Comp 125 - Visual Information Processing

Spring Semester 2018 - week 14 - friday

Dr Nick Hayward

HTML Canvas - add an image

basic image rendering - part 1

- draw image to canvas using `drawImage()` method

```
// image drawn full size from source to x & y in destination
context.drawImage(image, dx, dy)
// image drawn with scaled width and height for destination
context.drawImage(image, dx, dy, dw, dh)
// image drawn with source cropped...
context.drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)
```

- `d` represents the destination canvas
- `s` represents the source image

HTML Canvas - add an image

basic image rendering - part 2

- add a static image using `drawImage()` method
- use `Image()` constructor to create an image object
- use `img` object to set `src` for image file
  - *local and remote URL for image is OK*
- draw image to context
  - *`context.drawImage(image, dx, dy)`*

```
// 1. define optional image size
var img = new Image();

// image source file
img.src = './assets/images/philae1.jpg';

img.onload = function() {
  context.drawImage(img, 0, 0);
}
```

- image is not scaled to canvas width and height
- Example - draw image to canvas from local file
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-image/basic1/*

HTML Canvas - add an image

basic image rendering - part 3

- draw image to canvas with scaled source image
  - *`context.drawImage(image, dx, dy, dw, dh)`*

```
// 1. define optional image size
var img = new Image();

// image source file
img.src = './assets/images/philae1.jpg';

img.onload = function() {
    // context.drawImage(image, dx, dy, dw, dh)
  context.drawImage(img, 0, 0, 116, 77);
}
```

- Example - draw image to canvas from local file - `dw` & `dh`
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-image/basic2/*

HTML Canvas - add an image

basic image rendering - part 4

- draw part of the source image
- define source x, y, width and height
  - *i.e. crop part of source image*
- define destination x, y, width and height
  - *i.e. where to draw image on canvas*
  - *& scaled size on canvas*

```
// image source file
img.src = './assets/images/philae1.jpg';

img.onload = function() {
    // context.drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh)
  context.drawImage(img, 200, 200, 232, 144, 0, 0, 464, 288);
}
```

- Example - draw image to canvas from local file - `dw` & `dh` plus source crop
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-image/basic3/*

HTML Canvas - Canvas interaction

move a ball with keyboard controls

- create a new example to allow a user to move a ball
  - *move ball around canvas using keyboard controls*
- requirements include
  - *need to draw a **ball***
  - *listen for specific keypress commands, e.g UP, DOWN, LEFT, RIGHT*
  - *then update animation of ball to reflect each keypress*
- allowing a user to directly control animation of shape on canvas
- setup our initial example with a `canvas` and `context`
  - *use `Ball` constructor and **prototype** methods*
  - *start to add logic to control the `ball`, update animation...*
  - *extend the prototype for user control of the `ball` object*

HTML Canvas - Canvas interaction

keyboard listeners

- add listeners to the `canvas` for specifc keypress events
  - *e.g. up, down, `left`, and `right`*

```
// add event listener for keypress - e.g. up arrow key...
window.addEventListener('keydown', function (event) {
    // get code for key presses
  var key = event.keyCode;
  console.log("key pressed = " + key);
  ball.userControl(key);
})
```

- each keypress event returns a unique code
  - *use code to identify key pressed by user*
  - *`37` = LEFT arrow*
  - *`38` = UP arrow*
  - *`39` = RIGHT arrow*
  - *`40` = DOWN arrow*
- call `userControl()` method for each keypress

HTML Canvas - Canvas interaction

extend Ball prototype - `userControl()`

```
// 4. update prototype - user control
Ball.prototype.userControl = function( key ) {
  // key - UP arrow
  if (key === 38) {
    this.xSpeed = 0;
    this.ySpeed = -10;
    context.clearRect(0, 0, 400, 400);
    ball.draw();
    ball.move();
  }
};
```

