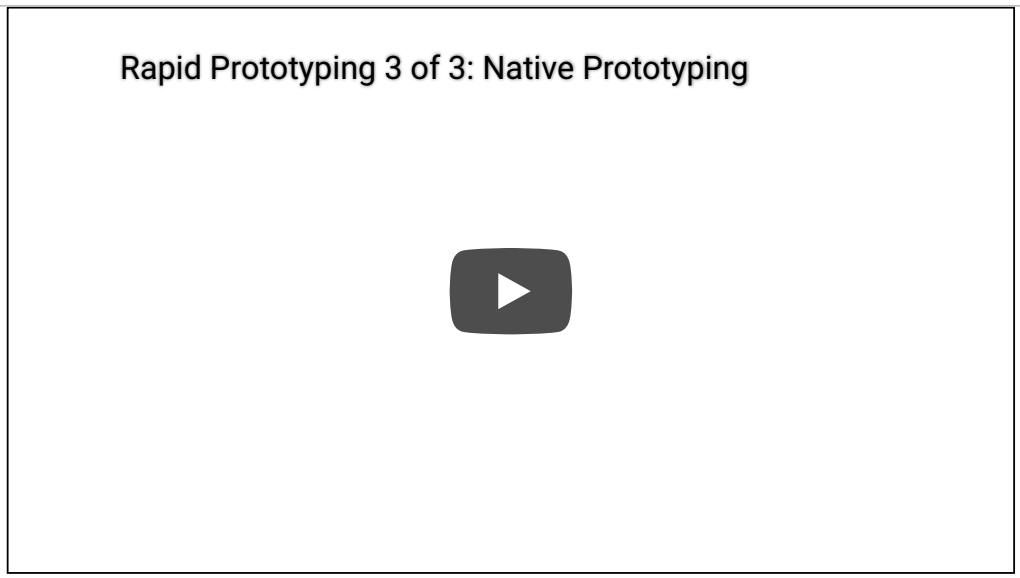
Comp 125 - Visual Information Processing

Spring Semester 2019 - Week 13 - Monday

Dr Nick Hayward

Video - Design

Native Prototyping



Rapid Prototyping 3 of 3: Native Prototyping
Source: YouTube - Google

JavaScript - Object Prototype

intro

- a *prototype* object may be used to delegate the search for a particular property
- i.e. a *prototype* is a useful and convenient option
  - *used for defining properties and functionality accessible to other objects*
- useful option for replicating many concepts in object oriented programming

JavaScript - Object Prototype

understanding prototypes - part I

- in JS, we may create objects
  - *e.g. using object-literal notation*

```
let testObject = {
    property1: 1,
    property2: function() {},
    property3: {}
}
```

- in this object we have
  - *a simple value for the first property*
  - *a function assigned to the second property*
  - *and another object assigned to the third object*

JavaScript - Object Prototype

understanding prototypes - part 2

- as a dynamic language, JS will also allow us to
  - *modify these properties*
  - *delete any not required*
  - *or simply add a new one as necessary*
- this dynamic nature may change properties in a given object
- in traditional object-oriented programming languages
  - *this issue is often solved using inheritance*
- in JS
  - *we can use prototypes to implement inheritance*

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 I

- 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
  }
}
// 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 >= blockSt
            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...*

