

Practice Exercise

This document provides a list of exercises to be practiced by learners. Please raise feedback in Talent Next, should you have any queries.

Skill	HTML5 & CSS3
Proficiency	S1
Document Type	Lab Practice Exercises
Author	L & D
Current Version	2.0
Current Version Date	26-Apr-2022
Status	Active

Document Control

Version	Change Date	Change Description	Changed By
1.0	9-Mar-2021	New Version created	Pradeep Chinhole
2.0	26-Apr-2022	Added new Problem Statement for HTML5 Semantic Markup Tags, Forms & CSS3 selectors and media queries.	Pradeep Chinhole

Contents

Practice Exercise.....	1
Document Control	2
Problem Statement 1: Creating Blog Using HTML5 Semantic Markup and CSS	4
Problem Statement 2: Showing Progress toward a Goal with the <meter> Element	5
Problem Statement 3: Defining an FAQ with a Description List.....	6
Problem Statement 4: Creating User-Friendly Web Forms.	7
Problem Statement 5: Styling the Web Form	9
Problem Statement 6: Jumping to the First Field with Autofocus	10
Problem Statement 7: Providing Hints with Placeholder Text	11
Problem Statement 8: Validating User Input without JavaScript	12
Problem Statement 9: Validation with Regular Expressions	13
Problem Statement 10: Styling the Fields	14
Problem Statement 11: Styling Tables with Pseudo classes	15
Problem Statement 12: Striping Rows with :nth-of-type	17
Problem Statement 13: Bolding the Last Row with :last-child	18
Problem Statement 14: Making Links Printable with :after and content.....	19
Problem Statement 15: Building Mobile Interfaces with Media Queries	20

Note: Every Problem Statement starts in a new page

Problem Statement 1: Creating Blog Using HTML5 Semantic Markup and CSS

Introduction

A client AwesomeCo approached Mphasis to develop the blog for their website using HTML5 and CSS3. You are hired as Mphasis UI developer and asked to develop the blog as shown below using HTML5 semantic markup tags such as <header>, <nav>, <section>, <aside>, <article>, , <a> and <footer> and appropriate CSS.

AwesomeCo Blog!

[Latest Posts](#) [Archives](#) [Contributors](#) [Contact Us](#)

How Many Should We Put You Down For?

Posted by Brian on October 1st, 2013 at 2:39PM

The first big rule in sales is that if the person leaves empty-handed, they're likely not going to come back. That's why you have to be somewhat aggressive when you're working with a customer, but you have to make sure you don't overdo it and scare them away.

One way you can keep a conversation going is to avoid asking questions that have yes or no answers. For example, if you're selling a service plan, don't ever ask "Are you interested in our 3 or 5 year service plan?" Instead, ask "Are you interested in the 3 year service plan or the 5 year plan, which is a better value?" At first glance, they appear to be asking the same thing, and while a customer can still opt out, it's harder for them to opt out of the second question because they have to say more than just "no."

"Never give someone a chance to say no when selling your product."

[25 Comments ...](#)

Copyright © 2013 AwesomeCo.

[Home](#) [About](#) [Terms of Service](#) [Privacy](#)

Archives

- [October 2013](#)
- [September 2013](#)
- [August 2013](#)
- [July 2013](#)
- [June 2013](#)
- [May 2013](#)
- [April 2013](#)
- [March 2013](#)
- [February 2013](#)
- [January 2013](#)
- [More](#)

Note :

- Write a HTML code in **index.html**
- Create external CSS file **styles.css** to style the blog using CSS and link to index.html

Problem Statement 2: Showing Progress toward a Goal with the <meter> Element

1. Create a new HTML5 document **html5_meter/index.html** to represent our meter to demonstrate and how it works with below code, hard-coding 2500.00 as our current value to demonstrate and how it works.

Our Fundraising Goal



Help us reach our goal of \$5000!

Problem Statement 3: Defining an FAQ with a Description List

Develop a Web page to display Description List as below.