- conditional check for key code - 38 = UP arrow
  - *`x` set to `0` to prevent horizontal move*
  - *`y` set to `-10` to move up canvas*
  - *canvas cleared to allow animation frames to be drawn*
  - *call **prototype** method `draw()` on `ball` object*
  - *call **prototype** method `move()` on `ball` object*
- Example - move ball with keyboard control
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move11*

HTML Canvas - Canvas interaction

update `move()` method

- update `move()` to check `canvas` boundaries
- stop `ball` from leaving canvas

```
// check ball relative to boundaries - canvas edge
  if (this.x < 0) {
      this.x = canvas.width;
  } else if (this.x > canvas.width) {
      this.x = 0;
  } else if (this.y < 0) {
      this.y = canvas.height;
  } else if (this.y > canvas.height) {
    this.y = 0;
  }
```

- Example - update `move()` to check canvas boundaries
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move2/*

HTML Canvas - Canvas interaction

abstract width and height

- canvas height and width need to be used throughout JS logic
  - *abstract to variables*

```
// define canvas width and height
var cHeight = canvas.height;
var cWidth = canvas.width;
```

HTML Canvas - Canvas interaction

update `userControl()` method

```
// 4. update prototype - user control
Ball.prototype.userControl = function( key ) {
  /*
    * 37 = LEFT
    * 38 = UP
    * 39 = RIGHT
    * 40 = DOWN
    */
  if (key === 37) {
        ball.userMove(-15, 0);
  } else if (key === 38) {
        ball.userMove(0, -15);
    } else if (key === 39) {
        ball.userMove(15, 0);
    } else if (key === 40) {
        ball.userMove(0, 15);
    }
};
```

- add conditional check for **four** keys
  - *LEFT, UP, RIGHT, DOWN*
- abstract user actioned movement of `ball`
- add `userMove()` method to `Ball` **prototype**

HTML Canvas - Canvas interaction

add `userMove()` method to `Ball` prototype

```
// 5. update prototype - user movement of ball
Ball.prototype.userMove = function (xS, yS) {
    // clear canvas for animation
    context.clearRect(0, 0, cWidth, cHeight);
  // update x and y speed
    this.xSpeed = xS;
  this.ySpeed = yS;
    // draw ball and move...
  ball.move();
    ball.draw();
}
```

- accept parameter for speed along X and Y axis
- clear canvas - use variables for canvas width and height
- call `move()` method on `ball` object
- call `draw()` method on `ball` object
  - *Example - move ball on 4-point axis*
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move3/*

HTML Canvas - Canvas interaction

add image as shape to move - part 1

- abstract drawing required image to canvas
- need to call this function for each animation frame

```
// define sprite draw function
function drawSprite(dx, dy) {
    // 1. define optional image size
  var img = new Image();

  // image source file
  img.src = './assets/images/player.png';

  img.onload = function() {
      // context.drawImage(image, dx, dy, dw, dh)
    context.drawImage(img, dx-30, dy-40, 60, 40);
  }
}
```

- dx and dy passed as parameter values
  - *minus image width and height to set start position for animation*

HTML Canvas - Canvas interaction

add image as shape to move - part 2

- extend prototype for `Sprite`
  - *add `draw()` method*
  - *call `drawSprite()` method - pass start `x` & `y`*

```
// 1. update prototype - method to draw sprite
Sprite.prototype.draw = function () {
  // draw image as sprite - specify start x and y coordinates
  drawSprite(this.x, this.y);
};
```

- Example - move sprite image
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-sprite-move1/*

HTML Canvas - check collisions

add blocks with colour - part 1

- draw some blocks for internal collision
  - *define array with objects*
  - *specify `x`, `y`, `width`, `height`, `color` for blocks*

```
// define game blocks
var blockDetails = [
  {
        x: 25,
        y: 25,
        width: 50,
    height: 10,
        color: 'blue'
    },
  {
        x: 150,
        y: 175,
        width: 50,
    height: 10,
        color: 'red'
    }
];
```

