HTML5 and CSS3

ICT 580

Ali Khalil, Ph.D.
Computer Science / Software Engineering
University of Calgary - Continuing Education

Lesson 6

Responsive Design part 1

Lesson Outcomes

By the end of this lesson, you will be able to:

- Identify what responsive design is and why it is important
- Create alternate styles depending on the size of the devices screen
- Create a responsive webpage using grids
- Create a flexible container using flexbox

What is Responsive Design

- When you use CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen or device
- Your web page should look good, and be easy to use, on all devices
- Information should be easily accessed on all devices

Why is Responsive Design Important

- **52.2%** of web traffic is accessed via mobile devices
- On average, mobile internet users spend nearly **3 hours** online every day
- Global brands recognize that – **Google generates about 95% of its revenue** through ads
- Ecommerce retail grows about three times faster than brick-and-mortar shops

Feb 2021:
https://hostingtribunal.com/blog/internet-statistics/

CSS3

Media Query

- Media queries ask the browser if something is true and specifies a style based on the response
- Media queries used to create responsive web designs include:
  - @media only screen and (max-width: 600px) {...}
  - @media only screen and (min-width: 768px) {...}
  - @media only screen and (orientation: portrait) {...}
  - @media only screen and (orientation: landscape) {...}

UNIVERSITY OF CALGARY

Media Queries

- Specify images that are appropriate for the screen size
- Manage spacing/measurements for the specific size
- Alter the layout of tables so they stack rather than span many columns
- Only add selectors and properties that must change

```
@media only screen and (min-width: 600px) {
    p {
        font-size: 12px;
    }
}
```

Applies the styles only if the query in brackets is true.

Nested selector

Nested declaration

Create a grid with the display property

- Grid is an alternative layout to positioning elements manually with position, overflow or float
- Defining the display property as a grid tells the browser to treat the div container as a grid

```
.container {
  display: grid;
}
```

CSS3

Create a gap between items – grid-gap

- The gird-gap property defines the of space between grid items both horizontally and vertically
  - **grid-column-gap: value;** – column spacing
  - **grid-row-gap: value;** – row spacing
  - **grid-gap: columnvalue rowvalue;** – shorthand for both with different values
  - **grid-gap: value;** – shorthand for both if they have the same value

Create columns – grid-template-columns

- The grid-template-columns property specifies the number of columns in the grid along with the width of each container
  - **grid-template-columns: auto auto auto;** – creates evenly spaced columns
  - **grid-template-columns: 100px 200px 200px;** – creates defines values for the columns

CSS3

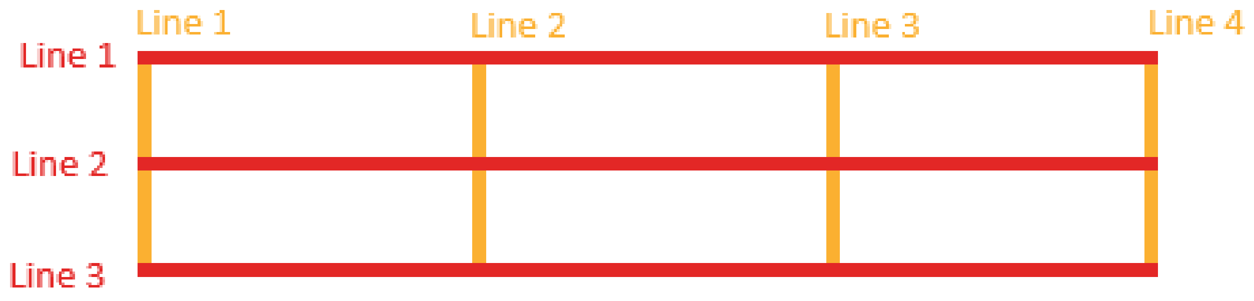
Create rows – grid-template-rows

- Use grid-template-rows to declare the height of each row
    - **grid-template-rows:** 100px 200px; – creates defines height values for the rows

CSS3

Grid lines

- Grid lines numbers declare where an item sits in the grid using the following properties:
  - grid-column-start, grid-column-end, grid-row-start, grid-row-end, grid-column, grid-row and grid-area

