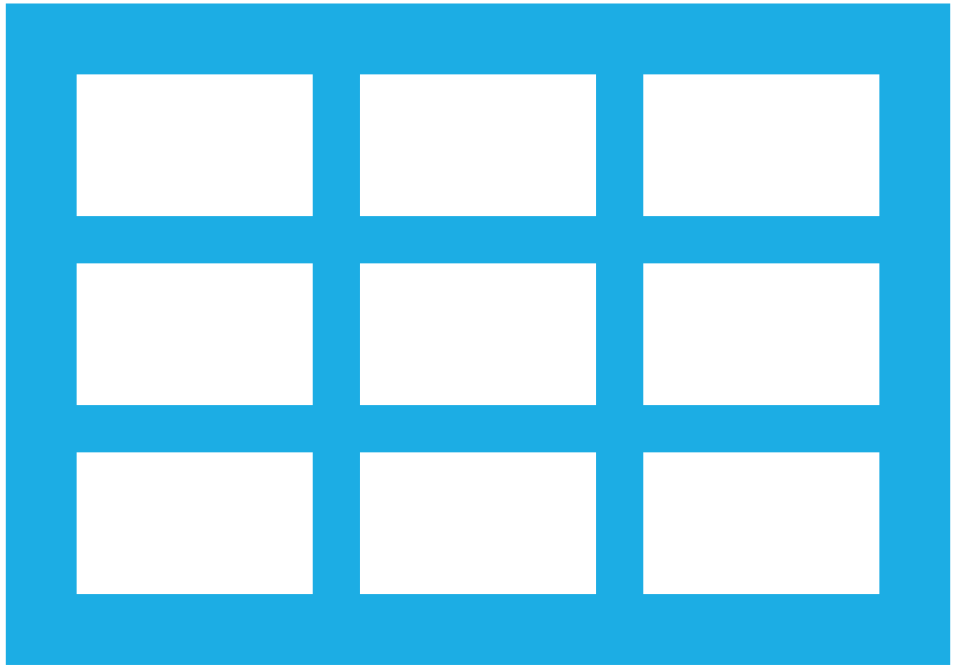
CSS LAYOUT AND GRID



INTRODUCTION

So far we have looked basic ways to change our page. We have changed color, sizes, fonts, shadows and many others.

We haven't talked about one of the most important aspects of CSS which is controlling the layout of your pages! Technically we could do this using complex combinations of position and display, but we need a dedicated CSS property that will allow us to really take control of our pages.

LAYOUT

Website layout is the boxed structure that makes up the flow of your page.

It is now necessary to have different layouts for different screen sizes so users have a better experience on your website. Layout is one of the main influences on your users. Is it laid out in a way that makes sense or does it feel more random? Are things too close together, too far apart, or does it feel appropriate?

It is worth noting that before flexbox and grid, CSS did not have any real dedicated layout tooling.

GRIDS

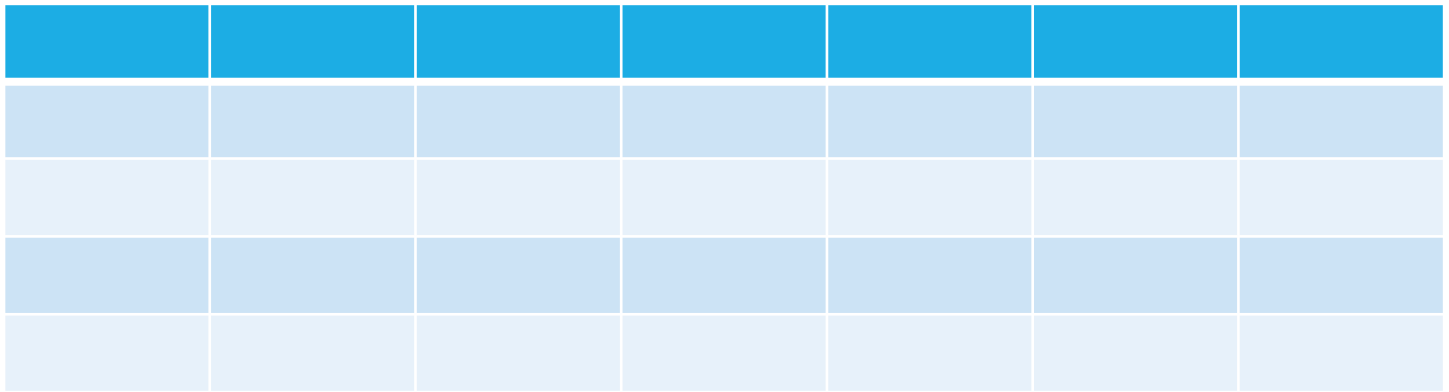
One of the most logical ways to try and layout a website is by imagining it as a grid/table on your user's screen. This means you break up your users screen into a system of rows and columns that separate sections of your site.