HTML Canvas - check collisions

add blocks with colour - part 2

- add custom function to draw blocks to canvas

```
function drawBlocks(blocks) {
    // iterate through blocks
  for (i = 0; i < blocks.length; i++) {
        context.fillStyle = blocks[i]['color'];
      context.fillRect(blocks[i]['x'], blocks[i]['y'], blocks[i]['width'], blocks[i]['height']);
  }
}
// draw blocks to canvas
drawBlocks(blockDetails);
```

- pass array as parameter to function
- iterate through array of `blocks`
- set `fillStyle` for block to draw
- draw a rectangle to canvas for block `x`, `y`, `height`, and `width`
  - *Example - move sprite image*
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move4/*

HTML Canvas - check collisions

internal canvas collisions - part I

- check `ball` position against block position
  - *`x` and `y` against block values*

```
// 3. update prototype - check collision
Ball.prototype.checkCollision = function ( blocks ) {
  // iterate through blocks and check collision
    for (i = 0; i < blocks.length; i++) {
        // start start and end of block - x & y axis
        let blockStartX = blocks[i]['x'];
        let blockEndX = (blocks[i]['x'] + blocks[i]['width']);
        let blockStartY = blocks[i]['y'];
        let blockEndY = (blocks[i]['y'] + blocks[i]['height']);
        // check block collisions - allow for radius of ball
        if (this.x >= blockStartX-5 && this.x <= blockEndX+5 && this.y >= blockStartY-5 && this.y <= blockEndY+5) {
            console.log('collision at block = ' + this.x);
        }
  }
}
```

HTML Canvas - check collisions

internal canvas collisions - part 2

- call this method in the `userMove()` method

```
// check collisions
    ball.checkCollision(blockDetails);
```

- Example - check collision against blocks
  - *http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move5/*

HTML Canvas - Canvas interaction

update movement to 8-point axis

- a player may also use other available combinations to move the shape
  - *at one of 4 available angles of 45 degrees...*

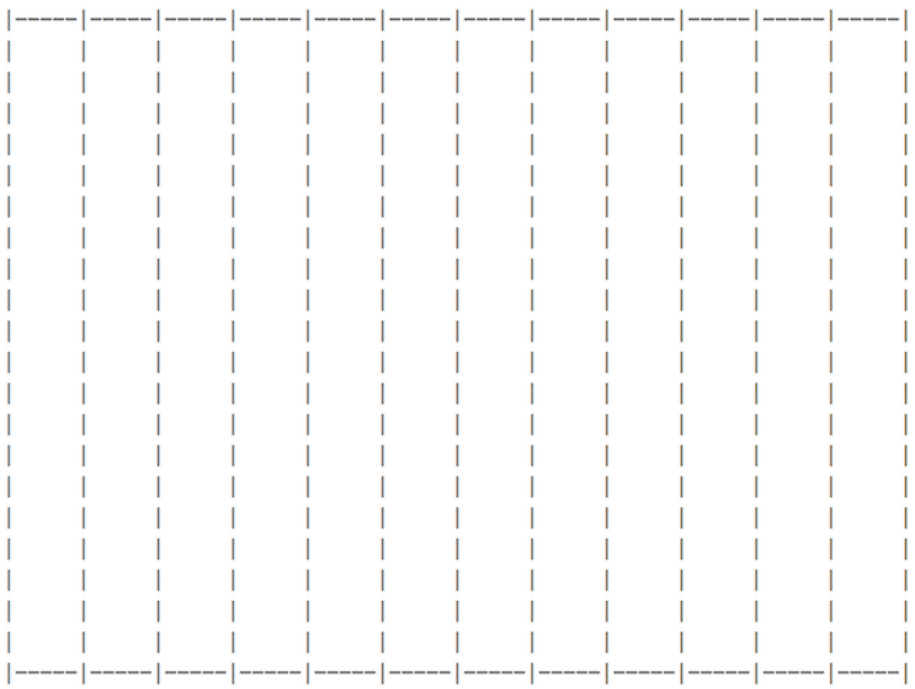
```
-x -y         +x -y
     \   |   /
      \  |  /
  ---- shape ----
      /  |  \
     /   |   \
-x +y         +x +y
```