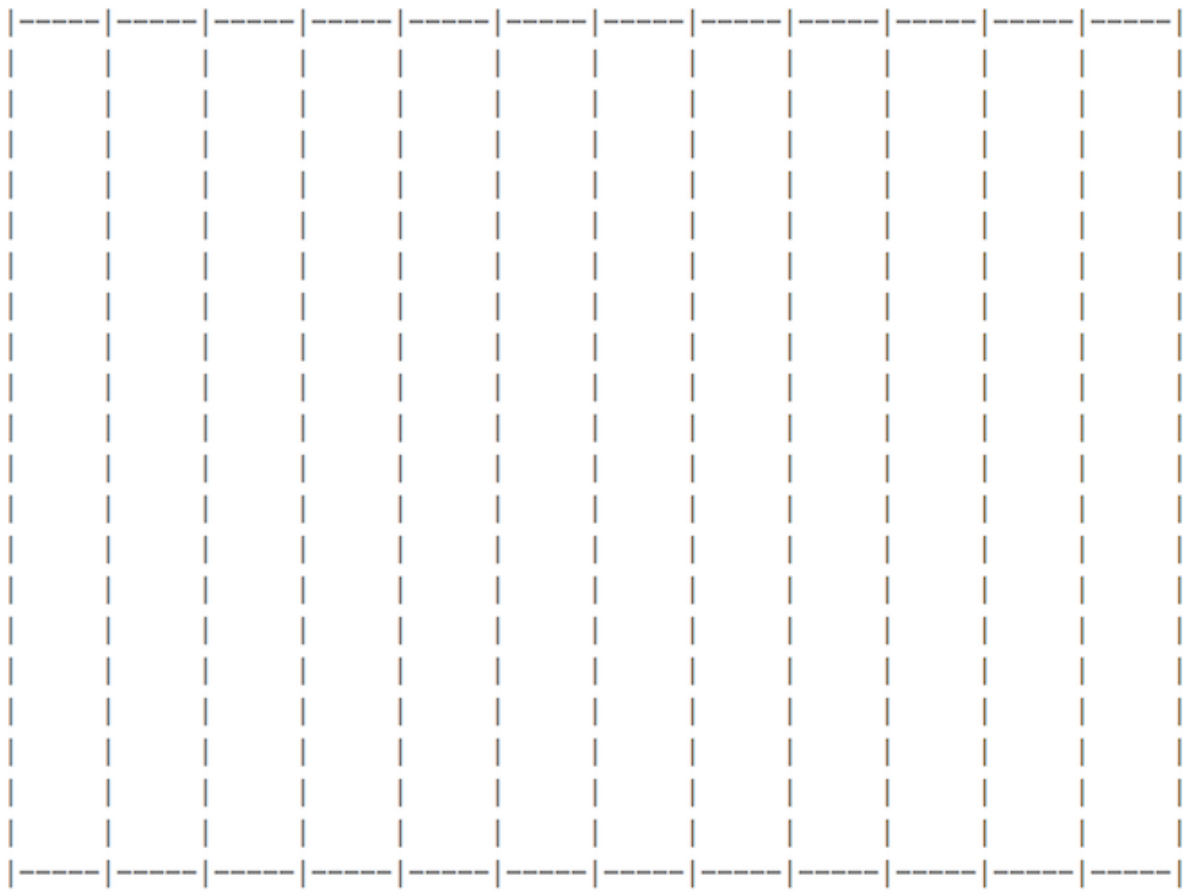
```
-x -y          +x -y
     \    |    /
      \   |   /
  ---- shape ----
      /   |   \
     /    |    \
-x +y          +x +y
```

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

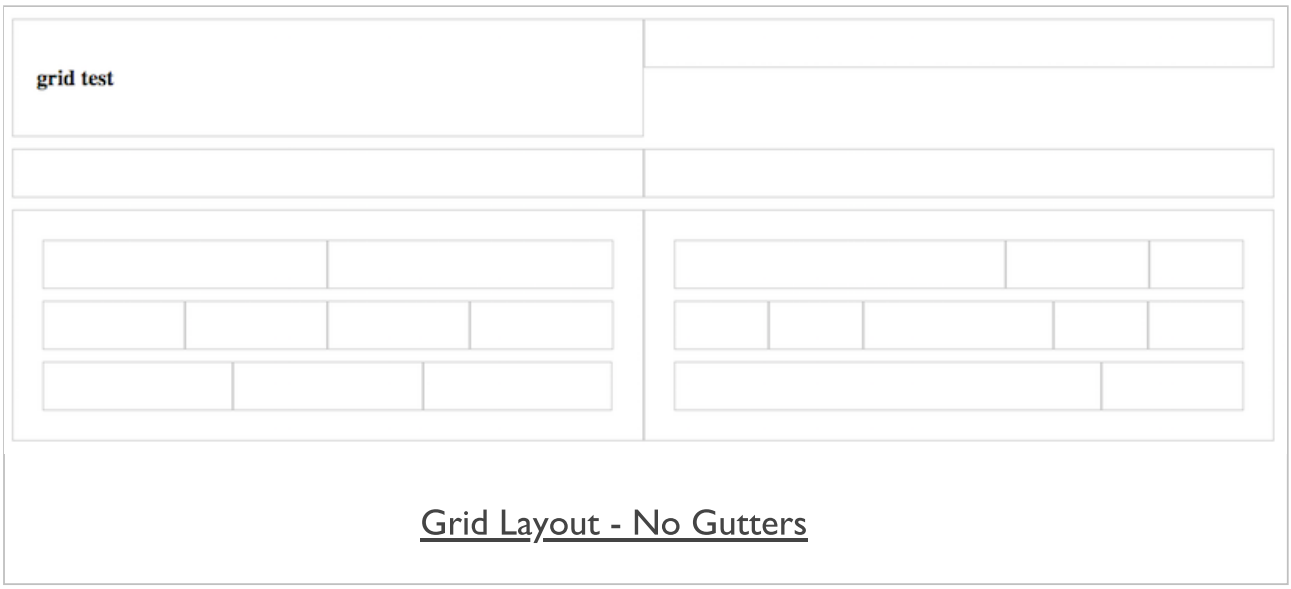
- due to the initial CSS of float left, each column is floated to the left
- columns are interpreted by subsequent elements in the hierarchy as non-existent
  - *initial placement will reflect this design*
- prevent this issue in layout, add the following CSS to grid stylesheet

```
.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