Grids are a very handy way to imagine our site because if they are flexible enough, we can create some really innovative layouts.

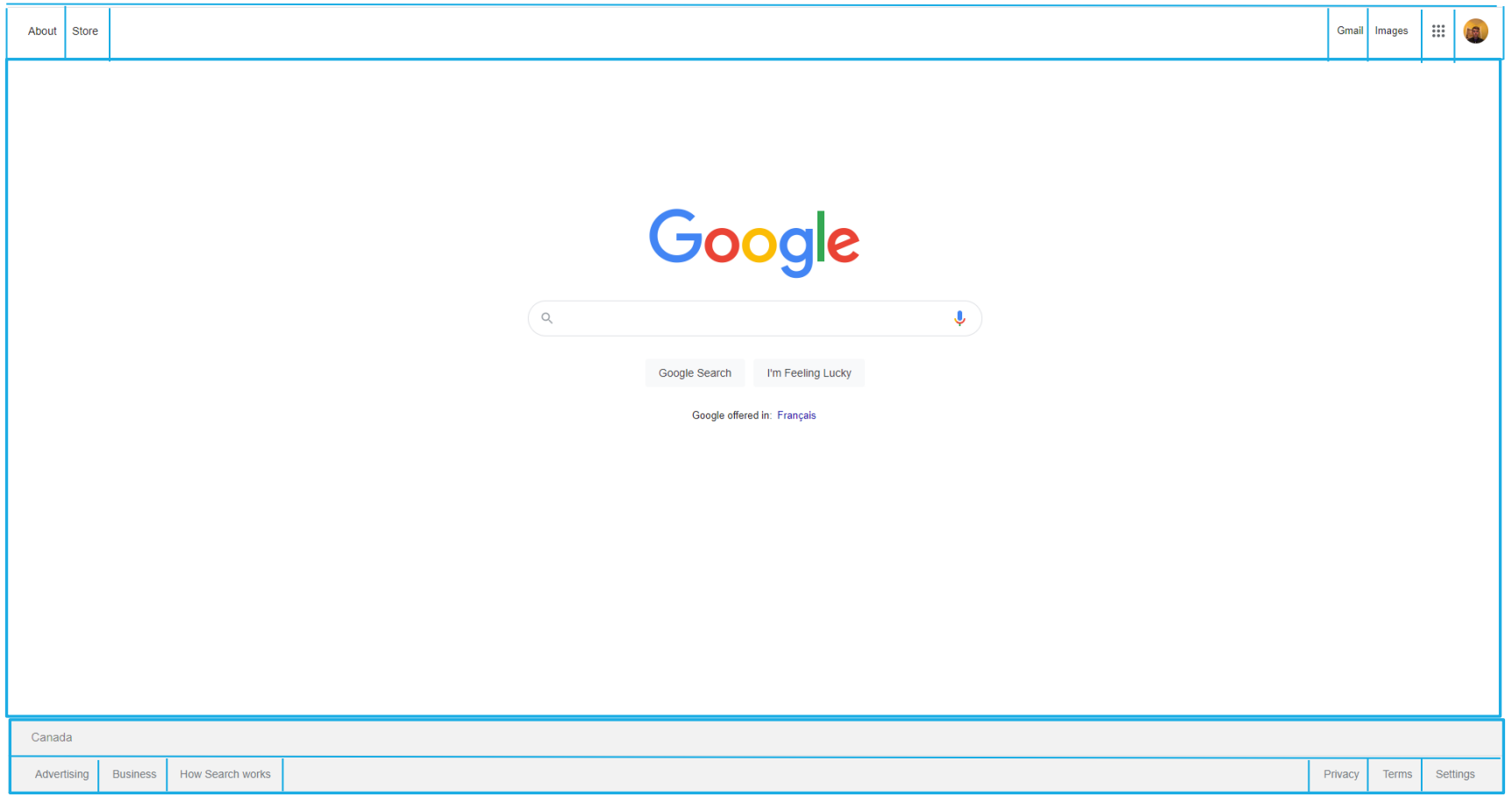
It should be noted that although grids are great, they are by definition boxy.

