Comp 324/424 - Client-side Web Design

Fall Semester 2024 - Week 10

Dr Nick Hayward

CSS3 Grid - sample layouts

intro

- grid layout enables more complex and interesting layout options
  - overlap, layers…
- sample layouts using CSS grid structure
  - common layout options and designs
  - useful repetition of design
  - modify base layouts for various site requirements
- sample layouts
  - responsive layouts
  - auto placement for dynamic content and media
  - platform agnostic designs
  - useful with SPA, SVG, async patterns &c.

HTML5, CSS, & JS - example - part 4

add flex to grid layout

- an additional option to consider - flex layouts
  - aims to provide efficient way to align and proportion content
- known as **Flexbox Layout**
  - idea is to apportion width and height for content
  - proportions relative to container even when their size is unknown or dynamic
- flex layout could, in theory, replace a full grid layout
  - considered more a complement to overall grid structure
- defined flex container expands items to fill the container's available space
  - can also shrink them to prevent any possible overflow
- think of a flex layout as supporting multiple directions
  - direction agnostic
- many properties available for **flex**
  - focus upon styling flex container and any flex items

CSS - Flexbox

intro

- helps solve many issues that have continued to plague layout and positioning
- used with HTML elements and components

  - both client-side and cross-platform apps
- a few issues it tries to solve
  - vertical and horizontal alignment
  - defining a centred position for child elements relative to their parent
  - equal spacing and proportions for child nodes regardless of available space
  - equal heights and widths for varied content
  - & lots more…

CSS - Flexbox

basic usage

- for any app layout, we need to define specific elements as *flexible boxes*
- i.e. those allowed to use flexbox in a given app
  - e.g.

```
section {
  display: flex;
}
```

- ruleset will define a `section` element as a parent flex container
  - child elements may now accept flex declarations
- initial declaration, `display: flex`
  - also includes default values for flexbox layout of child elements
- e.g. `<div>` elements in a section
  - by default now arranged as equal sized columns with the same initial height

---

CSS - Flexbox

axes

- elements arranged using flexbox are laid out on two axes
- main axis
  - axis running in the direction of the currently laid out flex items
  - e.g. rows or columns
  - start and end of axis = *main start* & *main end*
- cross axis
  - axis running perpendicular to the current main axis
  - start and end of axis = *cross start* & *cross end*
- each child element laid out inside flex container called a *flex item*

CSS - Flexbox

flex direction

- set a property for the flex direction
  - defines direction of flex items relative to main axis
  - i.e. layout direction for child elements
- default setting is `row`
  - direction will be relative to current browser language setting
  - e.g. for English language browsers = left to right

```
section {
  flex-direction: column;
}
```

- override the default `row` setting
  - arrange child items in a column

```
section {
  display: flex;
  flex-direction: column;
}
```

- ensures child flex items were 1aid out in a single column
- then override specific `section` elements
  - allow child flex items in a `row` direction

```
#tabs {
  flex-direction: row;
}
```

Image - CSS Flexbox

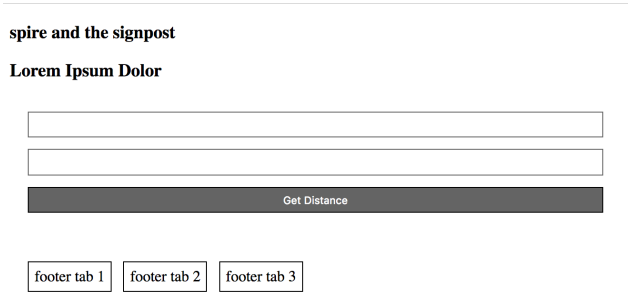

Figure 1: CSS Flexbox - flex direction

flex direction

CSS - Flexbox

flex item wrapping

- ensure child items do not overlap their parent flex container
  - add a declaration for `flex-wrap` to a required ruleset
  - e.g.

```
#tabs {
  flex-direction: row;
  flex-wrap: wrap;
}
```

