

Comp 125 - Visual Information Processing

Spring Semester 2019 - Week 13 - Wednesday

Dr Nick Hayward

CSS grid layout - example - part 6

grid.css

- add gutters to our grid to help create a sense of space and division in the content
- simplest way to add a gutter to the current grid css is to use padding
 - *rows can use padding, for example*

```
.row {  
  padding: 5px;  
}
```

- issue with simply adding padding to the columns
 - *margins are left in place, next to each other*
 - *column borders next to each with no external column gutter*
- fix this issue by targeting columns that are a sibling to a preceding column
- means we do not need to modify the first column, only subsequent siblings

```
[class*="col-"] + [class*="col-"] {  
  margin-left: 1.6%;  
}
```

Image - Grid Layout 2

grid test 2 - gutters



app's copyright information, additional links...

Grid Layout - Gutters Overflow

CSS grid layout - part 7

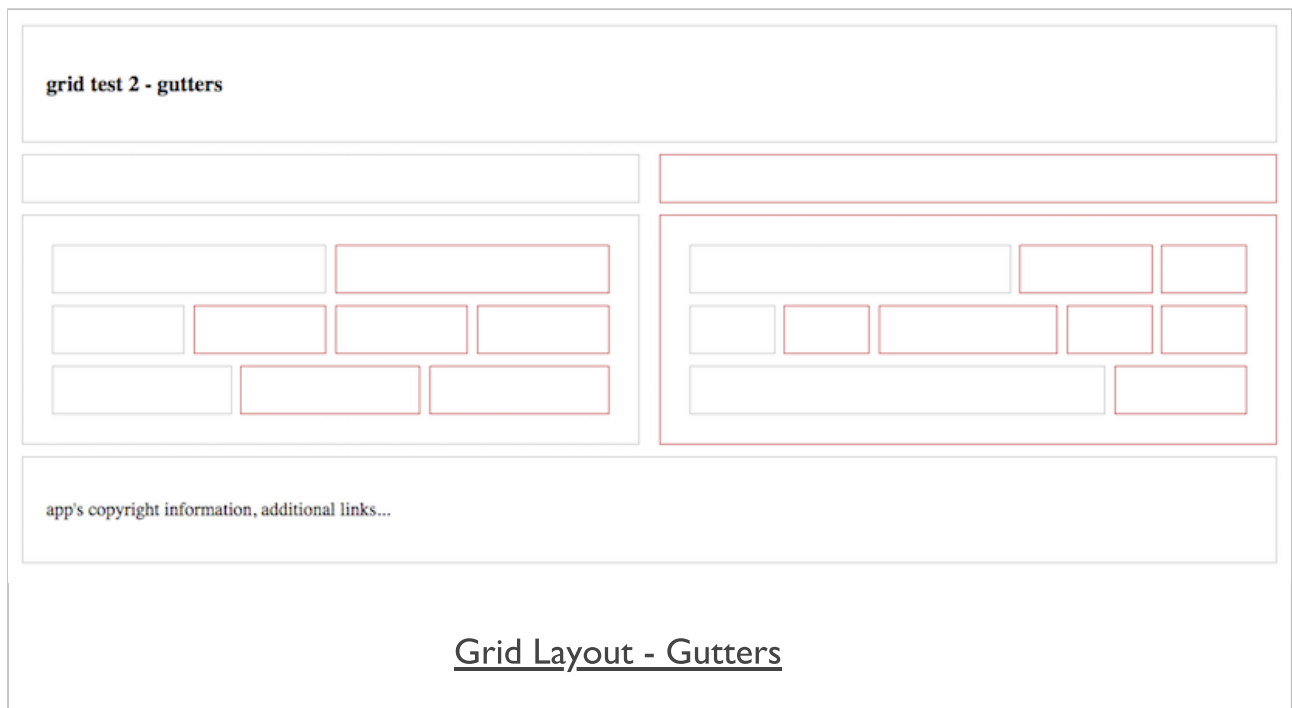
grid.css

- to fix this issue we recalculate permitted % widths for our columns in the CSS
 - *we now have % widths as follows*

```
.col-1 {width: 6.86%;}  
.col-2 {width: 15.33%;}  
.col-3 {width: 23.8%;}  
.col-4 {width: 32.26%;}  
.col-5 {width: 40.73%;}  
.col-6 {width: 49.2%;}  
.col-7 {width: 57.66%;}  
.col-8 {width: 66.13%;}  
.col-9 {width: 74.6%;}  
.col-10 {width: 83.06%;}  
.col-11 {width: 91.53%;}  
.col-12 {width: 100%;}
```