AwesomeCo FAQ

What is it that AwesomeCo actually does?

AwesomeCo creates innovative solutions for business that leverage growth and promote synergy, resulting in a better life for the global community.

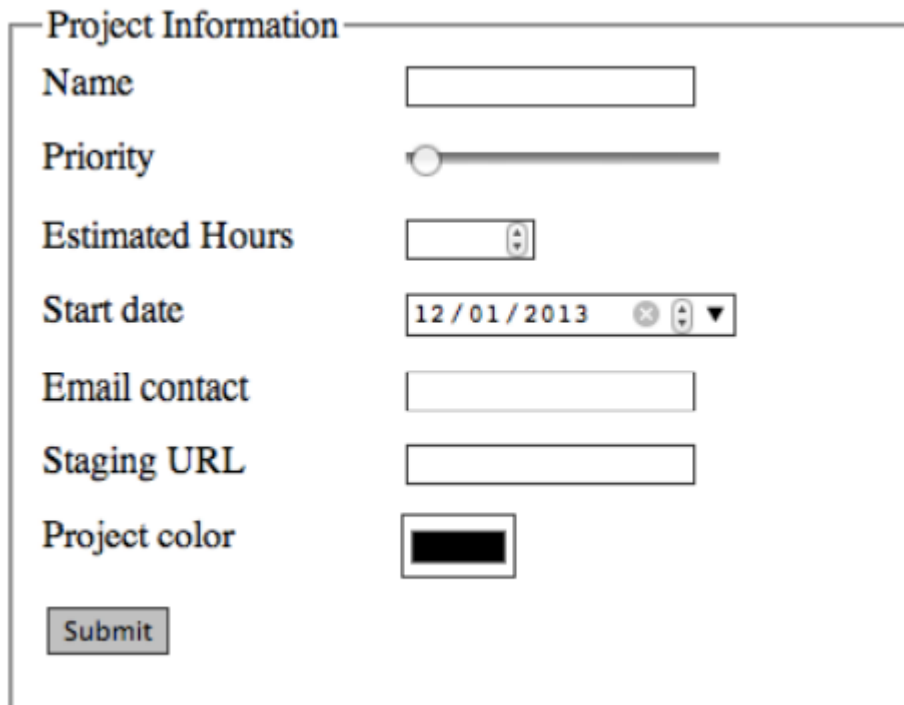
1. Create a new HTML5 document **html5_descriptionlist_faq/index.html** to use the <dl> tag to define the FAQ itself, <dt> tag for each question and <dd> tag for answer to each question as below.

html5_descriptionlist_faq/index.html

```
<article>
  <h1>AwesomeCo FAQ</h1>
  <dl>
    <dt>What is it that AwesomeCo actually does?</dt>
    <dd>
      <p>
        AwesomeCo creates innovative solutions for business that
        leverage growth and promote synergy, resulting in a better
        life for the global community.
      </p>
    </dd>
  </dl>
</article>
```

Problem Statement 4: Creating User-Friendly Web Forms.

AwesomeCo is working on creating a new project-management web application to make it easier for developers and managers to keep up with the progress of the many projects they have going on. Each project has a name, a contact email address, and a staging URL so managers can preview the website as it's being built. There are also fields for the start date, priority, and estimated number of hours the project should take to complete. Finally, the development manager would like to give each project a color so he can quickly identify projects when he looks at reports. Create quick project preferences page using the new HTML5 fields as shown below.



1. Create a new HTML5 page `html_forms/index.html` with a basic HTML form that does a POST request with the assumption that at some point there'll be a page that processes this form as below.