Image - CSS Flexbox

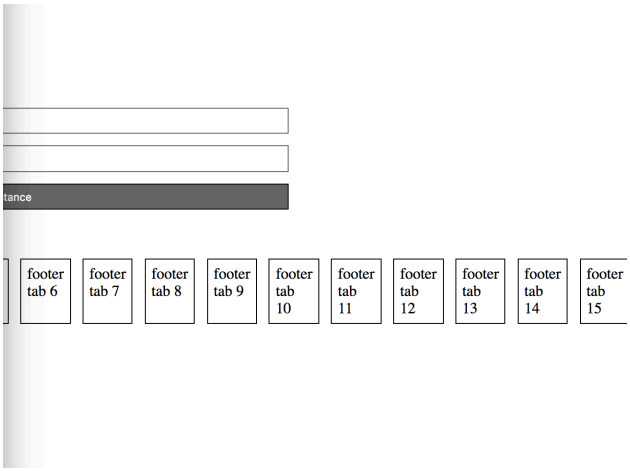

Figure 2: CSS Flexbox - no flex wrap

without wrap

Image - CSS Flexbox

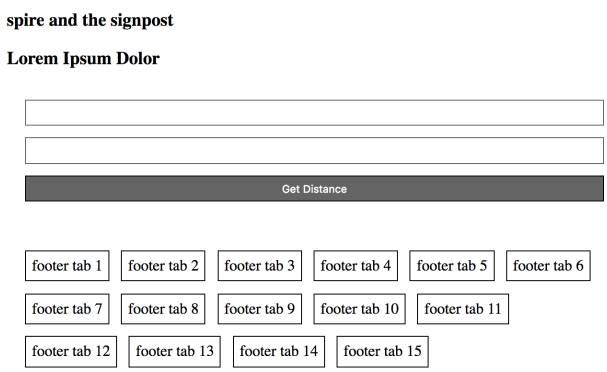

Figure 3: CSS Flexbox - flex wrap

with wrap

Video - Flexbox

flexible design Examples of Modular UI Design

Source - Modular UI Design - YouTube

HTML5, CSS, & JS - example - part 5

add flex to grid layout - option 1

- we might specify CSS properties for a flex container

```
.flex-container {
    display: flex; /* defines container as flex */
    flex-direction: row; /* defines positioning of items added to container */
    flex-wrap: wrap; /* defines whether to wrap items to another line */
    justify-content: flex-start; /* defines start point and distribution of items */
}
```

- allows us to position our container starting at the left
  - items contained in a row
  - contained items wrapping to additional lines if necessary
- many additional options available for each property
- also add rulesets for specific styling of items within a flex container
- we could add properties to a flex item such as
  - specify the order of the flex items
  - whether a particular item can grow or shrink relative to content
  - default size of an item before any remaining space is distributed
  - individual alignment for a given item...

CSS - Flexbox

flex direction reverse

- also set flex direction to reverse
  - starts flex items from the right on an English language browser

```
#tabs {
  flex-direction: row-reverse;
  flex-wrap: wrap;
}
```

Image - CSS Flexbox

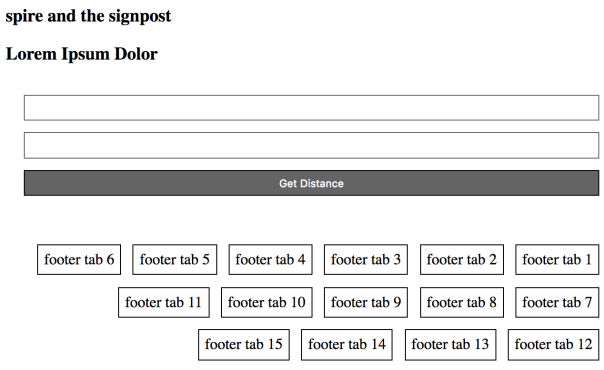

Figure 4: CSS Flexbox - flex direction reverse

flex direction reverse

CSS - Flexbox

`flex-flow` shorthand

- also combine *direction* and *wrap* into a single declaration
  - `flex-flow`
  - now contain values for both `row` and `wrap`
  - e.g.

```
#tabs {
  flex-flow: row wrap;
}
```

HTML5, CSS, & JS - example - part 6

add flex to grid layout - option 2

- flex container for option 2 design

```
/* note container - flex */
.note-output {
    display: flex;
    justify-content: space-between;
    flex-wrap: wrap;
    row-gap: 20px;/*applies to rows of items - not above first row... */
    padding-top: 20px;
}
```

- output notes section
  - organise single notes as flex items
  - add gap between rows of flex items
- justify content in container
  - notes start at left edge, end at right edge
  - space between evenly apportioned per note

CSS - Flexbox

sizing of flex items

- for each flex item, we may need to specify apportioned space in the layout
  - e.g. set space as an equal proportion for each flex item
  - we may add the following to a child item ruleset

```
div.fTab {
  flex: 1;
}
```

