

girls who  
code

# Girls Who Code At Home

Digital Memory Book  
Responsive Web Design with Flexbox

## Activity Overview

Have you ever wanted to make your own website to store your memories? Are you looking for a way to create a digital yearbook for you and your friends? If so, this activity is for you! Digital media like photos, audio, and video are important ways of remembering the people, events, and places that we love. In this activity, we will be mainly coding in HTML and CSS to build and design your digital memory book. First, we will use HTML to add and organize the photos and text of each memory, then we will use Flexbox, a tool in CSS, to design each memory and organize all of the memory cards into a grid layout.

**Flexbox** is a tool that web developers use to arrange and position visual elements on a website like images or boxes of text. Flexbox is especially helpful to create more **responsive** website layouts that allow elements on a page to resize and adjust their layout in order to fit different screen sizes. Responsive design has become increasingly important as more and more people access websites from smaller screens like phones and tablets.

**Note:** We will not be coding in Javascript for this activity. Check out our [Virtual Hike Activity](#) to learn more about Javascript and how developers use it to add interactivity on a website. If you want a quick refresher on HTML and CSS we highly suggest you check out our third activity: [Share Your Skillz](#) before starting this activity.

## Materials

- [Glitch](#) or the text editor of your choice
- [Digital Memory Book Starter Code](#)
- [Example Digital Memory Book \(no Extensions\)](#)
- [Example Digital Memory Book \(with Extensions\)](#)
- Optional: Planning Guide (page 18)
- Optional: Pen/Pencil/Markers

## Women in Tech Spotlight: Nancy Douyon



Image Source: [Medium](#)

Nancy Douyon has worked in the tech industry for 18 years. She currently works for Google as a UX, or User Experience, designer. UX designers are responsible for designing or thinking about the entire user experience of a product. This can include how a user finds a product, what it's like to use the product, and how they feel before, during, and after using the product.

In 2017 Nancy launched the Tech Social Impact Conference, which explores the wider implications of technologies in society, and sparks conversation about how design and technology can provide social and ethical benefits to the whole world. You'll learn about Nancy, the Tech Social Impact Conference, and watch an interview where she talks about what it's like to be a woman in tech.

Watch [this interview](#) to listen to her resistance to moving to Silicon Valley and the changes she'd like to see in the tech industry to make it more inclusive. If you have more time read this [Techies feature](#) of Nancy.

### Reflect

There's more to being a computer scientist than just being great at coding. Take some time to reflect on how Nancy and her work relates to the strengths that great computer scientists focus on building - bravery, resilience, creativity, and purpose.



**PURPOSE**

What responsibility do developers and designers have to think about the social impact of their products?

Share your responses with a family member or friend. Encourage others to read more about Nancy to join in the discussion!

## Step 1: Brainstorm Your Memory Book (15 mins)

In this activity you will be making a digital memory book. You will include a picture and text for each memory card, then lay these cards out in a grid to capture all your memories in one place. Take 5 minutes to explore some features of the [sample project](#) that we created.

As you explore the website think about the following questions:

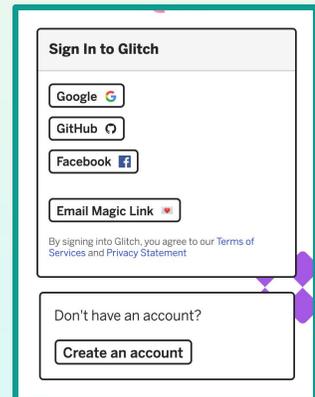
- **What do you want your memory book to be about?** This can be a memory book about a club, friend group, a creative project, or a family trip.
- **What images do you want to include?** Pictures can be of people, scenery, art, etc.
- **What kind of text will you write for each photo?** Do you want to include quotes for each photo, descriptive text, names of the people included, hashtags?

Use the planning document on **page 18** to help you brainstorm and plan your memory book before getting started.

## Step 2: Get Started with Glitch (5-10 mins)

Glitch is a simple tool for creating web apps. It comes with a text editor that allows you to see edits to your webpage in real time. It allows you to publish your project easily for the world to see! If you find a cool project someone else made, you can look at their code and remix it.