`html5_forms/index.html`

```
<form method="post" action="/projects/1">

  <fieldset id="personal_information">
    <legend>Project Information</legend>
    <ol>
      <li>
        <label for="name">Name</label>
        <input type="text" name="name" id="name">
      </li>
      <li>
        <input type="submit" value="Submit">
      </li>
    </ol>
  </fieldset>

</form>
```

2. Create a slider using range to accept the priority as shown below.

html5_forms/index.html

```
<label for="priority">Priority</label>
<input type="range" min="0" max="10"
      name="priority" value="0" id="priority">
```

3. Create a number field to accept the estimated hours as show below.

html5_forms/Index.html

```
<label for="estimated_hours">Estimated Hours</label>
<input type="number" name="estimated_hours"
      min="0" max="1000"
      id="estimated_hours">
```

4. Create a date field to record the date of the project as shown below.

html5_forms/index.html

```
<label for="start_date">Start date</label>
<input type="date" name="start_date" id="start_date"
      value="2013-12-01">
```

5. Create an email field to accept the email as shown below.

html5_forms/index.html

```
<label for="email">Email contact</label>
<input type="email" name="email" id="email">
```

6. Create an URL field to accept project's staging URL as shown below.

html5_forms/index.html

```
<label for="url">Staging URL</label>
<input type="url" name="url" id="url">
```

7. Create color field to accept project's color code as shown below.

html5_forms/index.html

```
<label for="project_color">Project color</label>
<input type="color" name="project_color" id="project_color">
```


Problem Statement 5: Styling the Web Form

Style the Web Form created in Problem Statement 4 using CSS.

1. Create a new file called **stylesheets/style.css** and link it in the **<head>** section of the form's page in index.html as below

html5_forms/index.html

```
<link rel="stylesheet" href="stylesheets/style.css">
```

2. Remove the numbering, margins, and padding from the list using below CSS code.

html5_forms/stylesheets/style.css

```
ol{
    list-style: none;
    margin: 0;
    padding :0;
}

ol li{
    clear: both;
    margin: 0 0 10px 0;
    padding: 0;
}
```

3. Align the labels and the input fields and add a little styling to the input fields.

html5_forms/stylesheets/style.css

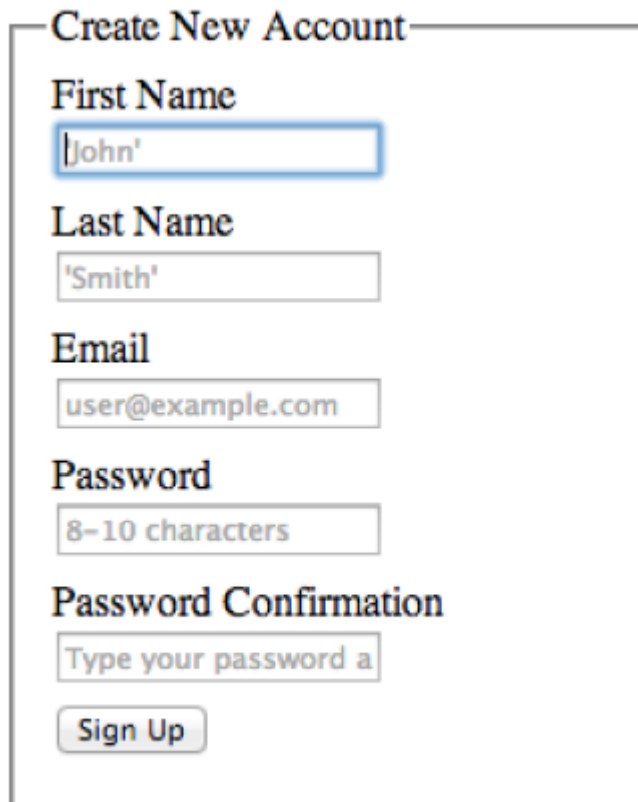
```
label{
    float: left;
    width: 150px;
}

input{ border: 1px solid #333; }

input:focus{ background-color: #ffe; }
```

Problem Statement 6: Jumping to the First Field with Autofocus

Create a HTML Page to demonstrate Jumping to the First Field with Autofocus as shown below.