- defines each child flex item `<div class="fTab">`
  - occupy an equal amount of space within the given row
  - after considering margin and padding
- **n.b.** this value is proportional
  - doesn't matter if the value is 1 or 100 &c.
- define additional flex proportions for specific child items
  - e.g.

```
div.fTab:nth-child(odd) {
  flex: 2;
}
```

- each odd *flex-item* will now occupy twice available space
  - space in the current direction

Image - CSS Flexbox

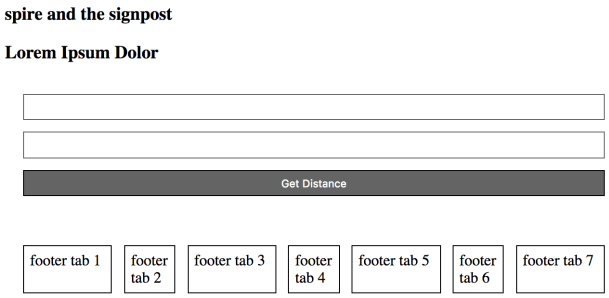

Figure 5: CSS Flexbox - flex item sizing

flex item sizing

CSS - Flexbox

minimum size

- then set a minimum size for a flex item
  - e.g.

```
div.fTab {
  flex: 1 100px;
}
```

- or a relative unit for the size

```
div.fTab {
  flex: 1 20%;
}
```

- each flex item will initially be given a minimum
  - e.g. `20%` of the available space
  - the remaining space will be defined relative to proportion units

---

Image - CSS Flexbox

flex item sizing

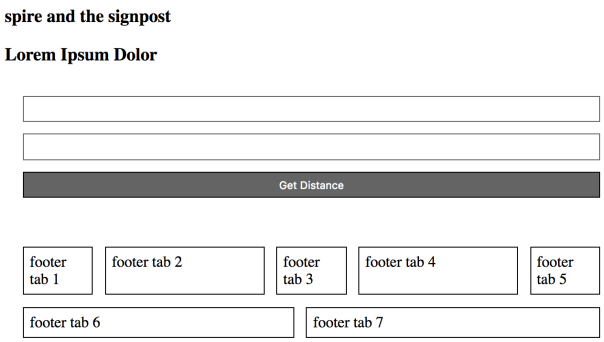

Figure 6: CSS Flexbox - flex item sizing - minimum size

HTML5, CSS, & JS - example - part 7

add flex to notes

- flex container and items useful for organising and positioning our notes
- due to uncertainty about content, size, and general note requirements
  - flex positioning and styling removes the need for assumptions or fixed sizes
- we can start to modify the styling and rendering of our notes using flex

```
/* flex item */
.flex-item {
  flex-basis: 300px; /* default size before extra */
  flex-grow: 1; /* all items will be equal */
}
```

- gives us a default smallest size for each note
- then the ability for each note to grow to fill the row as required
- also work with responsive layouts
  - due to the minimum size and the option to grow for each item
  - and wrap flex items per flex container
- modify and update styles as we develop travel notes app

DEMO - Travel Notes - grid layout with flex notes

Image - HTML5, CSS, & JS - Flex Notes

Image - HTML5, CSS, & JS - Flex Notes 2

Image - HTML5, CSS, & JS - Flex Notes 3

HTML5, CSS, & JS - example - part 8

add flex to notes

Figure 7: Grid Layout - flex notes

- Notes with Flex and Media Queries

HTML5, CSS, & JS - example - part 9

add flex to notes - option 2

- define styling for flex items in option 2 design
- note defined using card layout design
  - card-view, card-content

```
/* note card - flex */
.card-view {
  display: flex;
  flex-direction: column;
    flex: 0 0 250px;
    border: 1px solid #CCCCCC;
    padding: 20px;
}
.card-content {
  flex: 1;
}
```

- card is flex container for child flex items
  - e.g. note content, header, footer &c.
- `flex` defines shorthand property
  - flex-grow, flex-shrink, flex-basis
  - note set to initial length of `250px`