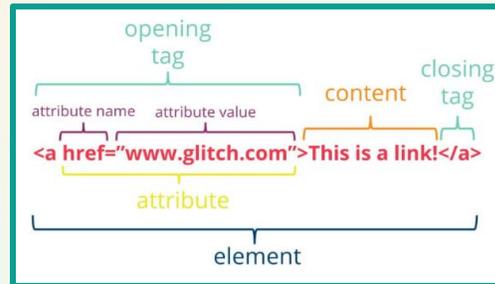
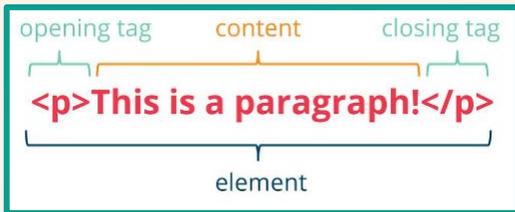
1. **Sign into [Glitch](#) using your Google, Facebook, or GitHub account.**
  - To save and share your work on Glitch, you will need to sign in. Glitch allows you to sign in using a Google, Facebook, or GitHub account. If you are under 13, you'll need your guardian's permission and email address to sign up.
2. **Remix this [starter code](#) for your digital memory book.** Watch this [video](#) for more detailed instructions.
3. **Explore the Glitch Interface.** The first thing you will see are the contents of the **README .md** file. READMEs are documents that you will constantly see in other people's projects. They usually contain information about how to navigate through the project's files and how to run the program. Watch this [video](#) to learn how to edit the README file.
4. **Click the *Show* button and choose *Next To The Code* option.** This will allow you to view your code and how the project looks on the web side-by-side. Watch this [video](#) for more detailed instructions.



You will see a lot of different elements in the starter code already written for you. In this activity, your goal is to learn how to use these existing web elements and lay them out or arrange them using CSS and Flexbox. If you're interested in learning more about the basics of HTML and CSS, check out our [Share Your Skillz Activity](#).

## Step 3: Explore the Starter Project HTML File (5-15 mins)

Let's take a look at the `index.html` file first. Click on the `index.html` file from the left navigation menu. **HTML**, short for Hypertext Markup Language, uses **tags** to organize content for a website. All of the content that shows in your live preview are contained within a `<body>` opening tag and `</body>` closing tag. Tags are part of the anatomy of an HTML element, which are usually made up of the opening tag, content, and the closing tag. Take a look at the images below to see some examples of HTML tags for different types of elements. The `<p>` tag is used to note "paragraph" while the `<a>` tag is more of a generic tag typically assigned for links.

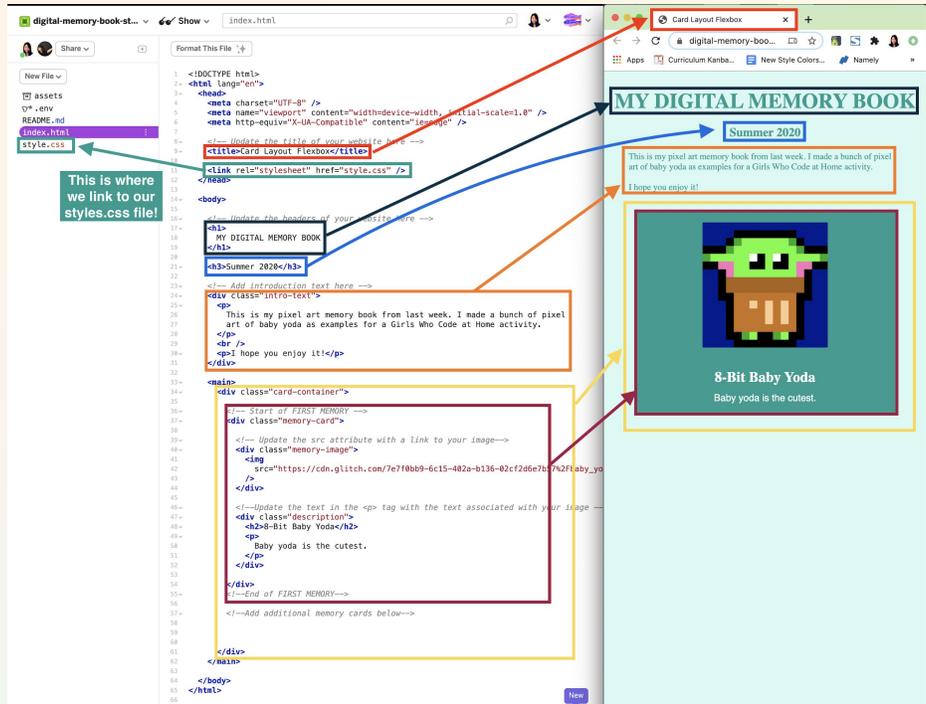


Here are the tags you will see in this project:

Tag	Use	Syntax	Example
<code>&lt;h1&gt;</code> <code>&lt;h2&gt;</code> <code>&lt;h3&gt;</code>	Use these tags to create headings that distinguish between information sections.	<code>&lt;h1&gt;Overview&lt;/h1&gt;</code> <code>&lt;h3&gt;Materials&lt;/h3&gt;</code>	<b>Overview</b> <b>Materials</b>
<code>&lt;div&gt;</code>	Use the <code>&lt;div&gt;</code> tag to define a division or section in an HTML file. We use <code>&lt;div&gt;</code> tags to organize and group elements together.	<code>&lt;div class="intro-text"&gt;</code> <code>&lt;p&gt; This is my memory book&lt;/p&gt;</code> <code>&lt;br /&gt;</code> <code>&lt;p&gt;I hope you enjoy it!&lt;/p&gt;</code> <code>&lt;/div&gt;</code>	<div style="border: 1px dashed black; padding: 5px;"> <p>This is my memory book</p> <p>I hope you enjoy it!</p> </div>
<code>&lt;p&gt;</code>	Use the <code>&lt;p&gt;</code> tag for paragraphs of text.	<code>&lt;p&gt;Puppy kitty ipsum dolor sit good dog stick canary.&lt;/p&gt;</code>	Puppy kitty ipsum dolor sit good dog stick canary.
<code>&lt;a&gt;</code>	Use the <code>&lt;a&gt;</code> tag for hyperlinks.	This is a <code>&lt;a href="www.glitch.com"&gt;</code> link <code>&lt;/a&gt;!</code>	This is a <a href="#">link!</a>
<code>&lt;img&gt;</code>	Use this tag to include images.	<code>&lt;img src="kitten.png" alt="kitten"&gt;</code>	

# Step 3: Explore the Starter Project HTML File (Continued)

The image below maps the tags that you see in your starter code to how they look live on your website.



You may notice on the index.html file there are some lines of code that are grayed out. These lines of code are called **code comments**. Code comments are used by a programmer to help leave notes and organize code so that is easier for other people to read. Here we use code comments to highlight parts of the instructions. In HTML any code that starts with `<!--` and ends with `-->` are code comments.

## Try Coding in HTML Yourself!

1. **Change the Title of your website.** The title of your website is the name that pops up when you load your website.



**Note:** this is not the same as changing the URL of your website. If you want to check that the title of your webpage is correct, click **Show** and choose the **New Window** option in the menu.

- o Locate the `<title>` tag in the HTML file. (This is outlined in the red box in the image above).
- o Replace the text between the opening tag `<title>` and closing tag `</title>` with the name of your webpage.

Ex. `<title> My Pixel Art Memory Book </title>`

## Step 3: Explore the Starter Project HTML File (Continued)

- 3. Change the intro text on your website.** In the sample code we created a section using a `<div>` tag with the class name "intro-text". (This is outlined in the light orange box in the image above). The `<div>` tag is used to define a division or section in an HTML file. We use `<div>` tags to organize and group elements together. You can even assign `<div>` tags a class name so you can distinguish them from each other and customize them individually in your CSS - more of that in the next section. In this section there are two paragraphs to introduce and describe the website. These paragraphs are separated by a `<br />` tag which helps us add a space between the elements.
  - Locate the `<p>` tag in the HTML file. (This is outlined in the orange box in the image above)
  - Update the text in the `<p>` to introduce your website. It is up to you if you want to include more or less paragraph elements in your intro-text section. You can use the `<br />` tag to add breaks or spacing between the elements.

## Step 4: Explore Starter Project CSS File (5-10 mins)

**CSS**, or Cascading Style Sheets, describes what kinds of presentation rules – or styles – should be applied to HTML elements. For example, if you wanted to make the text larger or smaller you would do that using CSS. CSS allows you to change the way content on your website looks, including text color, text size, the fonts used, background colors or images, and so much more. For this activity, instead of adding all the CSS, we have included a lot of it for you so we can focus on styling the layout of your images and memory cards. We encourage you to view [this](#) resource or try out our [Share Your Skillz](#) activity if you are interested in learning more about CSS. Let's quickly review the structure of a CSS rule sets to style specific elements. Onwards!

**CSS creates rules for presenting content through rule-sets.**

A **rule-set** is made up of:

- a **selector** that determines where the styles are being applied. For more on selectors, see [this](#) reference.
- a **declaration block**, contained between `{}`s, that lists all of the individual styles to be applied. These individual style declarations have a property name, which corresponds with a value. Each declaration is separated by a `;` or semicolon.

```
selector  property name  property value
  ↓      ↓              ↓
h1 {     font-style: italic;
        color: #373fff;
}
```

The diagram shows a CSS rule set for the `h1` selector. The selector `h1` is labeled as the "selector". The opening curly brace `{` is labeled as the "declaration block". The property `font-style: italic;` is labeled as the "property name" and the value `italic;` is labeled as the "property value". The property `color: #373fff;` is also labeled as the "property name" and the value `#373fff;` is labeled as the "property value". The closing curly brace `}` is labeled as the "declaration block".

## Step 4: Explore the Starter Project CSS (Continued)

### Rule Sets in `style.css`

Let's take a quick look at the code already written in `style.css`. You may notice that there are some lines of code that are grayed out. Similar to the grayed out lines of code in `index.html`, these lines of code are also **code comments**. In CSS, code comment start with `/*` and end with `*/`.

In the first section, labeled **RESET VALUES**, we include a rule-set to reset the margins, padding, and box-sizing. You may notice that this rule-set contains a `*` selector. A `*` selector applies the styling rules to ALL elements, regardless of type, class, or id. Website browsers (Chrome, Firefox, etc.) may have different default margins and padding sizes so this rule-set ensures that our content looks the same across all browsers.