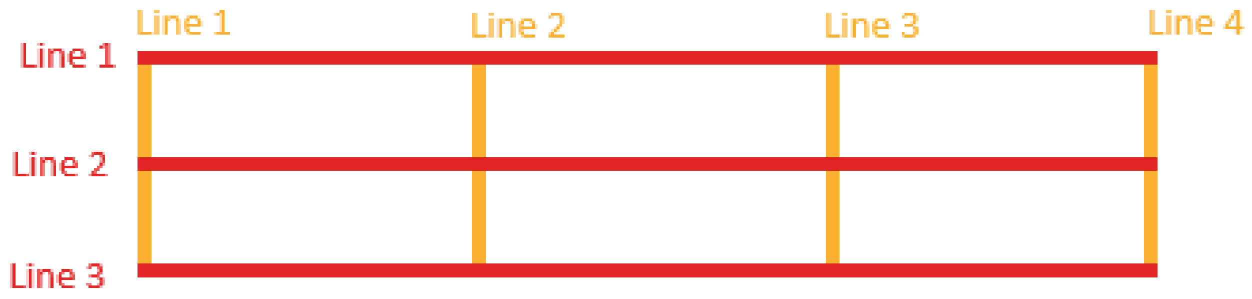


Column lines – orange

Row lines - red

grid-column-start and grid-column-end

- Grid-column-start, grid-column-end properties define where a grid item starts and ends using column lines
  - **grid-column-start: 1**; – start at the first column line
  - **grid-column-end: 3**; – end at the third column line



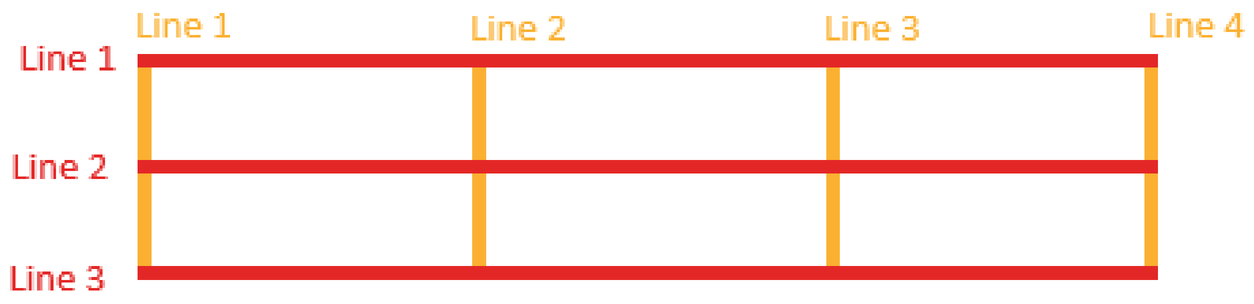
Column lines – orange

Row lines - red

CSS3

grid-row-start and grid-row-end

- Grid-row-start, grid-row-end properties define where a grid item starts and ends using row lines
  - **grid-row-start: 1;** – start at the first column line
  - **grid-row-end: 3;** – end at the third column line



Column lines – orange

Row lines - red

Grid – grid-column

- Grid-column is the shorthand for the grid-column-start and grid-column-end properties
  - **Grid-column:** startvalue / endvalue;

CSS3

Grid – grid-row

- Grid-row is the shorthand for the grid-row-start and grid-row-end properties
    - **Grid-row:** startvalue / endvalue;

CSS3

Grid – grid-area

- Grid-area is the shorthand for both the grid-row-start and grid-column-start and the grid-row-end and grid-column-end properties
  - **grid-area:** rowstartvalue/columnstartvalue /rowendvalue/columnendvalue;

Grid – grid-template-areas

- Defining the grid area using names can make it easier to easily see the pattern of your grid by laying out the pattern of the grid using named values

Grid – grid-area

- The grid-area defines what you want your grid item to be named
- Using these names, you can create a pattern for your div using the grid-template-areas property
  - grid-area: name;