Figure 8: Grid Layout - flex notes - medium

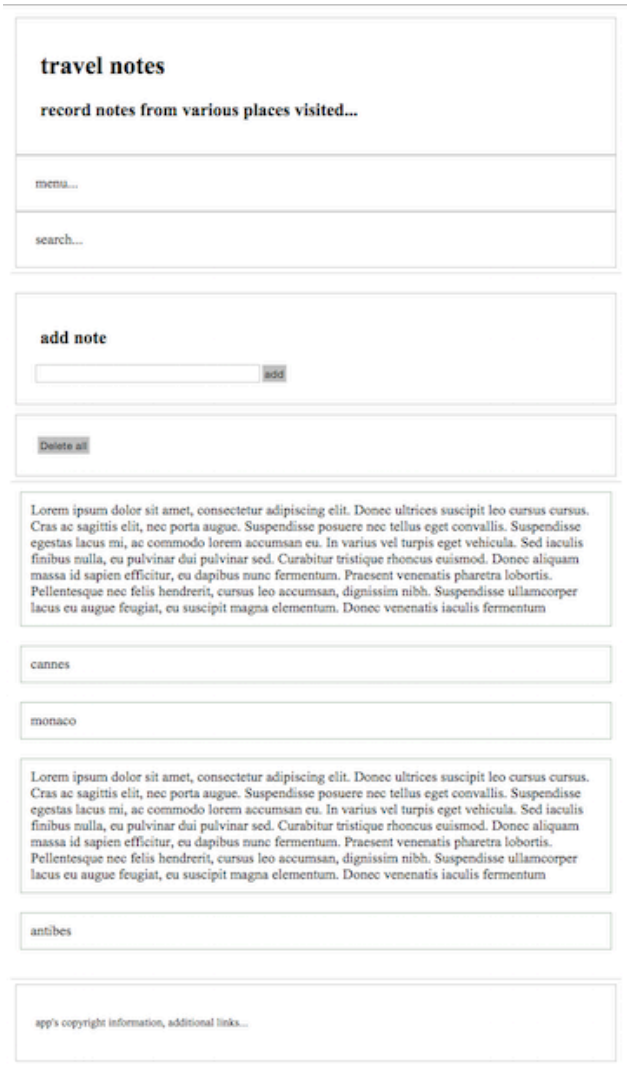

Figure 9: Grid Layout - flex notes - small

CSS - Flexbox

flex item alignment

- Flexbox allows us to define alignment for flex items in each flex container
  - relative to the main and cross axes
- e.g. we might want to specify a centred alignment for flex items

```
#tabs {
  flex-direction: row;
  flex-wrap: wrap;
  align-items: center;
}
```

- `align-items: center`
  - causes flex item in flex container to be centred along the cross axis
  - however, they'll still maintain their basic dimensions
- also modify value for `align-items` to either `flex-start` or `flex-end`
- such values will align flex items to either start or end of cross axis

---

CSS - Flexbox

override align per flex item

- as with `flex`
  - also override alignment per flex item
  - using `align-self` property add a value for positioning
- e.g. a sample declaration might be as follows

```
div.fTab:nth-child(even) {
  flex: 2;
  align-self: flex-end;
}
```

CSS - Flexbox

justify content for flex item

- also specify `justify-content` for flex items in a flex container
  - property allows us to define position of a flex item relative to main axis
- default value is `flex-start`
- then modify it relative to one of the following
  - `flex-end`
  - `center`
  - `space-around`
    - distributes each flex item evenly along main axis with space at either end
  - `space-between`
    - same as `space-around` without space at either end…

CSS - Flexbox

alignment and order - part 1

- define alignment relative to each axis using a specific declaration
  - e.g. for the main we may use `justify-content`
  - for the cross axis we use `align-items`
- also modify layout order of flex items
  - without directly changing underlying source order
- use the following pattern to specify order

```
div.fTab:first-child {
  order: 1;
}
```

- first flex item will now move to the end of the tab list

Image - CSS Flexbox

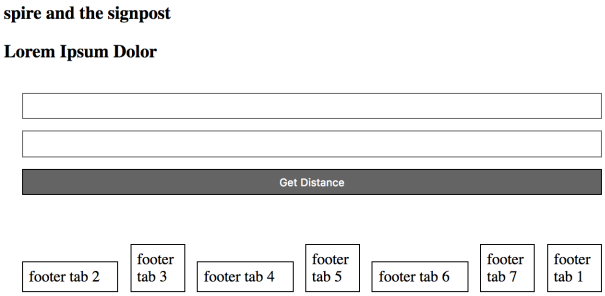

Figure 10: CSS Flexbox - flex item order 1

flex item order

CSS - Flexbox

alignment and order - part 2

- due to default order for flex items
  - by default, all flex items have an `order` value set to `0`
- higher the `order` value, later the item will appear in the list &c.
- items with the same order will revert to the order in the source code
- also possible to ensure certain items will always appear first
  - or at least before default `order` values
  - by using a negative value for the `order` declaration
  - e.g.

```
div.fTab:last-child {
  order: -1;
}
```