- DEMO - Grid Layout 2 - gutters

Image - Grid Layout 3



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

```
@media only screen and (max-width: 900px) {  
  [class*="col-"] {  
    width: 100%;  
  }  
}
```

- gutters need to be removed
 - *specifying widths of 100% for our columns*

```
[class*="col-"] + [class*="col-"] {  
  margin-left:0;  
}
```

Image - Grid Layout 4



HTML5, CSS, & JS - example - part I

Structure

- combine HTML5, CSS, and JavaScript, to create an example application
- outline of our project's basic directory structure

```
.
|- assets
|  |- images //logos, site/app banners - useful images for site's design
|  |- scripts //js files
|  |- styles //css files
|- docs
|  |- json //any .json files
|  |- txt //any .txt files
|  |- xml //any .xml files
|- media
|  |- audio //local audio files for embedding & streaming
|  |- images //site images, photos
|  |- video //local video files for embedding & streaming
|- index.html
```

- each of the above directories can, of course, contain many additional sub-directories
 - */- images may contain sub-directories for albums, galleries...*
 - */- xml may contain sub-directories for further categorisation..*
 - *and so on...*

HTML5, CSS, & JS - example - part 2

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>travel notes - v0.1</title>
    <meta name="description" content="information on travel destinations">
    <meta name="author" content="ancientlives">
    <!-- css styles... -->
    <link rel="stylesheet" type="text/css" href="assets/styles/style.css">
  </head>
  <body>
    ...
    <!-- js scripts... -->
    <script type="text/javascript" src="assets/scripts/jquery.min.js"></script>
    <script type="text/javascript" src="assets/scripts/travel.js"></script>
  </body>
</html>
```

- JS files at foot of body
 - *hierarchical rendering of page by browser - top to bottom*
 - *JS will now be one of the last things to load*
 - *JS files often large, slow to load*
 - *helps page load faster...*

HTML5, CSS, & JS - example - part 3

index.html - body

```
<body>
  <!-- document header -->
  <header>
    <h3>travel notes</h3>
    <p>record notes from various cities and placed visited...</p>
  </header>
  <!-- document main -->
  <main>
    <!-- note input -->
    <section class="note-input">
    </section>
    <!-- note output -->
    <section class="note-output">
    </section>
  </main>
  <!-- document footer -->
  <footer>
    <p>app's copyright information, additional links...</p>
  </footer>
  <!-- js scripts... -->
  <script type="text/javascript" src="assets/scripts/jquery.min.js"></script>
  <script type="text/javascript" src="assets/scripts/travel.js"></script>
</body>
```

HTML5, CSS, & JS - example - part 4

style.css

```
body {  
    width: 850px;  
    margin: auto;  
    background: #fff;  
    font-size: 16px;  
    font-family: "Times New Roman", Georgia, Serif;  
}  
h3 {  
    font-size: 1.75em;  
}  
header {  
    border-bottom: 1px solid #dedede;  
}  
header p {  
    font-size: 1.25em;  
    font-style: italic;  
}  
footer p {  
    font-size: 0.8em;  
}
```

HTML5, CSS, & JS - example - part 5.1

travel.js

```
//overall app logic and loader...
function travelNotes() {
    "use strict";

    $(".note-output").html("<p>first travel note for Marseille...</p>");
};

$(document).ready(travelNotes);
```

- a simple JS function to hold the basic logic for our app
- call this function any reasonable, logical name
- in initial function, we set the `strict` pragma
- add an example call to the jQuery function, `html ()`
 - *sets some initial note content*
- function `travelNotes ()` loaded using the jQuery function `ready ()`
 - *many different ways to achieve this basic loading of app logic*

HTML5, CSS, & JS - example - part 5.2

travel.js - plain JS

```
function travelNotes() {  
  "use strict";  
  
  // get a reference to `.note_output` in the DOM  
  // n.b. these can be combined as well...  
  let noteOutput = document.querySelector('.note-output');  
  noteOutput.innerHTML = '<p>first travel note for Marseille...</p>';  
  
}  
  
// load app  
travelNotes();
```

- DEMO I - travel notes - series I