In the next section, labeled **BODY & HEADER**, we included styling to set the background color for the whole body of the website and font styles for the headers. Here we use the tag selectors (e.g. `<h1>`) to apply the style rules on all header elements.

In the next section, labeled **MEMORY CARD**, we included styling for each component of the memory cards. You may notice that these selectors look slightly different. Here we use the class selector to style each `<div>` container separately. Remember that in the `index.html` file, we assigned a class attribute to each `<div>`. We can use this class attribute as a class selector in the `style.css` file and link the HTML element to a CSS rule-set. When using a class selector we must add a period, `.`, before the name of the class. This applies the style rules to a group of elements with the same class attribute.

Finally in the last section, labeled **FLEXBOX STYLES**, you may notice there are no style rule-sets written. That's because you'll be learning about what Flexbox is, why it is helpful in styling a website, and writing some Flexbox styles through this activity. In the next steps you will add CSS rule sets to this section of the code to incorporate Flexbox style rules and create a responsive layout.

### RESET VALUES

```
* {  
  margin: 0;  
  padding: 0;  
  box-sizing: border-box;  
}
```

### BODY & HEADER

```
body {  
  background: #D7F9F4;  
}  
  
/* Header Styles */  
  
h1 {  
  text-align: center;  
  color: #0d9c90;  
  font-size: 35px;  
  padding: 30px 15px 15px 15px;  
}  
  
h3 {  
  text-align: center;  
  color: #0d9c90;  
  font-size: 20px;  
  padding: 5px;  
}
```

### MEMORY CARD

```
main {  
  width: 100%;  
  /*border-style: dotted;*/  
}  
  
.card-container {  
  width: 90%;  
  margin: 0 auto;  
  padding: 0.5em 0;  
  /*border-style: dotted;*/  
}  
  
.intro-text {  
  width: 90%;  
  margin: 0 auto;  
  padding: 0.5em 0;  
  color: #0d9c90;  
  font-size: 11pt;  
  padding: 1em 1em;  
  /*border-style: dotted;*/  
}
```

## Step 5: Adding Your First Memory (10-15 mins)

Locate the HTML code for your first memory card. This is between the comments that read `<!-- Start of FIRST MEMORY -->` and `<!--End of FIRST MEMORY-->`. Let's break down the structure of a memory card.

```
37 <!-- Start of FIRST MEMORY -->
38 <div class="memory-card">
39
40 <!-- Update the src attribute with a link to your image-->
41 <div class="memory-image">
42 
45 </div>
46
47 <!--Update the text in the <p> tag with the text associated with your image -->
48 <div class="description">
49 <h2>8-Bit Baby Yoda</h2>
50 <p>
51   Baby yoda is the cutest.
52 </p>
53 </div>
54
55 </div>
56 <!--End of FIRST MEMORY-->
```

- Each memory card is grouped together by a `<div>` tag with a class attribute `"memory-card"`.
- Inside the memory-card `<div>` container are two other `<div>` tags, one with class attribute `"memory-image"` and the other with class attribute `"description"`.
- The `"memory-image"` `<div>` container has one element, an `<img>` element with a `src` attribute. The `src` attribute contains the image URL address. This is how the website gets the specific image that we want.
- The `"description"` `<div>` container has two elements, a `<h2>` element with a title and a `<p>` element with a short description of the image.

### Adding Your First Image

Break out your planning document - it's time to add in your first memory! The first thing we need to do is get the image URL address of our images.

- **If you are using an image from the web, locate the image URL address.** You can watch this [video](#) for instructions for how to find the image URL address.
- **If you are using your own images, be sure that your images are saved *digitally* on your computer.** To upload images to Glitch you can watch this [video](#).

Now that you have your image URL address, locate the `<img>` tag in the `"memory-image"` `<div>` container. Replace the text after `src=` in the `<img>` tag. When you paste your image URL address, be sure that you include double-quotation marks `" "` around your image URL address.

```
<!-- Start of FIRST MEMORY -->
<div class="memory-card">
  <!-- Update the src attribute with a link to your image-->
  <div class="memory-image">
    
  </div>
  <!--Update the text in the <p> tag with the text associated with your image -->
  <div class="description">
    <h2>8-Bit Baby Yoda</h2>
    <p>
      Baby yoda is the cutest.
    </p>
  </div>
</div>
<!--End of FIRST MEMORY-->
```

Replace this text with the URL address for your image. Be sure to include " " around the URL address.