CSS - Flexbox

nesting flex containers and items - part 1

- Flexbox can also be used to create nested patterns and structures
  - e.g. we may set a flex item as a flex container for its child nodes
- we might add a banner to the top of a page

```
<section id="banner">
  <header id="page-header">
    <h3>spire and the signpost</h3>
    <h5>point to the stars...</h5>
  </header>
  <section id="search">
    <input type="text" id="searchBox"/>
    <button id="searchBtn">Search</button>
  </section>
</section>
```

CSS - Flexbox

nesting flex containers and items - part 2

- set `#banner` , `#page-header` , and `#search` as flex containers
  - e.g.

```
#search {
  display: flex;
}
```

- then specify various declarations for `#search`
  - e.g.

```
#search {
  display: flex;
  flex-direction: row;
  flex: 2;
  align-self: flex-start;
}
```

- includes values for itself and any child elements
  - if we then add some rulesets for the nested flex items
  - e.g.

```
#searchBox {
  flex: 4;
}

#searchBtn {
  flex: 1;
}
```

- we get a simple proportional split of `4:1` for the input field to the button

---

Image - CSS Flexbox

nested flex containers

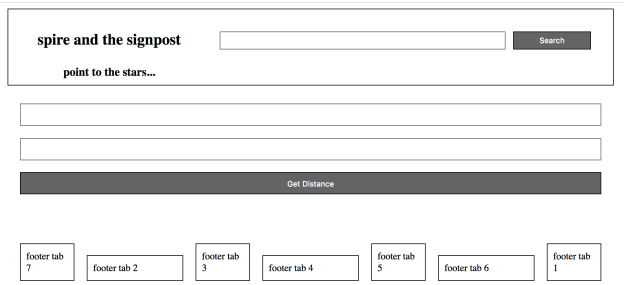

Figure 11: CSS Flexbox - nested flex containers

HTML5, CSS, & JS - example - part 10

add flex to notes - option 2

- define rulesets for child items
  - card-view header
  - card-view footer

```
.card-view header {
    padding: 10px;
    background-color: #666666;
    color: #EEEEEE;
    font-size: 17px;
}
.card-view footer {
    border-top: 1px solid #666666;
    padding: 10px 0;
}
```

- DEMO - Travel Notes - Version 3 - Grid

Image - HTML5, CSS, & JS - Flex Notes

Image - HTML5, CSS, & JS - Flex Notes

CSS grid layout - part 8

media queries

- often need to consider a mobile-first approach
- introduction of CSS3, we can now add **media queries**
- modify specified rulesets relative to a given condition
  - eg: screen size for a desktop, tablet, and phone device
- media queries allow us to specify a breakpoint in the width of the viewport
  - will then trigger a different style for our application
- could be a simple change in styles
  - such as colour, font etc
- could be a modification in the grid layout

