PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

LESSON 5: CSS Grid & Flexbox

HTML 300

OVERVIEW

1. Flexbox Review
2. CSS Grid Review
3. CSS Grid Features

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

Flexbox Review

Flexbox Review

- Why flexbox?
  - With modern browsers (and IE > 9), the flex spec allows us to control the distribution of elements much easier and simpler than using floats
- Usage
  - A great way to think about when to use flexbox is when you have a known/unknown number of elements that you want to distribute among a container in a certain way (spaced evenly, in a column/row at the start/end)
  - Nav links are a great use case for flexbox
    - A mobile stacked nav and desktop horizontal nav can be achieved with a media query and a few lines of flexbox CSS

Flexbox Review

- Axes
  - With flexbox you need to think in terms of two axes — the main axis and the cross axis. The main axis is defined by the flex-direction property, and the cross axis runs perpendicular to it. Everything we do with flexbox refers back to these axes, so it is worth understanding how they work from the outset.
- Direction
  - To work along the axes, use `flex-direction` property to control the flow of content within the flex container.
  - Possible values are `row, column, row-reverse, and column-reverse`

Flexbox Review

- Flex containers:
  - Giving an element display: flex enables the container
  - The justify-content property distributes the content among the main axis (default: row)
  - The align-items property aligns the content along its cross-axis
  - The flex-wrap property determines whether the child elements should wrap if there is no space left, in combination with the child elements flex-basis

```
.flex-container {
  display: flex;
  justify-content: space-between;
  align-items: center;
  flex-wrap: wrap;
}
```

Flexbox Review

- Flex shorthand
  - The ‘flex’ shorthand property has 3 values: flex-grow, flex-shrink, and flex-basis
  - Flex-grow tells the element how much proportionally each child should take in extra space that is left when the container grows
  - Flex-shrink tells the element how much proportionally each child should give up in extra space that is left when the container shrinks
  - Flex-basis sets the initial main size of a flex item, in a value for width

```
.flex-items {
  flex: 1 2 200px;
  // Also, individual properties
  flex-grow: 1;
  flex-shrink: 2;
  flex-basis: 200px;
}
```

Flexbox Review

- Flex-wrap and flex-basis
  - When combined, the children elements will wrap when there is no longer adequate space to place another child within the minimum width of its flex-basis
  - Make sure the container has flex-wrap: wrap (defaults to no-wrap)
  - Use flex-basis in combination with media queries to create responsive, flex-based components that redistribute extra space with far more control that positioning/floats

Flexbox Review

- Justifying content
  - property defines how the browser distributes space between and around content items along the main-axis of a flex container, and the inline axis of a grid container.

```
justify-content: center;     /* Pack items around the center */
justify-content: start;      /* Pack items from the start */
justify-content: end;        /* Pack items from the end */
justify-content: flex-start; /* Pack flex items from the start */
justify-content: flex-end;   /* Pack flex items from the end */
justify-content: left;       /* Pack items from the left */
justify-content: right;      /* Pack items from the right */
```

Flexbox Review

- Align-items
  - This property works differently in flexbox and CSS grid, but within flexbox it sets the alignment of children along the cross axis. Most commonly used are center and flex-start.
  - The container will require a height to vertically align its children

```
/* Basic keywords */
  align-items: normal;
  align-items: stretch;
/* Positional alignment */
  align-items: center; /* Pack items around the center */
  align-items: start; /* Pack items from the start */
  align-items: end; /* Pack items from the end */
  align-items: flex-start; /* Pack flex items from the start */
  align-items: flex-end; /* Pack flex items from the end */
  align-items: self-start;
  align-items: self-end;
```

Flexbox Review

- Order property
  - The order property allows you to explicitly give an order to the children of the flex container.
  - Order takes in a number.
  - This is especially useful for responsive components that change their layout order when going to various breakpoints.
  - This can help keep your code clean and DRY by re-ordering content rather than duplicating and show/hide to accommodate the design.

Flexbox Activity

- Let’s give it a try
  - Clone the activity repo from the https://github.com/UWFront-End-Cert/html300
  - In your terminal, run `npm install`
  - Make sure your packages installed successfully, flag if you need assistance.
  - Now run `gulp` in your terminal.
  - Create the navigation component using SCSS and flexbox.
  - BONUS: Add media queries to make the 'logo' and 'log in' links centered and the nav items spaced between.

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY of WASHINGTON

CSS Grid Review

CSS Grid Review

- **CSS Grid** is a modern syntax for creating layouts without floats.
- CSS Grid is great for projects that don't have strict browser requirements (can use autoprefixer for IE 11 support but it loses features).
- CSS Grid creates rows and columns similar to tables, with many more properties available to define the grid into templated areas.
- CSS Grid also has similar alignment properties as flexbox.

