Interesting layouts

Every so often I look at designs and try to duplicate them. This time I'm starting with a video from Layout Land to start setting expectations and to push myself out of my comfort zone as a designer and developer.

Most of these layouts will be based on CSS Grids so they may not work in older versions of browsers. I still will try not to use JavaScript unless I find a compelling reason.

The layouts will be published in my web layout experiments site. I will also post work in progress in Codepen.

Placing contents on the grid: Calendar of events

One good starting point is placing content in different cells of the grid; How to place items in different positions on the grid with little or no extra work.

Creating the calendar and placing single-day events is fairly straightforward. The calendar layout is in this Codepen demo

I'm doing something here that I don't always do. I'm letting the grid algorythm place the items in the grid. If we want to make sure that the days match the calendar we can tell the browser what column to start with using something similar to this:

```
.day:nth-child(1) {
  // assuming the first day of the month
  // is a sartuday
  grid-column: 6;
}
```

The problem comes when trying to place events on more than one day like a 3-day conference or a week long vacation...

Looking at Jonathan's Snook's implementation of a grid calendar as a demo of

what a calendar should look like

In his post, Snook mentions James Finley who used CSS Custom Properties in his version.

If we were not concerned about placement we could create the grid so that each week had two rows, like so:

```
.week {
  display: grid;
  grid-template-columns: repeat(7, 1fr);
  grid-template-rows: 50px, 200px;
}
```

But this would make it harder to use `display: contents` with the events. Instead, we create them as a 7 equal-width columns and one single row.

I've also chosen not to use grid-auto-flow and let them flow on the grid without consideration as to how will they line. My concern here is that if we have events packed across days we may loose sight that they are different events.

```
.week {
  display: grid;
  // comment the following line if you want to test
  // the difference of using
  // grid-column-autoflow: dense works or not
  grid-column-autoflow: dense;
  grid-template-columns: repeat(7, 1fr);
}
```

We can add as many weeks as we need to use to get our data across.

Inside each week we have 7 days, represented by div elements with the class day.

The day element has 2, or more children:

- `day-label` tells us what day it is
- `event` tells us what event is. We can have as many events as we need to

display our data.

Inside event there are two elements worth noting: `--start` and `--span` represent the two CSS custom properties that define what column the event starts in (meaning what day) and how many columns it will span (meaning the event's duration).

Be extra careful when setting up events. If they go beyond day 7 the browser will add columns to the existing row and the results will not be pretty (and most definitely not what you wanted).

```
<div class="day">
  <h3 class="day-label">1</h3>
  <div class="event event-start event-end" style="--start: 2; --span: 2">
    Class
  </div>
  <div class="event event-start event-end">Interview</div>
</div>
```

Background Reveal

This is different than parallax. The idea is that the background itself reveal itself as we scroll the page up or down. Again, the HTML is easy, broken down by sections:

```
<section>
  <h1>Welcome to Chile</h1>
</section>
<section>
  <p>Chile has a lot of different areas that may be of interests</p>
</section>
<section>
  <h1>Easter Island</h1>
</section>
```

The css is broekn in three blocks. The first block is common to all `section` elements whether we will use background images in them or not.

This will give us flexibility to use background images in whatever section we

decide.

```
section {
  height: 100vh;
  box-sizing: border-box;
  display: flex;
  align-items: center;
  justify-content: center;
  text-align: center;
  flex-direction: column;
  padding: 2vw;
  background-size: cover;
  background-repeat: no-repeat;
  background-attachment: fixed;
  font-size: 6vw;
}
```

The second block assigns values to odd and even sections. All even sections will have the same charactersitics, so will all odd sections that have a common set of characteristics all their own.

```
section:nth-of-type(odd) {
  color: #000;
  font-size: 4vw;
}

section:nth-of-type(even) {
  background-position: 50% 50%;
  color: rgb(255, 255, 255);
  font-size: 7vw;
  text-shadow: 10px 8px 5px rgba(0, 0, 0, 0.8);
}
```

The final block takes care of individual blocks. For example, each even block has its own backgroound image and we could add more item-specific attributes where needed.

```
section:first-of-type {
  background-image: url(../images/chile-01.jpg);
}

section:nth-of-type(3) {
  background-image: url(../images/chile-02.jpg);
}

section:nth-of-type(5) {
  background-image: url(../images/chile-04.jpg);
}
```

Again, this is different than Parallax effects as the content is not moving but the background appears to be.

Playing with Grid positioning

The video below, from Jen Simmons' Layout Land, shows an idea of what I want to do. The site is here.

The goal with this project is to create an overview page for a website with live links and some interesting layout possibilities:

Grid and overlays

https://farro-demo.squarespace.com/

Flat Designs

https://york-demo.squarespace.com/

Parallax?

https://dixonandmoe.com/rellax/

Overlaping items on the grid

https://gedd.ski/post/overlapping-grid-items/

Art Directed Layouts

https://v6.robweychert.com/blog/2018/11/css-grid-editorial-layouts/

How to use media queries and grids together

https://thoughtbot.com/blog/concise-media-queries-with-css-grid

Looking forward: Exclusions

Exclusions will hopefully be like more powerful grid and multicolumn friendly floats