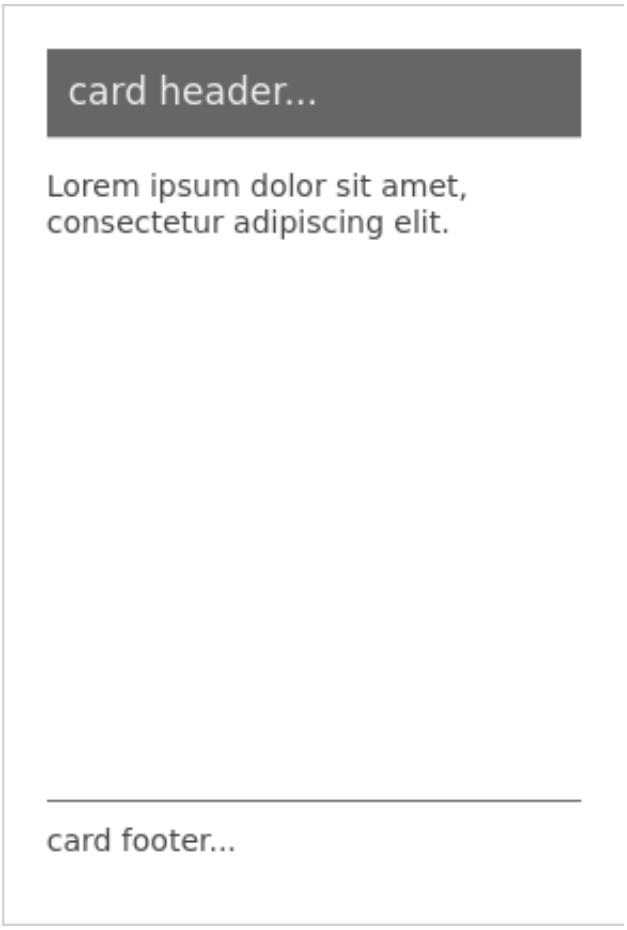

Figure 12: Grid Layout - flex notes - card design

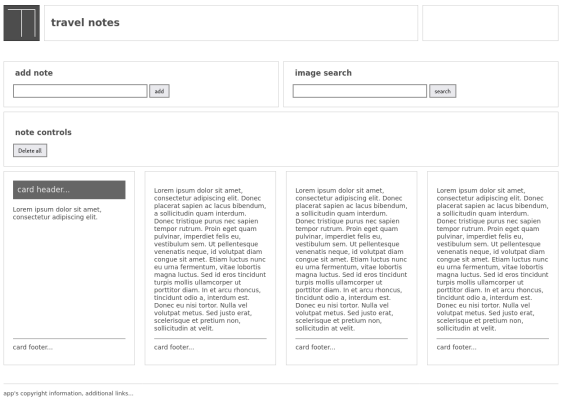

Figure 13: Grid Layout - flex notes - card view with space between

- effective widths for our columns per screen size etc...

Image - Grid Layout 4

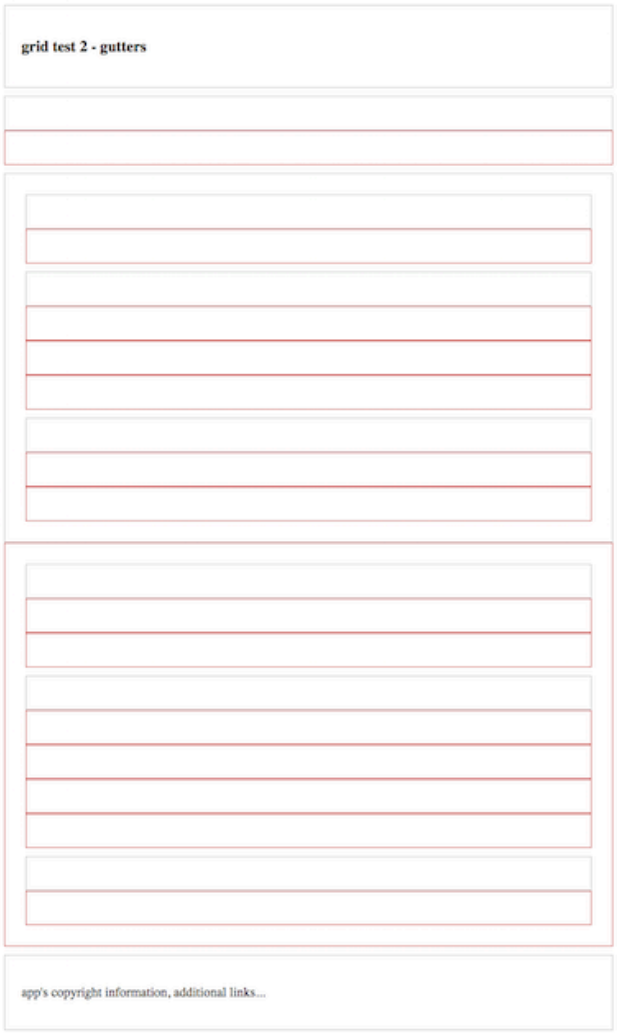

Figure 14: Grid Layout - Media Queries

CSS3 Grid - responsive layout

intro

- display a layout with a variety of patterns and structures, e.g.
  - single column for a phone
  - add a sidebar for a tablet of lower window resolution
  - full width view with dual sidebars &c.
- use responsive designs and structures for various games, media playback...
- responsive works with variety of markup
  - e.g. transform SVG designs

