Lesson 03.09 - Image Zoom, IEFE

- Image Zoom, IIFE, mousemover

Image Zoom

In this project, we will make an Image Zoomer:

- The UI consists of two divs and an image
- The user mouses over the image
- An empy framed box called "zoom-window" follows the mouse
- The other div shows the closeup from the "zoom-window"
- The function for this will be an IIFE

IIFE (Immediately Invoked Function Expression)

An IIFE (Immediately Invoked Function Expression):

- automatically runs as soon as it is declared.
- has no name: it's just function()
- is entirely wrapped in parentheses
- is called by another set of parentheses at the end
- runs immediately--but only once
- cannot be called again later, since it has no name

1. Declare the IIFE: everything is wrapped in parentheses, and then the function is called by a set of empty parentheses, right at the end:

```
/*   (function() {
        // do stuff;
    })()

*/

(function() {
```

2. Get the image:

```
const img = document.getElementById("image");
```

3. Have the image call a function on mousemove, that is, whenever the mouse is moved over the image, a function is called:

```
img.addEventListener("mousemove", moveView);
```

4. Get the "zoom-window" div. This is the div that follows the cursor as a little framed box:

   ```
   const zoomWindow = document.getElementById("zoom-window");
   ```

5. Have the zoomWindow also call the same function. Having the image and the framed zoom window box both call the function makes the zoom effect smoother:

   ```
   zoomWindow.addEventListener("mousemove", moveView);
   ```

6. Get the zoom image div for displaying the zoomed-in close-up of the image:

   ```
   const zoomImage = document.getElementById("zoom-image");
   ```

7. Declare variables for the offset width and the offset height. These are for defining the zommed-in portion of the image:

   ```
   let oW = zoomImage.offsetWidth / zoomWindow.offsetWidth;
   let oH = zoomImage.offsetHeight / zoomWindow.offsetHeight;
   ```

8. Set the zoomImage div background image to be the image we want to zoom in on. The image should be MUCH larger than the zoom image div that contains it, since the zoom image shows a detail.

   ```
   zoomImage.style.backgroundImage = "url(images/_chinese-day-bed-
   2.jpg)";
   ```

9. Set the background size (don't worry about what the math is doing). The backgroundSize property has two values: width and height:

   ```
   zoomImage.style.backgroundSize = (img.width * oW) + "px " +
   (img.height * oH) + "px";
   ```

position properties: pageX, pageY, pageXOffset, scrollX

- the event object has pageX and pageY properties pageX and pageY store the (x, y) coordinates of the cursor
- pageXOffset property, equal to the scrollX property returns how far the document has been scrolled from the top of the window.
- the offsetWidth property returns the viewable width of an element (in pixels) including padding, border and scrollbar, but not margin

getBoundingClientRect()

- the getBoundingClientRect() method is called on a DOM object and returns an object of 8 properties, pertaining to the object's size and position: **left, top, right, bottom, x, y, width, height**

Understanding (or not understanding) the math:

These various properties are ingeniously tapped to create the zoom effect. You are free to analyze the math on your own, but suffice it to say, there are many such "widget" projects where some or most of the math and logic may be hard to grasp. The thing to know is not the nuances of the math, but the fact that such widgets as the image zoomer exist, are Googleable, and can, for the most part be copy-pasted and then customized as needed--we don't usually need to mess with the core math.

10. Declare the moveView function that is called when the mouse if over the image and/or the zoom window that follows the cursor:

```
function moveView(event) {
```

11. Make the zoomWindow (the box that follows the mouse) appear:

```
zoomWindow.style.visibility = "visible";
```

12. Get the 8 properties of

```
bounds = img.getBoundingClientRect();
```

13. Declare a variable, x, set equal to a fairly complicated math expression:

```
let x = event.pageX - bounds.left - window.pageXOffset -
(zoomWindow.offsetWidth / 2);
```

14. If x is greater than the image's width, subtract the zoom window's offset width from x:

```
if (x > img.width - zoomWindow.offsetWidth) {
    x = img.width - zoomWindow.offsetWidth;
}
```

15. If x less than 0, set x to 0:

```
if (x < 0) {
    x = 0;
}
```

16. Set the left position of the zoom window to be x pixels:

```
zoomWindow.style.left = x + "px";
```

17. Repeat the "x calculations" for y:

```
let y = event.pageY - bounds.top - window.pageYOffset -
(zoomWindow.offsetHeight / 2);

if (y > img.height - zoomWindow.offsetHeight) {
    y = img.height - zoomWindow.offsetHeight;
}

if (y < 0) {
    y = 0;
}

zoomWindow.style.top = (y+60) + "px";
```

18. Set the zoom image background position as negative x and y values:

```
    zoomImage.style.backgroundPosition =
    "-" + (x * oW) + "px -" + (y * oH) + "px";
}

})();
```