## Step 5: Adding Your First Memory (Continued)

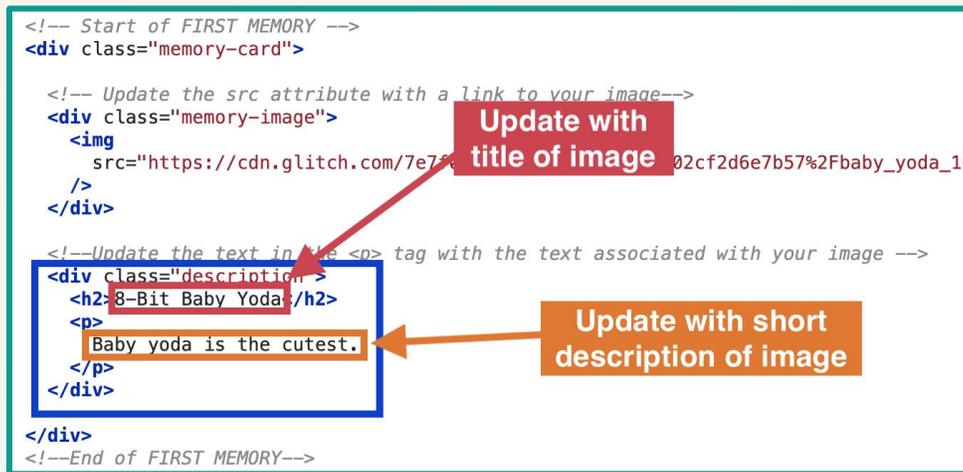
Look back at your live website, did the image change? If you see no changes, double check that you have the correct URL address. Copy and paste the address in a new window and see if your image pops up. Also, make sure that there are double quotation marks around your URL address in your HTML file.

### Adding Your First Image Title and Description Text

Refer back to your planning document. Locate the “description” <div> container and its two elements, the <h2> and <p> elements. Update the text in the <h2> tags with the title of the image and update the <p> tag with a short description of the image.

```
<!-- Start of FIRST MEMORY -->
<div class="memory-card">

  <!-- Update the src attribute with a link to your image-->
  <div class="memory-image">
    
    <h2>8-Bit Baby Yoda</h2>
    <p>
      Baby yoda is the cutest.
    </p>
  </div>
</div>
<!--End of FIRST MEMORY-->
```



Look back at your live website, did the text change? If you do not see any changes, make sure that your text was added *between* the opening and closing tags for each element. For example, the title of the image should be between the <h2> and </h2> tags and the description should be between the <p> and </p> tags.

## Step 6: Adding More Memory Cards (10-15 mins)

Now that you added your first memory card with an image, title, and description text, it is time to add more to your memory book.

For each memory, **copy** the code for the entire first memory card. Remember, the code for the first memory is between the comments that read <!-- Start of FIRST MEMORY --> and <!--End of FIRST MEMORY-->. **Paste** the code right below the first memory and update the code comments to say <!-- Start of SECOND MEMORY --> and <!--End of SECOND MEMORY-->.

## Step 6: Adding More Memory Cards (Continued)

```
<main>
<div class="card-container">
  <!-- Start of FIRST MEMORY -->
  <div class="memory-card">
    <!-- Update the src attribute with a link to your image-->
    <div class="memory-image">
      
    </div>
    <!--Update the text in the <p> tag with the text associated with your image -->
    <div class="description">
      <h2>8-Bit Baby Yoda</h2>
      <p>
        Baby yoda is the cutest.
      </p>
    </div>
  </div>
  <!--End of FIRST MEMORY-->
  <!--Add additional memory cards below-->
  <div>
    <!-- Paste code and update for additional memory cards here -->
  </div>
</main>
```

Copy all code for first memory card

Paste code and update for additional memory cards here

Follow the instructions in **Step 5** to add your images and text for your additional memories. Be sure to check and test your code by viewing your live website to confirm any changes.

Let's take a look at what our page looks like now with all of our memory cards! Right now you may notice that your website is very loooong. This is because all of our memory cards just appear after each other. Let's experiment on how to style our website into a GRID layout instead to make the organization of our page look even nicer! To do this, we need a little tool called **Flexbox**.

## Step 7: What is Flexbox? (2 mins)

**Flexbox** is a tool that web developers use to more easily arrange and position visual elements on a website. Developers use Flexbox to create more **responsive** website layouts that are designed so that elements on a page can resize and adjust their layout to fit different screen sizes. Responsive design has become increasingly important as more and more people access websites from smaller screens like phones and tablets.

In Flexbox, we use a `<div>` tag to define a container for the elements we want to make responsive. A **flex container** can shrink items it contains to prevent overflow or expand items to fill free space on a webpage. The elements included in the flex container are considered **flex items**. Styling on a flex container will affect all items inside the container, but you can also add properties on the individual flex item to allow more customization in the layout. But how do we create a flex container and flex items?

## Step 8: Creating a flex Container (10-15 mins)

Let's take a look back at `index.html` and remind ourselves of the `<div>` containers in our code.