CSS3 Grid - responsive layout

page structure

- start with a sample page structure for a HTML page

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>CSS Grid - Responsive Layout</title>
    <link rel="stylesheet" type="text/css" href="./assets/style.css">
  </head>
  <body>
    <div class="wrapper">
            ...
        </div>
    </body>
</html>
```

CSS3 Grid - responsive layout

page structure - HTML5

- add some HTML5 markup for a `header` , `navigation` , `footer` , and some `main` content

```
<div class="wrapper">
    <header class="site-header">
        <h3>Spire & the Signpost</h3>
        <h5>Shine through the gloom, and point to the stars...</h5>
    </header>
    <nav class="site-nav">
        <ul>
            <li><a href="">Home</a></li>
            <li><a href="">Charts</a></li>
            <li><a href="">Data</a></li>
            <li><a href="">Views</a></li>
        </ul>
    </nav>
    <!-- use aside for tangentially related content for parent section... -->
    <aside class="content-side">
        <header>
            <h5>sidebar...</h5>
        </header>
    </aside>
    <main class="content">
        <article class="content-article">
            <header class="article-header">
                <h5>Welcome</h5>
            </header>
            <p>...</p>
        </article>
    </main>
```

```
    <section class="site-links">
        <h6>social links...</h6>
    </section>
    <footer class="site-footer">
        <h6>footer...</h6>
    </footer>
</div>
```

- demo - basic responsive

CSS3 Grid - responsive layout

CSS and structure - part 1

- for the page structure
  - need to define some template areas for our grid in the CSS
  - e.g.

```
/* CONTENT */
.content {
    grid-area: content;
}
```

- use such template area names
  - defined with the `grid-area` property
  - define a layout for the overall page or part of a page

CSS3 Grid - responsive layout

CSS and structure - part 2

- template areas may then be used with the parent for the grid structure
  - e.g. `wrapper` for the overall page

```
.wrapper {
    display: grid;
    grid-gap: 10px;
    grid-template-areas:
        "site-header"
        "site-nav"
        "content-side"
        "content"
        "site-links"
        "site-footer"
}
```

- `wrapper` class will display as a grid
  - with a gap between each area of the grid
  - has a single column in this example
  - includes the required order for the grid areas

CSS3 Grid - responsive layout

define media query

- current example would be suitable for a collapsed phone view
  - single column view
  - will also render for other resolutions and devices
- then add a media query for alternative layouts and displays
  - may be triggered using a check of current width for screen
  - check width of window…

```
/* min 700 */
@media (min-width: 700px) {
    .wrapper {
        grid-template-columns: 1fr 3fr;
        grid-template-areas:
        "site-header site-header"
        "site-nav site-nav"
        "content-side content"
        "site-links site-footer"
    }
}
```

CSS3 Grid - responsive layout

specific media query

- add further media queries to handle various rendering requirements
  - e.g. add `height` property to fix footer at bottom of page

```
@media (min-width: 700px) {
    .wrapper {
        grid-template-columns: 1fr 3fr;
        grid-template-rows: 120px 60px calc(98vh - 240) 60px;
        grid-template-areas:
        "site-header site-header"
        "site-nav site-nav"
        "content-side content"
        "site-links site-footer";
        height: 98vh;
    }
}
```

- specify height of current *viewport* using a relative unit, `vh`
- add `grid-template-rows` to define known heights for three of the four rows
- add a variant height for the main content
  - main content is only given a height corresponding to available space in viewer window
  - height achieved using the `calc()` function
- demo - responsive with specific media query

HTML5, CSS, & JS - example - part 11

add responsive design - option 2

- add new CSS - `responsive.css`

- update `index.html` with new `<link>`

```
<link rel="stylesheet" href="assets/styles/responsive.css">
```

- define initial media queries, `915px` and `545px`
- 915px for desktop to tablet
- 545px for tablet to phone &c.

HTML5, CSS, & JS - example - part 12

add responsive design - option 2

- media query for 915px
  - update wrapper, banner
  - adjust site-header & banner extras

```
div.wrapper {
    grid-template-rows: 80px auto 80px;
    margin: 20px 10px 0 10px;
}
div.banner {
    grid-template-rows: 80px;
    grid-template-areas:
        "site-logo site-header"
}
.site-header {
    margin-right: 0;
}
.banner-extras {
    display: none;
}
```

- need to update note controls, main content
- e.g. update flex for note card design
  - better use of available width

Image - HTML5, CSS, & JS - Media Query - 915px

HTML5, CSS, & JS - example - part 13

add responsive design - option 2

- update note controls
  - move to stacked rows
  - one row per option, input field