CSS grid layout - part 1

intro

- grid designs for page layout, components...
  - *increasingly popular over the last few years*
  - *useful for creating responsive designs*
- quick and easy to layout a scaffolding framework for our structured content
- create boxes for our content
  - *then position them within our grid layout*
- content can be stacked in a horizontal and vertical manner
  - *creating most efficient layout for needs of a given application*
- another benefit of CSS grids is that they are framework and project agnostic
  - *thereby enabling easy transfer from one to another*
- concept is based upon a set number of columns per page with a width of 100%
- columns will increase and decrease relative to the size of the browser window
- also set break points in our styles
  - *helps to customise a layout relative to screen sizes, devices, aspect ratios...*
  - *helps us differentiate between desktop and mobile viewers*

Image - Grid Layout



Grid Layout - Columns and rows

CSS grid layout - part 2

grid.css

- build a grid based upon 12 columns
  - *other options with fewer columns as well*
- tend to keep our grid CSS separate from the rest of the site
  - *maintain a CSS file just for the grid layout*
- helps abstract the layout from the remaining styles
  - *makes it easier to reuse the grid styles with another site or application*
- add a link to this new stylesheet in the `head` element of our pages

```
<link rel="stylesheet" type="text/css" href="assets/styles/grid.css">
```

or

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

- ensure padding and borders are included in total widths and heights for an element
  - *reset `box-sizing` property to include the `border-box`*
  - *resetting box model to ensure padding and borders are included*

```
* {
box-sizing: border-box;
}
```

CSS grid layout - example - part 3

grid.css

- set some widths for our columns, 12 in total
  - *each representing a proportion of the available width of a page*
  - *from a 12th to the full width of the page*

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

- classes allow us to set a column span for a given element
  - *from 1 to 12 in terms of the number of grid columns an element may span*

CSS grid layout - example - part 4

grid.css

- then set some further styling for each abstracted `col-` class

```
[class*="col-"] {
  position: relative;
  float:left;
  padding: 20px;
  border: 1px solid #333;
}
```

- create columns by wrapping our content elements into rows
- each row always needs 12 columns

```
<div class="row">
  <div class="col-6">left column</div>
  <div class="col-6">right column</div>
</div>
```

CSS grid layout - example - part 5

grid.css

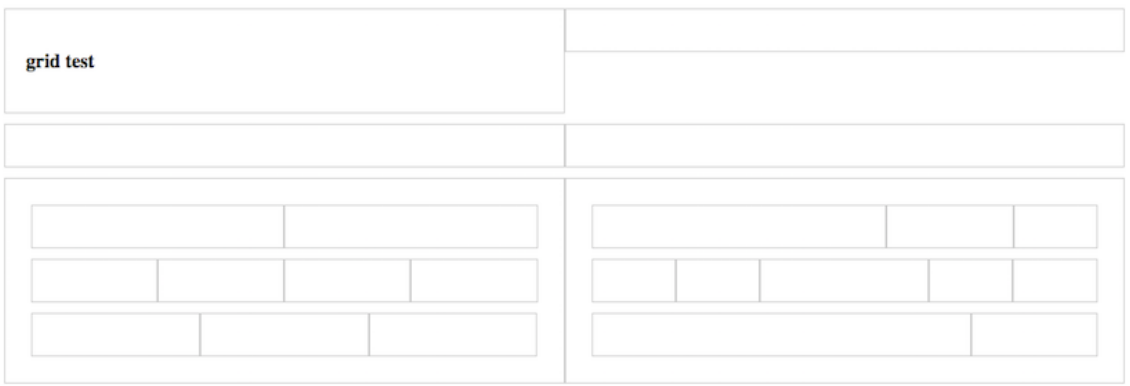
- due to the initial CSS of float left, each column is floated to the left
- columns are interpreted by subsequent elements in the hierarchy as non-existent
  - *initial placement will reflect this design*
- prevent this issue in layout, add the following CSS to grid stylesheet

```
.row:before, .row:after {
  content: "";
  clear: both;
  display: block;
}
```

