Coder Dojo 2012-03-03

HTML5 Shuffle Puzzle
CONTINUED

LAST week...

display and starting to shuffle.
But you covered a lot of topics...

github.com/willknott
github.com/coderdojo- I'm working on it

Last week's slides, code (puzzle2) and images are available

Huh? You created these variables...

currentPiece = null;
currentDropPiece = null;
Null means "No value".
mouse = {x:0,y:0};
This is a full one value array...and an object.

Think about it, what does a mouse position consist of...

pieces = [];
This is an empty array

Met arrays

http://www.w3schools.com/js/js_obj_array.asp

An array is a variable that can contain more than one value.

You can declare a complete array, or, push values on to an array.

Covered For loops

We are about to hit a for loop

http://www.w3schools.com/js/js_loop_for.asp

```
for (variable=startvalue;
      variable<=endvalue;
      variable=variable+increment)
     {
        do something
     }
```

And finished here....

```
            xPos += pieceWidth; //move a piece to the right

            if(xPos >= puzzleWidth){
                // if we fall off the edge of the image
                xPos = 0;
                //back to the left edge
                yPos += pieceHeight;
                // and move down one
            } // end of if
        } //end of for
        document.onmousedown = shufflePuzzle;
    }
```

// some of you might have commented out the mousedown to avoid errors

After dividing up our image, we need to shuffle them around.

Or cheat by just reversing the order (Yes this is code I want you to add)

```
function shuffleArray(o)
        {
            return o.reverse();
        }
```

http://www.w3schools.com/jsref/jsref_reverse.asp

Yes this does make a crap game.
We'll look at shuffling and other mathematics another time.

Lets make a function to shuffle the pieces

```
function shufflePuzzle(){
        pieces = shuffleArray(pieces);
        stage.clearRect(0,0,puzzleWidth,puzzleHeight);
        //Clear the canvas
        var i;
        var piece;
        var xPos = 0;
        var yPos = 0;

// This might be a good point to talk about
//passing between functions
```

Continued...

```
for(i = 0;i < pieces.length;i++){
    piece = pieces[i];
    piece.xPos = xPos;
    piece.yPos = yPos;
    stage.drawImage(img,
                    piece.sx, piece.sy, pieceWidth, pieceHeight,
                    xPos, yPos, pieceWidth, pieceHeight);
                   //Draw each puzzle piece
    stage.strokeRect(xPos, yPos,pieceWidth,pieceHeight);

    xPos += pieceWidth;
```

still continued

```
        if(xPos >= puzzleWidth){
            xPos = 0;
            yPos += pieceHeight;
          // Wrap around and go down one piece
          } //end of if statement
      } //end of for loop

      document.onmousedown = onPuzzleClick;
  } //end of shufflePuzzle
```

strokeRect?

Puts a border around it

onMouseDown?
MouseDown - pressing the mouse button
MouseUp - Releasing the button

There are more mouse and touch functions.
Look them up, you'd be surprised at what is missing...

Save and reload

Looks like nothing has happened...

So click on the image...

You have a puzzle.

Click again...

When we click

We want it to do more... back to coding

Jumping ahead slightly...

Figure out which piece was clicked on...

Skipping ahead as we haven't done the code to detect where the mouse click is...

Let's check if we've clicked on a piece

```
 function checkPieceClicked(){
        var i;
        var piece;
        for(i = 0;i < pieces.length;i++){
            piece = pieces[i];
            if(mouse.x < piece.xPos || mouse.x > (piece.xPos + pieceWidth)
|| mouse.y < piece.yPos || mouse.y > (piece.yPos + pieceHeight)){
                //PIECE NOT HIT
                // Why not return now?
            }
            else{
                return piece;
            }
        }
        return null;
    }
```

The what?

for(i = 0;i < pieces.length;i++)
we are looping through all the pieces in the puzzle
Arrays have properties, the length of the array is something you have access to.
We also have (because we forced it in there) the starting x and y coordinates locations of each piece

The big if

```
if(mouse.x < piece.xPos ||
    mouse.x > (piece.xPos + pieceWidth) ||
    mouse.y < piece.yPos ||
    mouse.y > (piece.yPos + pieceHeight))
```

First off || means "or". && Means "and" (Not true in all languages)

We have the mouse position, we have the top left position of each piece and their size.
So the if is... if the click is not on the square do...

Nothing

Do nothing. This piece was not clicked on, so look at the next one.
And
Since we are only looking at the position of each click, if a click outside the canvas is detected, it will loop through and return NULL.

If the function finds that we have clicked on a piece, it spits back the position of the piece.

```
else{
        return piece;
    }
```

BUT
We need to write the code to detect the mouse click.

This function handles when we click on a piece in the puzzle.
Or rather... where.

```
function onPuzzleClick(e){
        if(e.layerX || e.layerX == 0){
            mouse.x = e.layerX - canvas.offsetLeft;
            mouse.y = e.layerY - canvas.offsetTop;
        }
        else if(e.offsetX || e.offsetX == 0){
            mouse.x = e.offsetX - canvas.offsetLeft;
            mouse.y = e.offsetY - canvas.offsetTop;
        }
        currentPiece = checkPieceClicked();
        // call the detection function
```

Nasty isn't it

This is needed because I don't know which browser you are using.
Some use Layer, some use Offset.

You want the coordinates inside the entire puzzle
Which we'll cover next week.