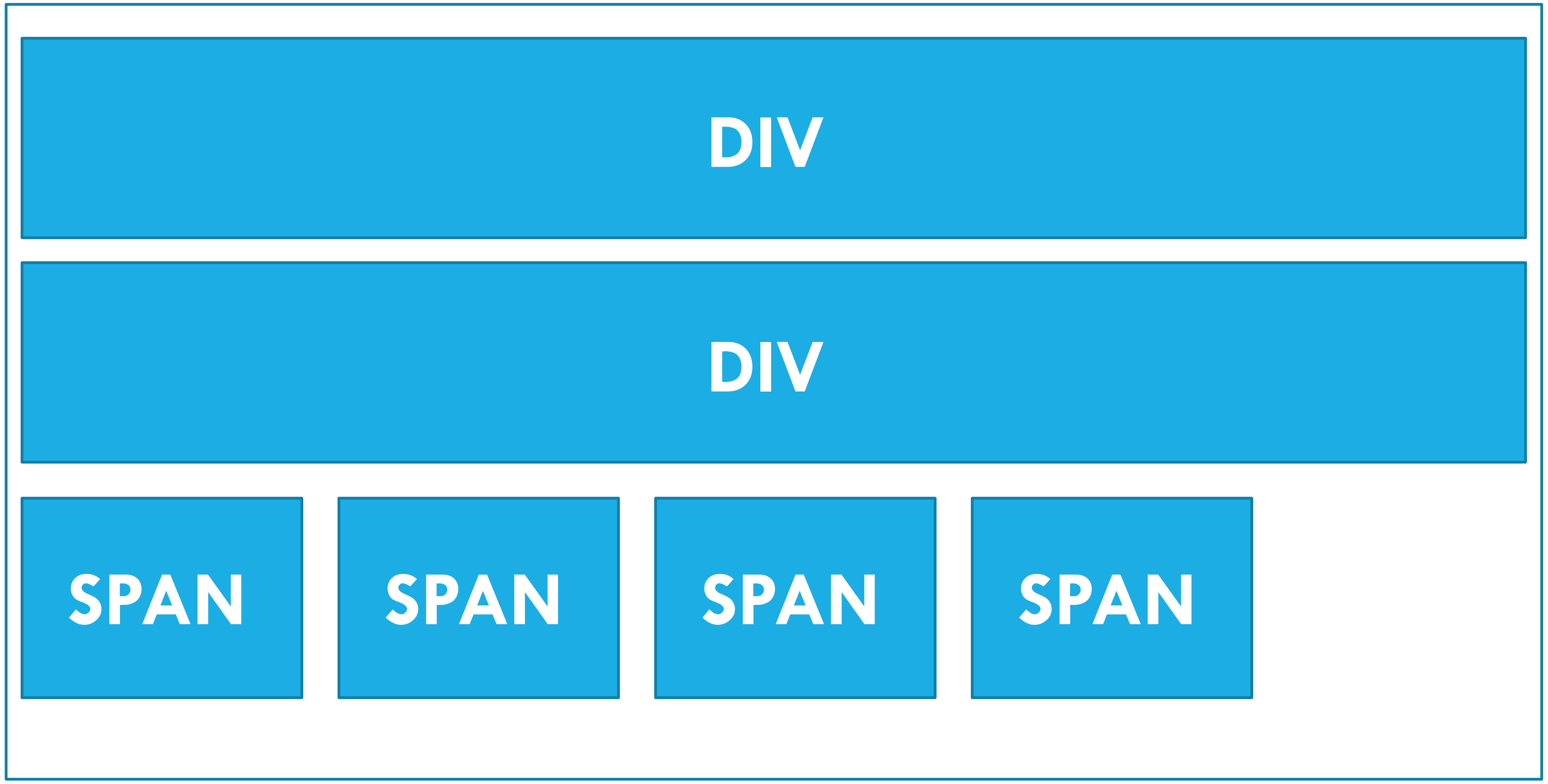
Almost all HTML elements fall into one of two categories being block or inline elements by default.

A block element is designed to take up the **whole width** of the screen (but only the height of its tallest child element) and force a new line.

An inline element will only take up as much width (and height) of its largest child element and will not force a new line. What this means is a block element will be on its own line, while an inline element will allow other elements to align beside them.

An element being **block** vs **inline** is actually just a CSS property called **display**. What's neat about this is the fact that you can actually change it with your own CSS! For now we will just use the defaults given to us.

