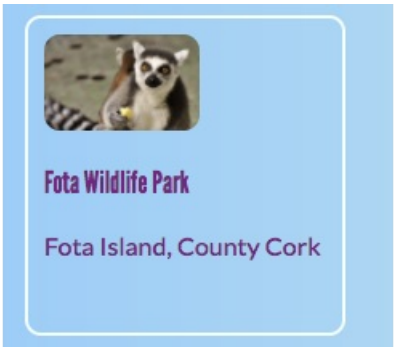
Table of Contents

1 This Sushi Series will show you how to make a website look polished and professional with cool effects and layouts. You'll learn tricks to make any layout you want, see how to make it adjust itself automatically to different screen sizes, make a collage of overlapping pictures and text, and apply some neat hover and click effects that you might have seen on other websites.

2 To work through the Sushi Cards you will need to start with a website that you already made. For example you can use the one that you made in the Intermediate HTML and CSS series. If you prefer, you can work with my example by either **Downloading** or **Remixing** this trinket: dojo.soy/html3-website-start

3 For a website with lots of code, like this one, you may find it easier to work with an offline text editor on your computer. If you prefer to work online and access your code from any device, go to dojo.soy/html3-trinket and create an account or sign in. It is free to create an account on Trinket; all you need is an email address.

1 Here's a technique you could use to make a photo gallery, or a portfolio page showing off all of your projects: little **preview cards**.



2 Add the following HTML code to your website, anywhere you like. I'm doing mine on index.html. You can change the picture and text to suit your own preview cards. I'm going to do a bunch of highlights of the tourist attractions in Ireland.

```
<article class="card">
    <img src="monkey-2223271_640.jpg" class="smallPics">
    <h4>Fota Wildlife Park</h4>
    <p>Fota Island, County Cork</p>
</article>
```

3 Add the following CSS code for the classes *card* and *smallPics* as well as for the heading *h3*:

```
.smallPics {
    height: 60px;
    border-radius: 10px;
}
.card {
    width: 200px;
    height: 200px;
    border: 2px solid #F0FFFF;
    border-radius: 10px;
    box-sizing: border-box;
    margin-top: 10px;
    font-family: "Lato", sans-serif;
}
.card:hover {
    border-color: #1E90FF;
}
```

4 The **Lato** font-family is used a lot in CoderDojo materials. You can also use **League Gothic** for a "CoderDojo-y" heading:

```
h3 {
    font-family: "League Gothic", sans-serif;
    font-style: normal;
    font-weight: 400;
}
```

5 Let's turn the whole thing into a link so people can click to see more information. Place the whole **article** element inside a link element. Make sure the closing `</a>` tag is after the closing `</article>` tag! Feel free to change the link **URL** to whatever you want to link to. That could be another page on your website or it could be another website entirely.

```
<a href="attractions.html#scFota">
    <article class="card ">
        <img src="monkey-2223271_640.jpg" class="smallPics">
        <h3>Fota Wildlife Park</h3>
        <p>Fota Island, County Cork</p>
    </article>
</a>
```

Notice how the value of **href** in my link ends in *#scFota*? This is a neat trick you can use to jump to a particular part of a page. First you type the URL of the page to link to, followed by **#**. In the code file for the page you are linking to, find the part you want to jump to and give that element an **id**, for example, `<section id="scFota"` . The value of the **id** is what you type after the **#** in your link.

6 Now that the whole thing is a link, the text font may have changed. You can fix it by adding a **CSS class** to the link, `class="cardLink"` . Here's the CSS code to put in your stylesheet:

```
.cardLink {
    color: inherit;
    text-decoration: none;
}
```

Setting the value of any property to *inherit* makes it use the value that the **parent** element has, so in this case the text colour will match the rest of the text on the homepage.

7 Make at least four or five of these cards. If you are working from my example website you could do one for each of the sections on the attractions page. On the next Sushi Card you'll arrange the cards with a cool trick!