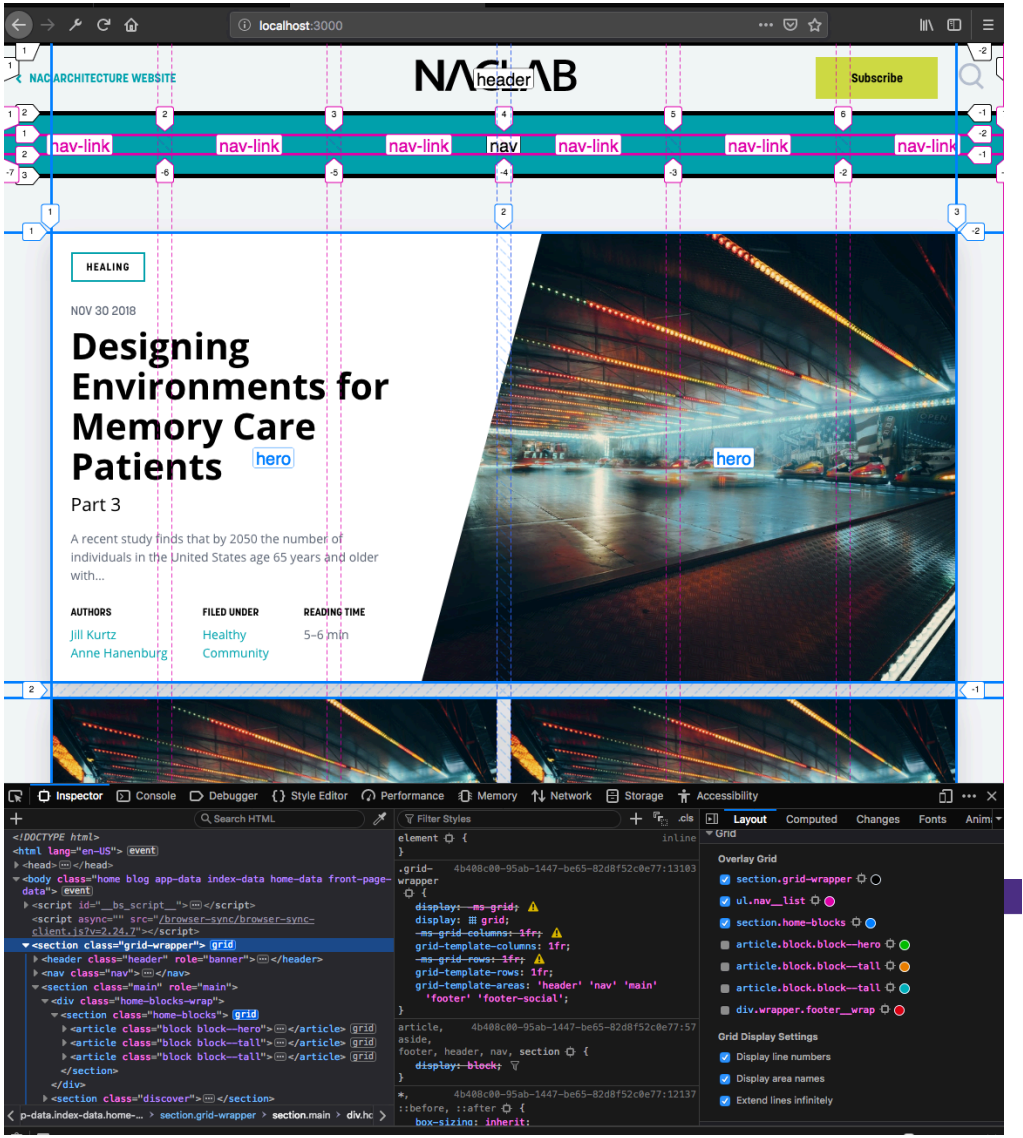
CSS Grid Review

- Developer Tools
  - Since CSS grid is relatively new to browsers, developer tools features have been recently been integrated into Firefox and Chrome.

Firefox has a rich grid inspector that visualizes the grid/rows/cols and labels template areas, the most robust of any browser currently	Chrome also has some grid inspector tools, but they aren't as clean or nice as Firefox

CSS Grid Review

- Firefox grid tools



CSS Grid Review

- The grid
  - CSS Grid uses rows and columns
  - Each cell is split, with a potential for a grid-gap value
    - grid-gap: 1rem; applies to row/col
    - Can shorthand row/col with multiple values
      - grid-gap: 1rem 2rem;
  - Grid-template allows for building out templated grids

```
article-authors {
  display: grid;
  max-width: 1040px;
  margin: auto;
  align-items: center;
  grid-template-columns: 200px 1fr;
  grid-template-rows: 1fr;
  grid-gap: 1rem;
}
```

PROFESSIONAL & CONTINUING EDUCATION

UNIVERSITY *of* WASHINGTON

CSS Grid Features

CSS Grid

- Implicit/Explicit Grid
  - When a cell is defined by a grid-area property, it fits within the explicit grid. If we had a grid with 2 columns and 1 row with 2 child elements given their grid-area matching the template, those would fit within the explicit grid.
  - The implicit grid is any child element not explicitly placed within the grid. This is generated by the browser and can be seen within the dev tools.
  - IE 11 has ONLY an explicit grid, the auto function does not work the same as modern browsers so IE support requires additional work.