GRIDS CONT.

The simplest grid to imagine is a simple table:



SEEING GRIDS EVERYWHERE



CSS GRID

Let's start with some terms:

- The **grid container** is the parent element in your HTML that the CSS property "display: grid" is applied to
- Children of the grid container are called **grid items** and are the direct children of the parent HTML item with display grid
- The **grid lines** make up the space in between our grid items. Here we can specify column line or row lines (horizontal vs vertical gap distance)

CSS GRID CONT.

When working with CSS grid it is helpful to remember that **most of the styling is done in the parent element** or “grid container”.

The grid container will choose things like how many rows and columns the grid should be made out of and even how large those should be. The grid container can also tell the children how the should align themselves in their grid cell.

One thing we need to keep in mind is that in CSS grid, all children of the grid container will be placed in **rows** by default. Even if they are inline tags the grid will place them onto their own lines!

CREATING A GRID

To turn an element into a grid you just have to add one line of CSS:

```
#gridcontainer {
    display: grid;
}
```

Just like that, our containing element (usually something like div, article, section, nav, header, etc.) will change itself into a simple grid based layout!

THE GRID CONTAINER

The grid container is where most of the action is going to take place. Here we will define the general layout of the grid based on the rows and columns we want.

Remember that by default the grid container will change all children into rows by default. If we want to change this behaviour we can use the property **grid-auto-flow** and change it between **row** or **column**

```
#gridcontainer {
    display: grid;
    grid-auto-flow: column
}
```

THE GRID CONTAINER CONT.

We can take direct control of our grid by telling it how many rows and columns it should create. We can do this using the **grid-template-rows** and **grid-template-columns** properties. This is going to look new to us:

```
#gridcontainer {
    display: grid;
    grid-template-columns: 200px 200px 200px;
}
```

We have created a grid that default all new children into rows, but it also has 3 columns defined that are each 200px wide! This means that each new child will go into the highest row and furthest left empty column by default. Note that you can use any measurement unit you would like here, doesn't have to be absolute!

KNOWLEDGE CHECK

Let's try and make a really basic grid:

1. Create a folder called CssGrid in your Scratch folder
2. Connect this folder to Git/GitHub
3. Add a basic index.html and style.css
4. Add in the following:
    1. One section tag
    2. Three articles inside the section
    3. Give each article one image, one heading and one paragraph
5. Turn the section tag into a grid container and ensure it has 3 equally sized columns
6. Add commit and push your code

THE GRID CONTAINER CONT.

We are also introduced the **fr** unit of measurement with grid. It stands for “fraction” and is a relative measurement.

The **fr** unit will add up all other **fr** units into a total, and then divide themselves up amongst the other **fr** tags. It will also just work around any rows/columns that have an absolute unit of width/height.

This is best understood with examples.

THE GRID CONTAINER CONT.

We can also interact with our grids line spacing from the parent. This will create some white space between our elements.

We use the “column-gap” and the “row-gap” to define these spaces:

```
#gridcontainer {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    column-gap: 20px;
}
```

THE GRID CONTAINER CONT.