- **card-container:** This `<div>` container with a class name `card-container` holds the code for ALL memory cards.
  - This `<div>` will be our **flex container**.
- **memory-card:** This `<div>` container contains the code for **one** memory card. Inside this `<div>` container are the containers for the image and description of the memory.
  - Elements with this `<div>` will be our **flex items**.
- **memory-image:** This `<div>` container contains the code for the image in a memory card.
- **description:** This `<div>` container contains the code for the title in a `<h2>` tag and a short description in a `<p>` tag.

All Flexbox styling is coded in the CSS file. Let's navigate to `style.css` and scroll to the bottom to locate the section labeled `FLEXBOX STYLES`.

Let's create the flex container first. Since we want all memory cards to be organized in a specific layout, we want to make sure the container for ALL memory cards (the `<div>` container with class name `card-container`) contains the flex property.

We need to create a new ruleset that uses the class selector to select the `card-container` class and set the property `display` to the value `flex`. By adding the rule, `display: flex;` this turns our `<div>` container into a flex container. Remember that when using a class selector we must include a `.` before the class name.

```
108 ~ /*****  
109 ~ /***** FLEXBOX STYLES *****/  
110 ~ /*****  
111  
112 ~ /* Add flexbox styles here! */  
113 ~ .card-container{  
114     display: flex;  
115 }
```

Take a look at your live website, what do you see? Did anything change?

You should see something similar to what is displayed in the image on the right, where all of your memory cards are now lined up in one row. Experiment changing the size of your window. You should notice that the images and text descriptions resize proportionally with the size of your window. How cool!



If you added a lot of cards to your memory book you may notice that having all of your memory cards on one line isn't great. It's too squished!

## Step 8: Creating a flex Container (Continued)

Now that we have our flex container, let's explore some properties for flex containers.

- **flex-direction:** This establishes the main direction that flex items are placed in a flex container. Flex items can be laid out in horizontal rows or vertical columns. The flex-direction property can be set to row, row-reverse, column, or column-reverse.
- **flex-wrap:** Flex items will try to fit on one line by default. Sometimes you may want to change this and allow items to wrap and position items on the next line. The flex-wrap property can be set to nowrap, wrap, or wrap-reverse.

There are so many more flex container properties. To learn more about the Flexbox properties and their values take a look at this [CSS Tricks' A Complete Guide to Flexbox](#). We also recommend playing [Flexbox Froggy](#) for some fun practice with Flexbox properties.

Let's start adding more styling to our flex container to get our **grid** layout. In your card-container rule-set, add these additional styling rules:

- Add **flex-direction** property and set it to row.
- Add **flex-wrap** property and set it to wrap. This rule will allow your memory cards to overflow to a new row. This is great when you have a lot of memory cards on your website so that it is not squished on one row!

```
.card-container{  
  display: flex;  
  flex-direction: row;  
  flex-wrap: wrap;  
}
```

Let's go back and now check our website. You may notice that not much has changed, but why? We laid out the groundwork for our flex container and items, but there is still more work to be done! Now that we have styled our flex container it is time to style the flex items!

## Step 9: Styling Each Memory Card (5-15 mins)

Flex items also have their own properties we can apply to them in addition to the properties we can use on the flex container. Let's take a look at what they are:

- **flex-grow:** This defines the ability for a flex item to grow if necessary to fill the flex container. The property can accept unitless values and has a default value of 0.
- **flex-shrink:** This defines the ability for a flex item to shrink if necessary. It also accepts unitless values and has a default value of 1.
- **flex-basis:** This sets the initial size of the flex item before space is distributed according to the rules in the flex container. The flex-wrap property can be set auto, which defaults to the element size, or a specific size using px or em unit, or even a percentage to make size proportional to the window size.

To learn more about the Flex item properties and their values take a look at this [CSS Tricks' A Complete Guide to Flexbox](#).

## Step 9: Styling Each Memory Card (Continued)

Let's start styling our flex items, but which one? We want to add styling rules to each memory card. We can easily do this by adding a new rule set using the class selector on all memory-card <div> containers.