CSS Grid

- The ‘fr’ unit
  - CSS Grid utilizes a new unit called fr
  - This stands for fractional unit.
  - An easier way to think of the fr property is as the ‘free space’ unit.
  - Grid columns/rows can use fr units to allow the width to be proportional amounts of the leftover ‘free space’.
  - In the example, the grid would have 2 columns of equal free space.

```
.grid-wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 1fr;
}
```

CSS Grid

- Grid templates
  - Grid templating takes 3 properties:
    - Grid-template-columns: setup the pattern for the column layout
    - Grid-template-rows: setup the pattern for the row layout
    - Grid-template-areas: setup the pattern for the grid areas
      - Child elements that correspond to these can use the property grid-area: with the name of the parent's template-area WITHOUT quotes.

```
.article {
    max-width: 1040px;
    margin: auto;
    display: grid;
    grid-template-areas:
      'article'
      'article';
    grid-template-columns: 1fr 1fr;
    grid-template-rows: 1fr;
    grid-gap: 1rem 2rem;
}
```

CSS Grid

- Grid template area
  - Grid template-areas are arranged within their layout, with quotes around each declared row.
  - These can be updated with media-queries to shift layouts by just changing the grid-template-areas and/or grid-template-columns and grid-template-rows values.

```
.hero {
  max-width: 1280px;
  margin: auto;
  display: grid;
  grid-template-areas:
    'hero-img'
    'hero-content';
  grid-template-columns: 1fr;
  grid-template-rows: 1fr;
  @include media-query(tab) {
    grid-template-areas: 'hero-content hero-
img';
    grid-template-columns: 275px 1fr;
  }
```

CSS Grid

- Grid area
  - The grid-area property is used to place the children elements within the template-area.
  - The areas will also appear labeled within the dev tools which helps with development and debugging.

```
.grid-wrapper {
  display: grid;
  grid-template-columns: 1fr;
  grid-template-rows: 1fr;
  grid-template-areas:
    'header'
    'nav'
    'main'
    'footer'
    'footer-social';
}
.main {
  grid-area: main;
  min-width: 0;
}
```

CSS Grid

- Repeat keyword
  - The grid-template-rows/grid-template-columns have a repeat() function that can be used.
  - Repeat takes 2 parameters, the repeat number (number or auto-fill) and the col/row track pattern.
  - Combined with auto-fill, it can handle implicit grid items easily.

```
.grid-wrapper {
  display: grid;
  grid-template-columns: repeat(2, 1fr);
  grid-template-rows: 1fr;
}
```

CSS Grid

- Minmax keyword
  - The grid-template-rows/grid-template-columns have a minmax() function that can be used in place of a single value.
  - minmax takes 2 parameters, the minimum size and the maximum size
  - The maximum can use fr to declare the max is whatever free space is left in the track
  - Minmax can be used in combination with repeat()

```
.blocks {
    display: grid;
    max-width: 425px;
    margin: auto;
    grid-template-areas:
      'block block'
      'block block'
      'block block';
    grid-template-columns: repeat(2, minmax(140px, 1fr));
    grid-template-rows: 1fr;
    grid-gap: 1rem;
  }
```

CSS Grid

- Auto-fill keyword
  - The grid-template-rows/grid-template-columns repeat function can take in auto-fill as the repeater parameter.
  - By default, an explicit repeater will not wrap its children if there are more elements that fit within the viewport.
  - Auto-fill will wrap those extra cells into new rows within the grid, leaving implicit cells within the grid if not enough children.
  - Doesn't work in IE11

```
.blocks {
  display: grid;
  margin: auto;
  grid-template-columns: repeat(auto-fill, minmax(140px, 1fr));
  grid-template-rows: 1fr;
  grid-gap: 1rem;
}
```

CSS Grid

- Auto-fit keyword
  - The grid-template-rows/grid-template-columns repeat function can take in auto-fit as the repeater parameter
  - By default, an explicit repeater will not wrap its children if there are more elements that fit within the viewport
  - Auto-fit will wrap those extra cells into new rows within the grid, creating equal cells with no empty cells that can't be possible with fill
  - Doesn’t work in IE11

```
.blocks {
  display: grid;
  margin: auto;
  grid-template-columns: repeat(auto-fit, minmax(140px, 1fr));
  grid-template-rows: 1fr;
  grid-gap: 1rem;
}
```