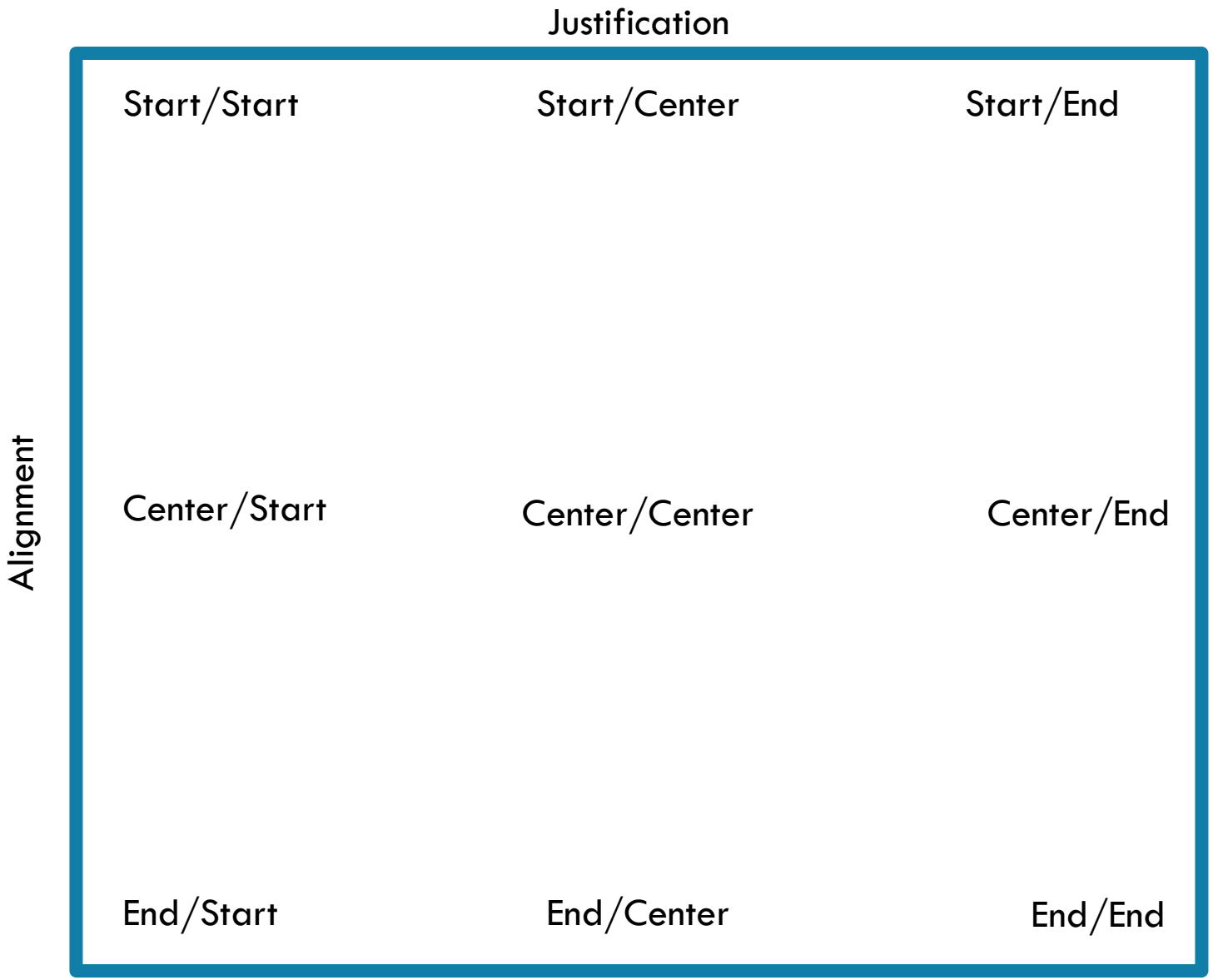
Your parent container can even tell its children how to align their content within a cell (although individual cells can override their parents if they want).

We use the “justify-items” and the “align-items” property to position the content of each child cell.

The justify-items property is responsible for horizontal (or row) positioning while align-items is responsible for vertical (or column) positioning.

We can also use the short hand “place-items” property to specify both justify-items and align-items.

THE GRID CONTAINER CONT.



THE GRID CONTAINER CONT.

These properties will tell all grid children to align their content right into the middle of their grid cells!

```
#gridcontainer {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    column-gap: 20px;
    justify-items: center;
    align-items: center;
}
```

THE GRID CONTAINER CHILD

We aren't going to dive into too much details about the child cells of a grid in these slides, but for now we will simply talk about overriding the grid containers alignment or justification properties:

```
#gridcontainer {
    display: grid;
    grid-template-columns: 1fr 2fr 1fr;
    column-gap: 20px;
    justify-items: center;
    align-items: center;
}

#specificcell {
    align-self: start;
}
```

KNOWLEDGE CHECK

Let's try and make a really basic grid:

1. Add in the following to your HTML:
    1. One header tag at the top of the page
    2. One image inside the header
    3. One nav tag inside the header
    4. 3 a tags inside the nav
2. Turn the header tag into a grid container and set it to default flow columns. Give it a height of 50px
3. Justify the image to the left side and the nav to the right side
4. Align all content to the center
5. Add, commit and push your code

KEEPING IT SIMPLE

When starting out with grid, many students over-complicate the entire process. You will find when you search things up about grid many people have different ways to implement the exact same solutions.

The tools we have gone over here in these introduction slides are more than enough and give you the ability to create some very fancy layouts.