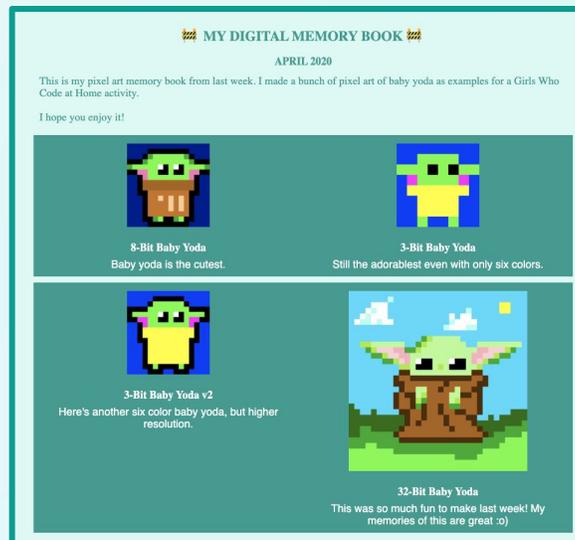
- Create a new rule-set at the bottom of your style.css file that uses the class selector on `memory-card` elements.  
**Note:** Since `memory-card` is a flex item in the `card-container`, we don't need to use the `display` property here. Setting the `display` property to `flex` is only used for flex containers.
- Add the `flex-grow` property and set it to value `1`. This will distribute the space equally as we increase the browser size.
- Add the `flex-shrink` property and set it to value `0`. We want to set the value `0` so that we preserve the original size of our image. By setting the `flex-shrink` property to `0` this restricts the browser and image from becoming smaller than its original size.
- Add the `flex-basis` property and set it to value `50%`. To create our grid layout we decided to only have two cards per line. Since we want only two cards per line, we want *each* memory card to take up only 50% of the browser space. If we wanted three cards per line, we would set the `flex-basis` property to `33%` instead.

```
/* Add flexbox styles here! */  
.card-container{  
  display: flex;  
  flex-direction: row;  
  flex-flow: wrap;  
}  
  
.memory-card{  
}
```

```
.memory-card{  
  flex-grow: 1;  
  flex-shrink: 0;  
  flex-basis: 50%;  
}
```

Let's take a look at our website now. Experiment making your browser larger and smaller. Do your images grow as you make your browser larger? Do two memory cards share one row? Does your browser stop shrinking if you try to make it too small?

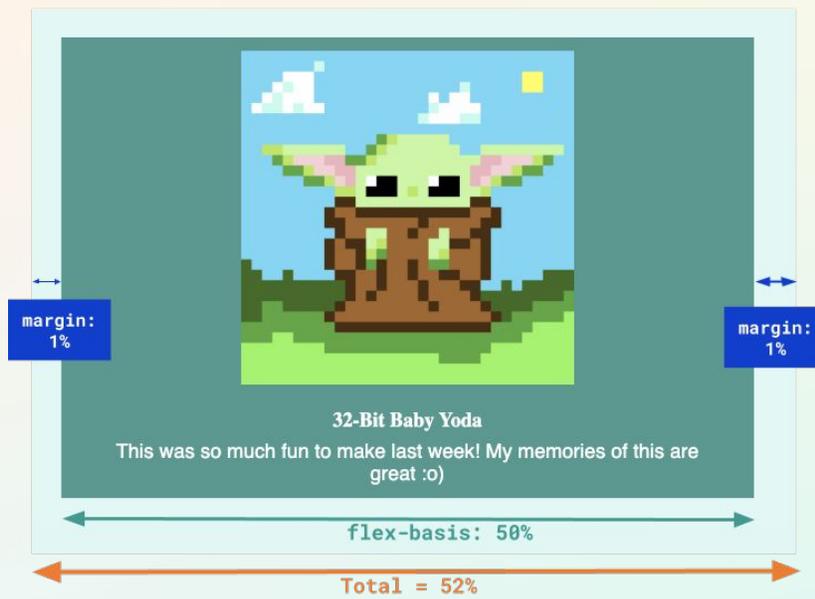
You may notice that there is a small divider between your two rows, but not between the columns. Add one more property to your `memory-card` that sets `margin` to `1%`. This will allocate 1 percent all around to space in between the memory cards. It is up to you if you would like to make the spacing larger or smaller.



## Step 9: Styling Each Memory Card (Continued)

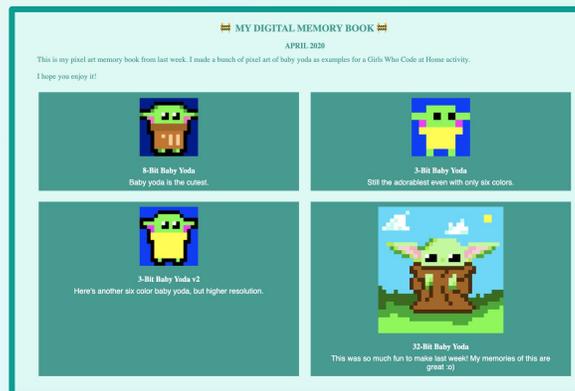
Let's check back to see if we have spacing around all of our images. Whoops! Why do we no longer have two memory cards sharing one row? This is a simple math problem!

Now that we allocated 1% of space on the left side, 1% space on the right side of the memory card, and 50% of the space for the memory card itself, each memory card actually takes up 52%! If we have two memory cards they take up 104% of our row which is why the memory card overflows to the next row.



To accommodate our margins, we need to subtract 2% from our value in flex-basis. Now let's set the `flex-basis` property to 48% instead.

```
.memory-card{  
  flex-grow: 1;  
  flex-shrink: 0;  
  flex-basis: 48%;  
  margin: 1%;  
}
```



Perfect! We have successfully created a responsive grid layout for all our memory cards!