Grid alignment – justify-content property

- The justify-content property defines how the items are distributed within the container when there is **extra** horizontal space
  - **justify-content: space-evenly;** – the items are spaced evenly **justify-content: space-around;** – the items are spaced evenly **justify-content: space-between;** – the outer items are aligned on the outside edge of the container and remaining items have equal space between
  - **justify-content: center;** – the items are centered inside the container separated by the grid-gap space
  - **justify-content: start;** – the items are left justified inside the container separated by the grid-gap space
  - **justify-content: end;** the items are right justified inside the container separated by the grid-gap space

Grid Alignment – align-content property

- The align-content property defines how the items are distributed vertically within the container when there is **extra** vertical space
  - **align-content: space-evenly;** – the vertical spacing is spread out evenly **align-content: space-around;** – the vertical spacing is spread out evenly **align-content: space-between;** – the outer items are aligned on the top and bottom edges and remaining items have equal space between
  - **align-content: center;** – the vertical spacing of items are centered inside the container separated by the grid-gap space
  - **align-content: start;** – the vertical spacing of items are left justified inside the container separated by the grid-gap space
  - **align-content: end;** – the vertical spacing of items are right justified inside the container separated by the grid-gap space

Flexbox

- Flexbox is another alternative to positioning elements manually with position, overflow or float.
- The flex layout specifies how to handle the either the row or the column not both.
  - **Main axis** – the axis that the items are distributed along, specified by the flex-direction property.
  - **Cross axis** – perpendicular to the main axis
  - **Cross size** – the width or height of a flex item depending on what the cross axis is. If the main axis is row, the cross axis is column so the cross size would be specified as height.

CSS3

Create a flexbox with the display property

- Defining the display property as a flex tells the browser to treat the div container as a flexbox

```
.container {
  display: flex;
}
```

CSS3

Setting the Flexbox direction – row

- The flexbox-direction property sets the main axis or how the items will be laid out either horizontally or vertically
  - **flex-direction: row;** – sets the main axis to be a single row of columns
  - **flex-direction: row-reverse;** – sets the main axis to be a single row of columns that runs in the reverse direction

Setting the flex-wrap property

- The flex-wrap property declares if the items can wrap to the next line if they run out of room
  - flex-wrap: wrap;
  - flex-wrap: nowrap;
  - flex-wrap: wrap-reverse;
- The default value id nowrap

Setting the flex-flow property

- The flex-flow property lets you declare both the direction and the wrap properties
  - flex-flow: row wrap;
  - flex-flow: row nowrap;

Setting the flex-grow property

- The flex-grow property declares how the item will grow in relation to the other items in the container
  - **flex-grow: 1;** – if all items are set to 1 they grow equally
  - **flex-grow: 2;** – if all items are set to 1 and one of them is set to 2 the items set to 1 would grow equally and the item set to 2 would grow at 2 times the rate of the others

Setting the flex-shrink property

- The flex-shrink property declares how the item will shrink in relation to the other items in the container
- Items that have a higher shrink value will shrink more than items with a lower shrink value
  - **flex-shrink: 1;** – if all items are set to 1 they shrink equally
  - **flex-shrink: 2;** – if all items are set to 1 and one of them is set to 2 the items set to 1 would shrink equally and the item set to 2 would shrink at 2 times the rate of the others

Setting the flex-basis property

- The flex-basis property declares the **size** of the item
  - flex-basis: 100px;
  - flex-basis: 60%;

- *Depending on the flex direction (row, column), the **size** can be understood to be as either height or width.*

Setting the Flexbox direction – column

- The flexbox-direction property sets the main axis or how the items will be laid out either horizontally or vertically
  - **flex-direction: column;** – sets the main axis to be a single column of rows
  - **flex-direction: column-reverse;** – sets the main axis to be a single column of rows that runs in the reverse direction

CSS3

Setting the flex property

- flex is shorthand for flex-grow, flex-shrink and flex basis. 1 indicates equal growing or shrinking and the basis sets the initial height to 60%.
    - **flex:** 1 1 60%;

Setting the order property

- The order property allows you to change the order of the items
- All items in the container must be redefined for this property to function

Review

In this class we discussed:

- How to create alternate styles depending on the size of the screen with media queries
- How to create a responsive layout with grid
- How to create a responsive layout with flexbox