1 On this card you will learn some tricks for arranging things **horizontally** on a page.

2 First, getting stuff centered! By setting the left and right margins to *auto* you can make any element be in the middle instead of over to the left. Try it now on the *.card* class.

```
margin-left: auto;
margin-right: auto;
```

3 That's one common problem solved! Another is arranging elements side by side in a row. Put all of the card links you just made into a new container element. It's not going to be an **article** or a **section**, but one called **div**. It's a general purpose container you can use for grouping things and making nice layouts.

```
<div class="cardContainer">
```

4 Add the following CSS in your stylesheet:

```
.cardContainer {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
    padding: 10px;
}
```

Voilà! Thanks to **Flex** Your cards are now displayed side by side! Drag the vertical slider on Trinket's code pane and watch how the cards move around to fit the window size.

5 Try deleting the **width** and **height** properties from the *.card* class and see what happens: **Flex** cleverly fits the cards together like a jigsaw puzzle, keeping an even height across everything that's in the same row.

6 If you have a navigation menu at the top of your page, that's another place you might use this trick. Find the CSS rules for your menu. If you prefer, you can try it out with my website. Delete `display: inline;` from the **list items** (in my website, that's the *nav ul li* block.). Then, in the list, *nav ul*, add in

```
display: flex;
justify-content: flex-start;
```

You end up with pretty much the same menu, right? The cool thing about Flex is you can control the layout with the property **justify-content**. Change its value to *flex-end* and see what happens. Or change it to *space-around* to make the menu items evenly spaced, just like you did for the cards.

7 A **responsive** website is one that adjusts itself to the screen size: so it always looks great whether you're looking at it on a computer, mobile phone or tablet. Let's make your menu responsive. Start with the regular styles: this will be your **default** behaviour.

8 Add the following CSS rules to your menu. You probably have colours and borders defined as well; I've left them out to save space here! If you already have CSS rules defined for your menu, just add in or change the properties and values that you are missing.

```
nav ul {
    padding: 0.5em;
    display: flex;
    flex-direction: column;
}
nav ul li {
    text-align: center;
    list-style-type: none;
    margin-right: 0.5em;
    margin-left: 0.5em;
}
```

9 With the CSS above you're going **mobile first**: the default style is good for small screens. You define different styles for bigger screens like this:

```
@media all and (min-width: 600px) {
    nav ul {
        flex-direction: row;
        justify-content: space-around;
    }
}
```

Everything inside this block applies whenever the window is wider than *600 pixels*. Can you add another block for screens bigger than *800 pixels*, with *flex-end* instead of *space-around*?

- **Flex** is a pretty powerful layout tool that could fill a whole Sushi Series of it's own, but you can learn more at dojo.soy/html3-flex

10 Trinket's project window is quite small. To test out your website on a full size screen, download the project and open up the file *index.html* in a browser. Adjust the width of the window to see the menu change! You can put any CSS rules you like into these blocks, to define different styles for different screen sizes. It'll be especially useful when you do **CSS grid** layouts later!

1 If you want to add a **caption** to a picture, that is, some text that goes with it like a title or short description, then you could make use of two elements designed just for that purpose: **figure** and **figcaption**!

2 Find an **img** element where you have text above or below that goes with the picture. I'm working with the Tito picture on i*ndex.html*, but you can go with whatever is on your website.

```
<img id="imgTito" class="solidRoundBorders" src="tito.png" alt="Tito the dog" />
<p>
  Tour guide Tito!
</p>
```

3 On the line above the code, add the tag `<figure>`. Place the closing tag `<\figure>` on a new line after the code.

4 Next, remove the **p** tags, or whatever tags you have around the text (maybe it's a heading, like **h2**?) and put the text in between `<figcaption> <\figcaption>` tags instead. The whole thing should look something like this:

```
<figure>
    <img id="imgTito" class="solidRoundBorders" src="tito.png" alt="Tito the dog" />
    <figcaption>
    Tour guide Tito!
    </figcaption>
</figure>
```

The **figcaption** element is your **caption**. It can go either above the **img** element or below it.

5 When you run your code, the picture and text might have changed position. Maybe you were happy with the original positioning of the elements and don't want this. Simply define CSS rules for the **figure** and set the margin properties to zero, or values that suit you. You can style both **figure** and **figcaption** as you would any other element, with background colour, borders, and everything else.

```
figure {
    margin-top: 0px;
    margin-bottom: 0px;
    margin-left: 0px;
    margin-right: 0px;
}
```

The **figure** element acts as a sort of **container** for your picture and its caption. As well as allowing you to treat them as one unit when defining styles, grouping them together logically helps to maintain a good structure to your website.