## Step 10: Extensions (5-30 mins)

### Add Some Fancy Fonts (5-10 mins)

In browsers like Chrome and Firefox the default fonts are Times New Roman or Ariel, but you can change this for your own project! You can easily import and use fancy fonts in your website from [Google Fonts](#). Watch this [video](#) to learn how to change the fonts on your website!

### Change Background Colors (5-10 mins)

Let's change the background colors with more interesting colors! Colors are given numerical values to make it easier for the computer to distinguish different colors. They can be represented by their hex, rgb, or hsl value. We recommend choosing one type for all colors to be consistent. W3Schools has a great [color picker](#) to help you find the values of colors you might want to add to your website! Want to add a cool gradient effect like our sample project? Use this [website](#) to help plan and generate the CSS code. Watch this [video](#) for detailed instructions on changing colors.

### Research More CSS Styles! (10-20 mins)

You can do so much with CSS to make your website unique! We compiled a list of resources that you can browse for more tips on CSS.

- W3Schools has a great [reference](#) section for CSS.
- Mozilla has a wealth of helpful information in their CSS module, including:
  - [How to style text](#)
  - [CSS Layout](#)
  - [How to use CSS to solve common problems](#)
  - [Debugging your CSS](#)

### Add @media Rules to Make your Website Responsive! (15-20 mins)

Responsive web design means you style websites to change based on the size of the screen. Have you ever viewed a website not built for mobile? It's usually impossible to see or click the buttons! We can do this by creating @media rules that only include CSS properties if a certain condition is true (like a screen being larger than 700 pixels so we know it's not a smartphone). All of the CSS we have now is designed to work with smaller screens, so we need to add styling for larger screens. Check out this tutorial from [Mozilla](#) or this one from [W3 Schools](#) to get started.

### Add Emojis In Your Text (5-10 mins)

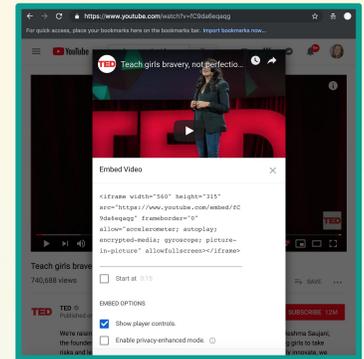
You can add emojis as part of your text by simply adding the `&#` symbol, followed by a number reference to your emoji, and finally ending with a `;`. Each emoji has a different number that you find [here](#). In any text element tags, you can add this code to display an emoji!

## Step 10: Extensions (Continued)

### Embed YouTube videos into your site. (5-10 mins)

Embedding YouTube videos is a fun way to make your website interactive.

- Navigate to your favorite YouTube video
- Press the "**SHARE**" button under the video
- Select the "**Embed**" option
- Copy the `<iframe>` HTML element given by YouTube and paste it into your HTML file. That's it!

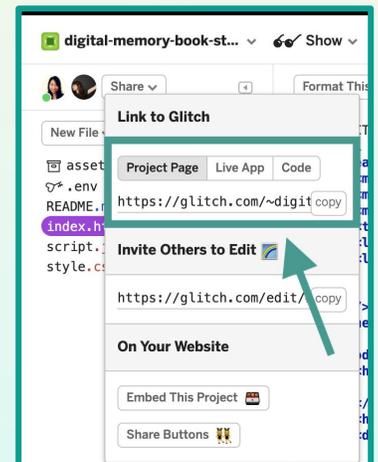


## Step 11: Share Your Girls Who Code at Home Project! (5 mins)

We would love to see your virtual hike projects! Don't forget to tag @girlswhocode #codefromhome and we might even feature you on our account!

### To Share your Glitch Project:

1. Click the **Share** button on the top left next to your profile icon.
2. You may choose to share your Project Page link (which will allow others to see your code, or just the Live App. Click on whichever option you would like to share and copy the link provided.



# Digital Memory Book Project Planning Worksheet

## Project Overview Planning

**Theme:** What do you want your memory book to be about? *This can be a memory book about a club, friend group, a creative project, or a family trip.*

**Image Content:** What kind of images do you want to include? *Pictures can be of people, scenery, art, etc.*

**Text Content:** What kind of text will you write for each photo? *Do you want to include quotes for each photo, descriptive text, names of the people included, hashtags?*

## Image Planning

Choose at least **three** images that you want featured on your digital memory book. Fill out the table below and include the image, the image URL address, and text.

<b>Image</b>	<b>Image Source</b> <i>(indicate if this will be your own photo or a photo from another website. If using a photo from a website document a link to the image. To get an image address watch this <a href="#">video</a>.)</i>	<b>Title</b>	<b>Description</b> <i>(what text do you want to display with this picture?)</i>

## Image Planning (Continued)

<b>Image</b>	<b>Image Source</b> <i>(indicate if this will be your own photo or a photo from another website. If using a photo from a website document a link to the image. To get an image address watch this <a href="#">video</a>.)</i>	<b>Title</b>	<b>Description</b> <i>(what text do you want to display with this picture?)</i>