CSS Grid

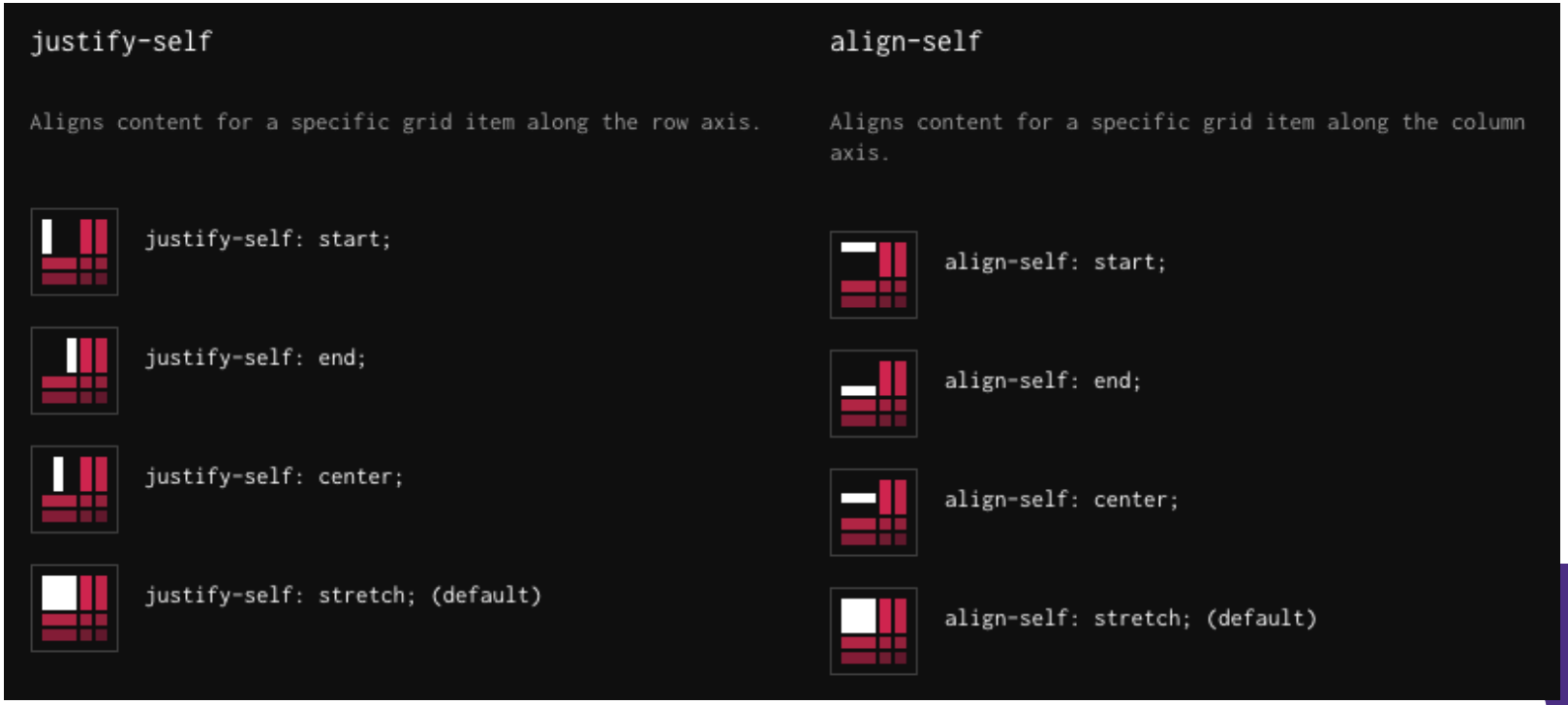
- **Parent Alignment:** justify-content and align-content sounds familiar from flexbox, but work differently with CSS Grid

CSS Grid

- **Item Alignment:** justify-items and align-items controls the children content alignment.

CSS Grid

- **Children Alignment:** justify-self and align-self are able to overwrite the alignment for the children.



CSS Grid

- **Children placement:** the grid-column and grid-row properties determine where the children are placed within the grid.

CSS Grid – More Info

- https://cssgrid.io/
- https://css-tricks.com/snippets/css/complete-guide-grid/
- http://grid.malven.co/
- https://css-tricks.com/css-grid-in-ie-debunking-common-ie-grid-misconceptions/

QUESTIONS

html3@uw.edu

As always feel free to contact the instructional team if you have any questions. They do have a full-time jobs, so they might not get back to you immediately.

If you do not hear back from them in 48 hours, please try again.