6 Another useful container is **aside**. You use it when you have extra stuff that doesn't really belong with the main information on a page. For example, the Attractions page on my website is a list of places to visit. I want to add some notes about weather and how to get around. That information doesn't really belong in the **article** element with all the attractions.

7 Outside of the **article** element, add one or more pairs of `<aside> <\aside>` tags containing your extra stuff.

```
<aside class="lightPurpleBackground">
    <h2>Getting around</h2>
    <h3>Train and bus</h3>
    <p>You can get to most of the major towns by train from Dublin. There are many buses that do tours to popular locat
ions and tourist attractions.</p>
    <h3>Car</h3>
    <p>The easiest way to get around outside of the cities is by car.</p>
  </aside>
  <aside class="lightPurpleBackground">
    <h2>Weather</h2>
    <p>The weather in Ireland is <span class="specialText">very unpredictable!</span> It's best to <span class="specialText">be prep
ared</span> for any kind of weather, even if it's a nice day!</p>
</aside>
```

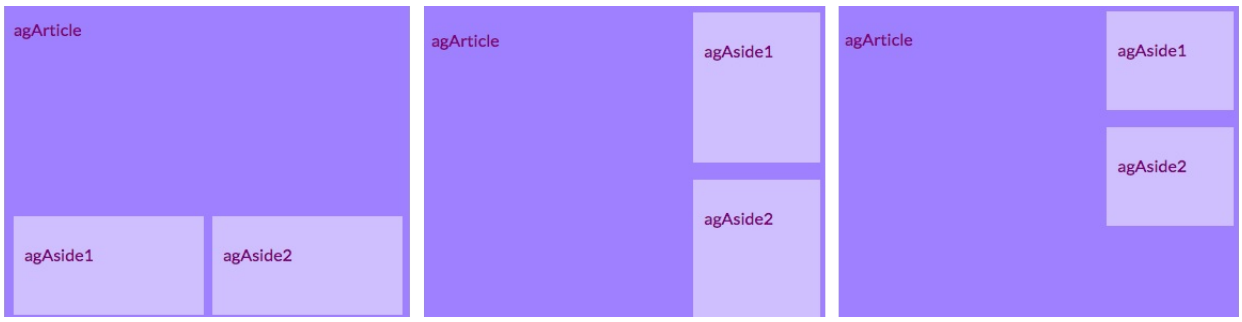
- The **aside**, **article** and other containers are similar. The only real difference is in the *meaning*, that is, what you use them for. It's important to use meaningful HTML elements whenever you can. It gives your website better structure and is especially helpful for people using **screen readers**.

8 You can write some CSS rules to make the **aside** elements look different or stand out if you want to. Did you spot the bonus element in there, **span**? It's a special tag you can use just for adding extra CSS! You can put anything in between a pair of **span** tags. It's useful for things like styling a *part* of the text in a paragraph.

```
.lightPurpleBackground {
  background-color: #CFBFFF;
}
.specialText {
    color: #FF4500;
    font-size: larger;
}
```

9 On the next card you're going to learn how to make the layout more interesting! To get ready, make a page that has one **article** and two **aside** elements inside the `<main> </main>` tags. Or if you prefer, you can work with the Attractions page on my website.

1 For this card you should work with a page that contains a **main** element with three elements inside, one **article** and two **aside**. Go ahead and create these first if you need to. If you want to work with my website, add the **aside** code from the previous Sushi Card onto the Attractions page. Here are three different page layouts you'll be applying:



2 Add new CSS classes to **main** and each of three elements inside it.

```
<main class="attPageLayoutGrid">
    <article class="attGridArticle">
        <!--other stuff here-->
    </article>
    <aside class="attGridAside1">
        <!--other stuff here-->
    </aside>
    <aside class="attGridAside1">
        <!--other stuff here-->
    </aside>
</main>
```

The container you'll change the layout of is **main**, but you could do this with any kind of container, like a **div** or **article**, or even the whole page **body**. The technique you're going to use is called **CSS grid**.

- In this example the **header** and **footer** will be left out of the design, but it's quite common to include them in the grid too.

