6. Running Code Responsively with jQuery and JavaScript

In this week's explorations of using JavaScript with the jQuery library, we will work with function-based JavaScript code blocks that allow us to manage content display and respond to user-generated events.

In the first experiment, we face the real-world problem of too much content on a page. The "wall of text" syndrome is a well-known experience. Come upon a Web page containing too much text – especially without a strong visual hierarchy – and one's comprehension is immediately overwhelmed. Good content design should utilize pagination.

jQuery Methods in Use

Here we look at the use of the jQuery library's built-in **width()** method.

$('#myitem').width();	Returns the width in pixels of the content box of element with *id="myitem"*.
$(window).resize(function() { } **);**	The **resize()** event handler runs code in anonymous function code block when user resizes the browser window containing the page.
$(document).ready(**function() { }** **);**	The **ready()** event handler runs code in anonymous function code block when the browser completes loading the page.

Example

Our example applies a special class to the body element of the page for each of the valid Bootstrap breakpoint screen display widths in pixels. We do this by using conditional tests for the width of the current page in the browser window whenever the user resizes the window.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>jQuery Example: Running Code Responsively</title>
  <!-- load bootstrap css via cdn -->
  <link rel="stylesheet" href="https://netdna.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
  <!-- load jquery js via cdn -->
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
  <!-- load bootstrap js via cdn -->
  <script src="https://netdna.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
  <!-- custom styles to make visible responsive body classes -->
  <style>
    .body-xs {
      background-color:orange;
    }
    .body-sm {
      background-color:pink;
    }
    .body-md {
      background-color:yellow;
    }
    .body-lg {
      background-color:silver;
```

```
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="row">
      <div class="col-xs-12">
        <h2>Running code responsively</h2>
      </div>
    </div>
  </div>
  <!-- custom javascript using jquery to handle dynamic search filtering -->
  <script>
    // create event handler for window resizing
    $(window).resize(
      function() {
        // get document width
        var w = $(document).width();

        // remove all classes from body
        $("body").removeClass();

        // add class to body for lg breakpoint = 1200px
        if ( w >= 1200 ) {
          $("body").addClass("body-lg");
        }

        // add class to body for md breakpoint = 992px
        if ( w >= 992 ) {
          $("body").addClass("body-md");
        }

        // add class to body for sm breakpoint = 992px
        if ( w >= 768 ) {
          $("body").addClass("body-sm");
        }

        // add class to body for xs breakpoint = 992px
        if ( w >= 480 ) {
          $("body").addClass("body-xs");
        }
      }
    );
  </script>
</body>
</html>
```

You can try switching the **$(window).resize()** event handler for the **$(document).ready()** handler to observe the difference in behavior for the code.