- benefit of the clearfix, `clear: both`
  - *make row stretch to include columns it contains*
  - *without the need for additional markup*

DEMO - Grid Layout 1 - no gutters

Image - Grid Layout I



Grid Layout - No Gutters

CSS grid layout - example - part 6

grid.css

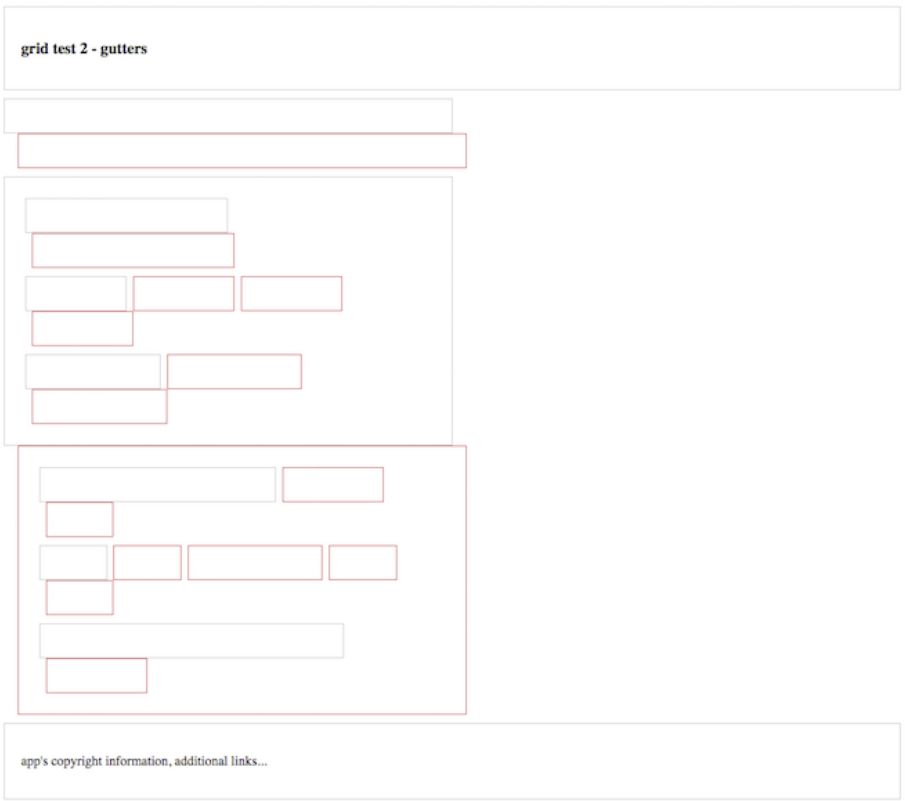
- add gutters to our grid to help create a sense of space and division in the content
- simplest way to add a gutter to the current grid css is to use padding
  - *rows can use padding, for example*

```
.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 Layout - Gutters Overflow

CSS grid layout - part 7

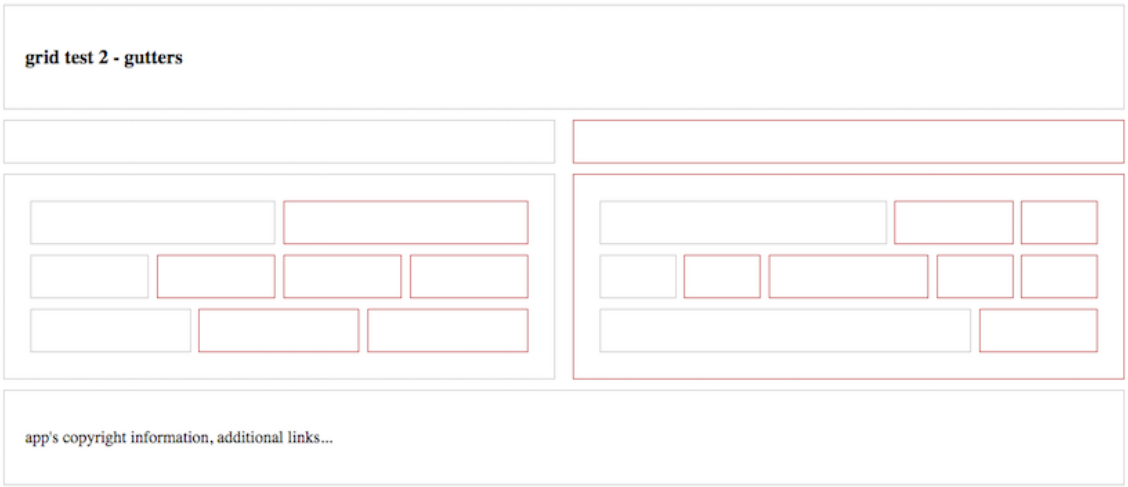
grid.css

- 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



Grid Layout - Gutters

CSS grid layout - part 8

media queries

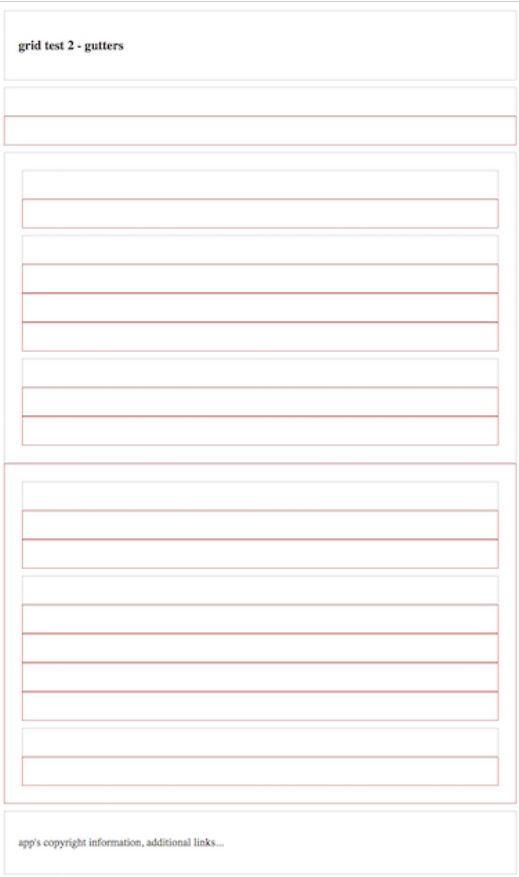
- often need to consider a mobile-first approach
- introduction of CSS3, we can now add **media queries**
- modify specified rulesets relative to a given condition
  - *eg: screen size for a desktop, tablet, and phone device*
- media queries allow us to specify a breakpoint in the width of the viewport
  - *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



Grid Layout - Media Queries

JS ES6 - template literals

```
// create object
var object = {
  archive: 'waldzell',
  access: 'castalia',
  purpose: 'gaming'
};

// log output with template literals
console.log(`archive is ${object.archive}`);

// log output
console.log('archive is ' + object.archive);

// log output all object properties with template literals
console.log(`archive = ${object.archive}, access = ${object.access},  purpose = ${object.purpose}`);

// log output all object properties
console.log('archive = ' + object.archive + ', access = ' + object.access + ' purpose = ' + object.purpose);
```

References

- MDN JS - keyboard event
- W3Schools - HTML5
- media elements
- canvas element
- W3Schools - JS
- event listener
- MDN - CSS
- CSS documentation
- W3
- CSS Flexible Box Layout Module 1
- W3Schools - CSS
- CSS