3 Set the **display** property to *grid* on the overall container:

```
.attPageLayoutGrid {
    display: grid;
    grid-column-gap: 0.5em;
    grid-row-gap: 1em;
}
```

What do you think the **grid-column-gap** and **grid-row-gap** properties do?

4 Next, you name a **grid-area** for each element:

```
.attGridArticle {
    grid-area: agArticle;
}
.attGridAside1 {
    grid-area: agAside1;
}
.attGridAside2 {
    grid-area: agAside2;
}
```

5 Then you design your layout! Let's put the two **aside** elements side by side at the bottom. For this you need two **columns** of equal width. You can keep the **row** height automatic. Put the following code inside the *.attPageLayoutGrid* CSS rules:

```
grid-template-rows: auto;
grid-template-columns: 1fr 1fr;
grid-template-areas:
    "agArticle agArticle"
    "agAside1 agAside2";
```

fr stands for *fraction*. Notice how you make the **article** take up all the space over the two columns.

6 Let's try putting the **aside** elements over on the right, and making them half the width of the **article**. Change the values of **grid-template-columns** and **grid-template-areas** to:

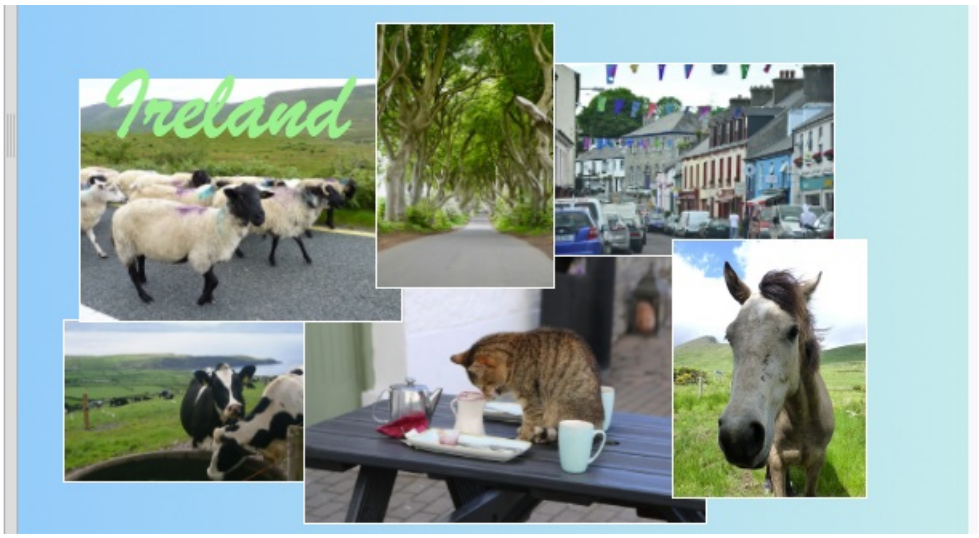
```
grid-template-columns: 2fr 1fr;
grid-template-areas:
    "agArticle agAside1"
    "agArticle agAside2";
```

If you don't want the **aside** elements to stretch all the way to the bottom, you can add a blank space using a dot:

```
grid-template-areas:
    "agArticle agAside1"
    "agArticle agAside2"
    "agArticle . ";
```

- With **CSS grid** you can make almost any layout you like. If you want to learn more, go to dojo.soy/html3-css-grid

1 On this card you will use CSS to position HTML elements exactly and make a photo collage.



2 Add a **div** to your page and put as many images in it as you like. Give the **div** and the **img** elements **id** values.

```
<div id="photoBox">
    <img id="imgHorse" src="connemara-pony-512028_640.jpg" alt="Connemara pony" />
    <img id="imgTeaCat" src="ireland-2360846_640.jpg" alt="Even cats drink tea in Ireland!" />
</div>
```

The photos will appear one after the other on the web page, in the order they appear in your code.

3 In your CSS file, add separate style rules for each of the elements using **id selectors**. For each of the elements inside the container, add the property `position: absolute;`. This lets you choose exact positions for them. You will want to define those positions *relative to* (i.e. *within*) the container, so you also need to add the property `position: relative;` to the container itself.

```
#photoBox {
    width: 800px;
    height: 400px;
    position: relative;
}
#imgHorse {
    width: 120px;
    position: absolute;
}
#imgTeaCat {
    width: 250px;
    position: absolute;
}
```

4 Then, you choose the exact positions you want for each picture. There are four properties you can use: **left**, **right**, **top**, and **bottom**. They represent how far each of the edges should be from the parent's edge. Use either **top** or **bottom** for the vertical position and use either **left** or **right** for the horizontal position.