DIVS VS SPANS



NESTING AND LAYOUT

Nesting HTML tags that have different display types can get a bit tricky.

As a general rule of thumb (but not a concrete rule), know that:

- You **can** nest inline elements inside of block elements with ease and as much as you want
- You **can't** nest block elements inside of inline elements as it defeats the purpose of the original inline element.

Why would this be?

GOOD NESTING

An example of **good** nesting:

```
<div>
 <h1>What am I learning?</h1>
 <p>I am learning <span>HTML AND CSS</span></p>
</div>


<span>
 <a href="#">One Link!</a>
</span>


<span>
 <a href="#">Two Link!</a>
</span>
```

BAD NESTING

An example of **bad** nesting:

```
<span>
  <h1>What am I learning?</h1>
  <p>I am learning <div>HTML AND CSS</div></p>
  <div>
    <a href=“#”>One Link!</a>
  </div>
  <div>
    <a href=“#”>Two Link!</a>
  </div>
</span>
```

ORGANIZING YOUR PAGE

We are now coming to a point with HTML where something interesting can occur. **Your code can be correct, but still bad**.

One thing that people new to code always forget is that a project you write is not a static thing. It changes over time, you get new ideas, want to add or change things. This is unavoidable and actually a really good thing.

If you write a messy and unorganized heap of HTML, that will cause you to write a messy and unorganized heap of CSS. Chances are you will still be able to get things to work, but it will almost impossible to change things when you want to!

Organizing yourself will make your code easier to work with and will make learning easier as you will be able to make sense of your code.

UNORGANIZED CODE

```
<body>

 <h1>Welcome to my site!</h1>

 <p>You will find all kinds of cool things to do here, so don't be shy! Click around and have a good time.</p>


 <h2>Summer Vacation</h2>

 <img src="lake.jpg" alt="A sunset over a calm lake">

 <p>This is the lake I went to. Pretty sweet I know.</p>


 <h2>Work Days</h2>

 <img src="work.jpg" alt="Man sitting at a desk writing code">

 <p>This is my typical work day. Pretty sweet I know.</p>



 <button>

  <a href="mailto:info@me.com">Contact Me</a>

 </button>

 <button>

  <a href="tel:4035555555">Call Me</a>

 </button>

</body>
```

ORGANIZED CODE

```
<body>

  <div>

    <h1>Welcome to my site!</h1>

    <p>You will find all kinds of cool things to do here, so don't be shy! Click around and have a good time.</p>

  </div>
```

ORGANIZED CODE CONT.

```
<div>

  <div>

   <h2>Summer Vacation</h2>

   <img src="lake.jpg" alt="A sunset over a calm lake">

   <p>This is the lake I went to. Pretty sweet I know.</p>

  </div>
  <div>

   <h2>Work Days</h2>

   <img src="work.jpg" alt="Man sitting at a desk writing code">

   <p>This is my typical work day. Pretty sweet I know.</p>

  </div>

</div>
```

ORGANIZED CODE CONT.

```
  <div>
    <button>
      <a href="mailto:info@me.com">Contact Me</a>
    </button>
    <button>
      <a href="tel:4035555555">Call Me</a>
    </button>
  </div>
</body>
```

BAD VS GOOD

Depending on how comfortable you are with HTML at this point you might feel that the second example was actually harder to understand and less organized because of all the extra code we added.

I understand where that feeling is coming from, but trust me the structure added with the divs make this much more organized.

Think about cleaning your room, yes technically you could just lay out all of your clothes and you would be able to get dressed every morning. You could also organize your clothes and keep them in a logical grouping and ordering in your closet to keep yourself sane.

KNOWLEDGE CHECK

Let's make a standard 3 tier website. We will have a header, content and a footer.

1. Create a new folder called HtmlOrganization inside your Scratch folder.
2. Turn this folder into a Git repository and connect it to GitHub
3. In your index.html file, create the following:
    1. A div a the top of the site that contains a title and welcome message
    2. A div in the middle of the site that contains any content you want (images, videos, text, iframes, etc.). This div must contain at least 3 divs inside of it.
    3. A div at the bottom of the site that contains a contact button for email and phone (they don't need to actually work).
4. Add commit and push your code.

Remember, our use of HTML is first and foremost supposed to get the content we want onto the page.

When we introduce tags like div and span we introduce a couple new ideas like **basic layout** and **logical grouping**.

I strongly recommend using divs and spans more on the grouping side of things as we can use CSS to change the layouts of our pages later.