```
.page-heading {
    grid-template-columns: 100%;
    grid-template-areas:
        "add-note"
        "search-images"
        "note-controls";
}
```

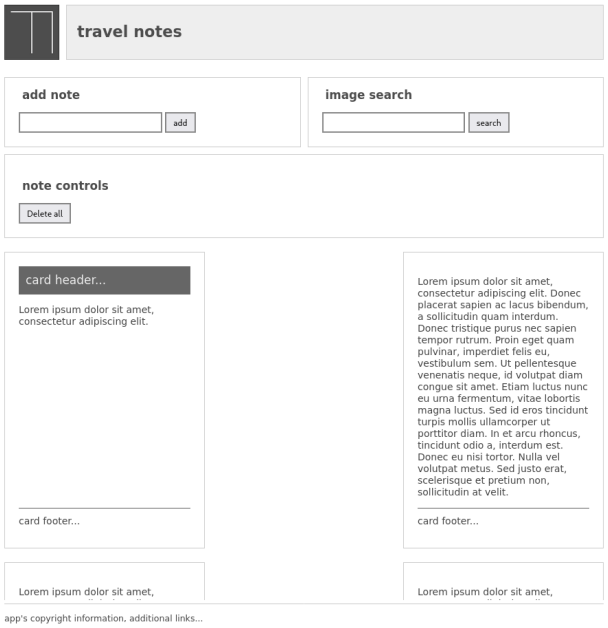

Figure 15: Media Query - 915px - banner

- update HTML for input group
  - container for input field and button

```
<div class="input-group">
    <input type="text" id="input-note" />
    <button id="add-note">add</button>
</div>
```

- update CSS with flex for input group

```
.input-group {
    display: flex;
    justify-content: space-around;
    flex-grow: 1;
}
```

Image - HTML5, CSS, & JS - Media Query - 915px

HTML5, CSS, & JS - example - part 14

add responsive design - option 2

- update input group with modified styles, aesthetics, same sizes &c.
- add ruleset for `input-group` to style.css only
  - style applied regardless of page width

```
/* group input and button */
.input-group {
```

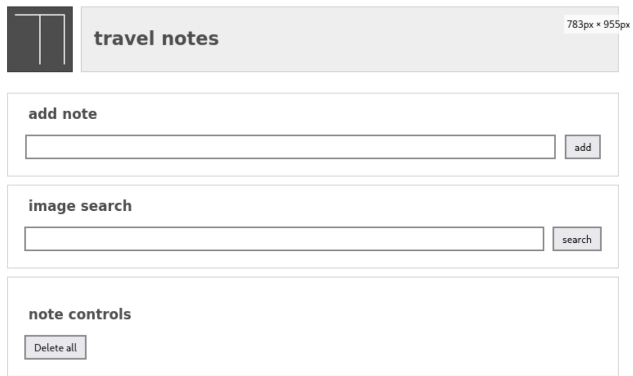

Figure 16: Media Query - 915px - note controls & input

```
    display: flex;
    justify-content: space-around;
    flex-grow: 1;
}
/* input field */
input {
  width: 90%; /* redo with flex to fit space... */
    margin-right: 10px;
  border: 2px solid #888;
    padding: 5px;
}
```

Image - HTML5, CSS, & JS - Media Query - 915px

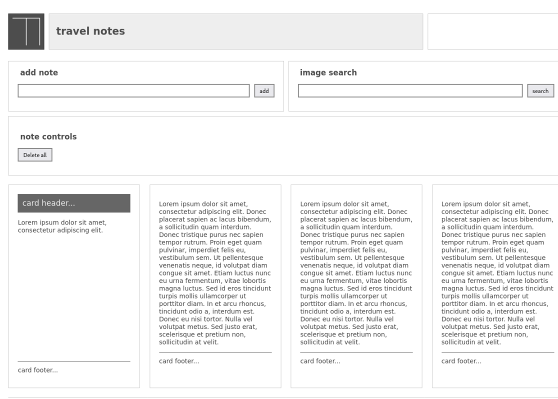

Figure 17: Media Query - 915px - input group - desktop

References

- MDN - CSS Flexbox
- W3 Schools - CSS Flexbox
- Various
  - Example Responsive UI Designs - YouTube
  - MDN - CSS3 Grid
  - Modular UI Design - YouTube