This code places the cat picture *100 pixels* from the top and *60 pixels* in from the left.

```
#imgTeaCat {
    width: 250px;
    top: 100px;
    left: 60px;
    position: absolute;
}
```

- The position values can also be negative!

5 You can also decide how the pictures overlap, using the **z-index** property. The value can be any whole number. The picture with the *highest* number ends up on *top* of the pile!

```
#imgHorse {
    width: 120px;
    top: 20px;
    left: 10px;
    position: absolute;
    z-index: 10;
}
#imgTeaCat {
    width: 250px;
    top: 100px;
    left: 60px;
    position: absolute;
    z-index: 7;
}
```

6 You can position any html elements in this way, not just images. Try creating your own collage of photos, perhaps with some text over the top! Use exact positioning together with different **z-index** values to get the overlap effect the way you want it.

1 Let's add a little movement when you hover over the cards you made earlier! You may remember the **transform** property from the Intermediate Sushi Cards (when you made things rotate with **@keyframes** animations!). You can also use it to move something up or down with *translateY* and left or right with *translateX*. Find the *.card:hover* CSS class from earlier and change it to the following. Try out different values in the traslate function!

```
.card:hover {
    box-shadow: 0px 2px 2px rgba(0,0,0,0.2);
    transform: translateY(-2px);
}
```

Play about with different pixel values in the `box-shadow` property to see what they do.

2 *rgba(0,0,0,0.2)* is another way of defining a colour. It's got the usual three numbers (from 0 up to 255) for Red, Green and Blue. The fourth number, called the **alpha** value, sets how see-through something is; it is a number between **0** and **1**.

3 Finally, make the movement smooth by adding the following property to the *.card* class from earlier:

```
transition: all 0.2s ease-out;
```

A duration of *0.2s* means the transition lasts for *0.2 seconds*.

4 Another effect you've probably seen on loads of websites is **lightbox**, where you click on something and the screen dims while something else, like a bigger picture or a popup box, appears in front of everything. To get this effect you will make two links.

5 The first link is for the actual **lightbox**. It contains all the stuff that will appear when you click. Make sure you give the link itself an **id**. I'm doing mine on the Attractions page of my website. You go with whatever page you have pictures on!

```
<a href="#_" class="lightbox" id="boxLemur">
    <h3>Lemur!!</h3>
    <img src="monkey-2223271_640.jpg" alt="Picture of a lemur" class="bigPics"/>
    <p>A lemur enjoying a little snack</p>
 </a>
```

You can put anything you like in between the link tags. I've got a big picture, a heading and some text. Maybe you just want a picture and no text.

- It doesn't matter where inside the **main** element you put this code, as you will be making it invisible soon!

6 The other link is of course the thing that you click to make the lightbox appear. Simply add a pair of **a** tags around the element, in this case a smaller picture of a lemur. The **target** of the link will be the lightbox, which you set using the **id**. You might recognise this technique from earlier!

```
<a href="#boxLemur">
    <img src="monkey-2223271_640.jpg" class="smallPics">
</a>
```

7 Here's the CSS for the lightbox. Can you work out what most of it does?

```
.lightbox{
    background: rgba(0,0,0,0.8);
    color: #ffffff;
    text-align: center;
    text-decoration: none;
    width: 100%;
    height: 100%;
    top: 0;
    left: 0;
    position: fixed;
    visibility: hidden;
    z-index: 999;
}
.lightbox:target {
    visibility: visible;
}
```

Setting the **position** property to *fixed* means it stays put when you scroll. The `:target` **pseudo-class** only applies when the lightbox was the target of the last link clicked. So when you click anywhere the **visibility** will be set back to hidden.

8 You can add as many lightboxes as you want to a page. They can all use the same CSS class. Just make sure each one has a different **id**! For each one, you need to make something on your webpage into a link that you can click to make the lightbox appear, and you use the **id** as the **href** value in that link; just you've done above!