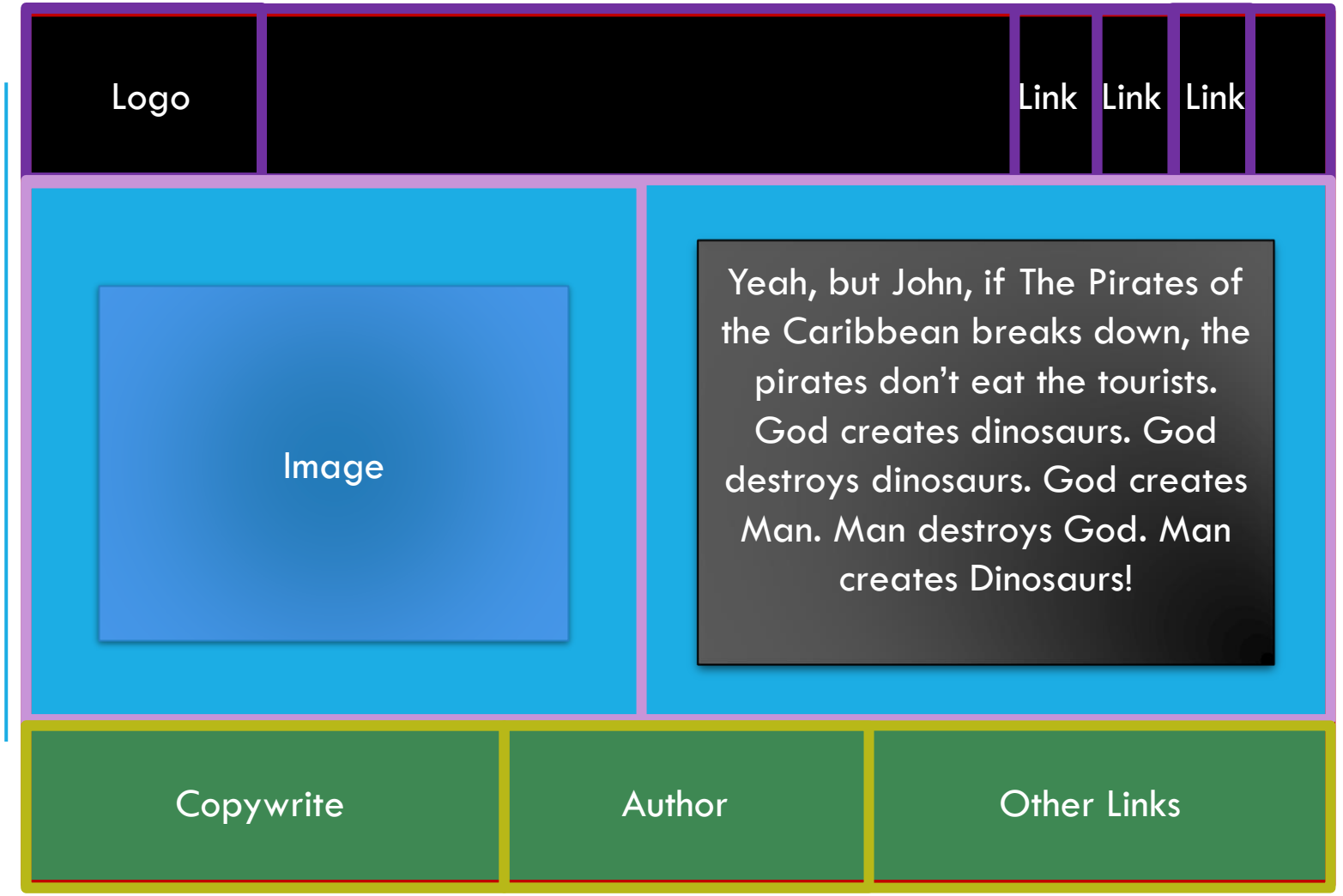
Grid is also filled with CSS shortcuts! Again if you are doing some research and you see something that looks odd, it is probably just a shortcut for what we have talked about here.

NESTING GRIDS

This is another topic that tends to confuse people at the start but it is really simple. You can turn any grid cell into it's own grid container (which means you're probably using some kind of semantic or layout tag for the child).

This has **no** effect on the parent grid! This simply turns the grid cell into it's own grid container. You will actually find that nesting grids is pretty much needed to make complex layouts. Let's take a look at this by dissecting a pretty standard layout.

NESTING GRIDS CONT.



GRIDS

Now we have the power to create some truly awesome layouts! Next we will cover how to use some of the more advanced grid features that allow for automatic responsiveness!

I do recommend that from now on when making designs on Figma and such to really think about how your layout translates into a grid. I also suggest trying to think about grid for more uses than just the overall layout of the site! Turning things like articles into grids can also make their formatting easier.