The image shows a 'Create New Account' form. It contains the following fields and labels:

- Create New Account** (Section Header)
- First Name** (Label) with a text input field containing 'John'.
- Last Name** (Label) with a text input field containing 'Smith'.
- Email** (Label) with a text input field containing 'user@example.com'.
- Password** (Label) with a text input field containing '8-10 characters'.
- Password Confirmation** (Label) with a text input field containing 'Type your password a'.
- Sign Up** (Button)

1. Create a new file called **html5_forms/autofocus/index.html** to demonstrate Jumping to the First Field with autofocus.

html5_forms/autofocus/index.html

```
<label for="name">Name</label>
<input type="text" name="name" autofocus id="name">
```

Problem Statement 7: Providing Hints with Placeholder Text

AwesomeCo's support site requires users to sign up for an account as below and one of the biggest problems with the sign-ups is that users keep trying to use insecure passwords. Let's use placeholder text (see the following figure) to give users a little guidance on our password requirements. For consistency's sake, we'll add placeholder text to the other fields too.

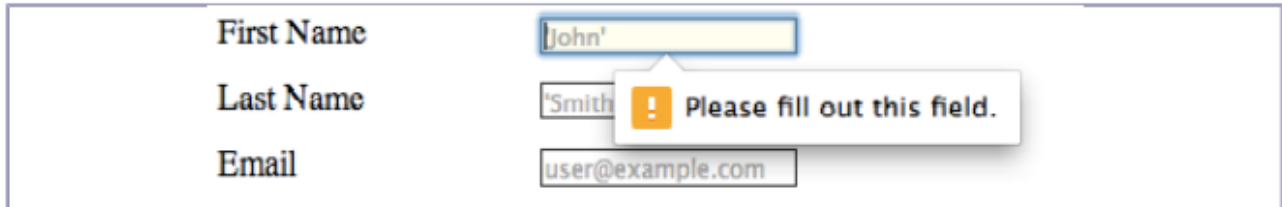
2. Create `html5_placeholder/index.html` for sign up using below source code.

```
html5_placeholder/index.html
<form id="create_account" action="/signup" method="post">
  <fieldset id="signup">
    <legend>Create New Account</legend>
    <ol>
      <li>
        <label for="first_name">First Name</label>
        <input id="first_name" type="text"
          autofocus
          name="first_name" placeholder="'John'" />
      </li>
      <li>
        <label for="last_name">Last Name</label>
        <input id="last_name" type="text"
          name="last_name" placeholder="'Smith'" />
      </li>
      <li>
        <label for="email">Email</label>
        <input id="email" type="email"
          name="email" placeholder="user@example.com" />
      </li>
      <li>
        <label for="password">Password</label>
        <input id="password" type="password" name="password" value=""
          autocomplete="off" placeholder="8-10 characters" />
      </li>
      <li>
        <label for="password_confirmation">Password Confirmation</label>
        <input id="password_confirmation" type="password"
          name="password_confirmation" value=""
          autocomplete="off" placeholder="Type your password again" />
      </li>
      <li><input type="submit" value="Sign Up"></li>
    </ol>
  </fieldset>
</form>
```

Problem Statement 8: Validating User Input without JavaScript

The AwesomeCo support site's signup page is a great place for us to test out these new attributes. Users need to provide their first name, last name, and email address. Ensure that the first name, last name, and email fields are validated as per below conditions.

- First Name is required.
- Last Name is required.
- Email is required and It should be valid email address.



1. Create a HTML5 form **html_validation/index.html** where user fills in a form field by adding the required attribute to the element just like we do with the attribute. So, we can add the required attribute to the First Name, Last Name, and Email fields on the original signup form.

html5_validation/index.html

```
<li>
  <label for="first_name">First Name</label>

  <input id="first_name" type="text"
    autofocus="true"
    required
    name="first_name" placeholder="'John'">
</li>
<li>
  <label for="last_name">Last Name</label>

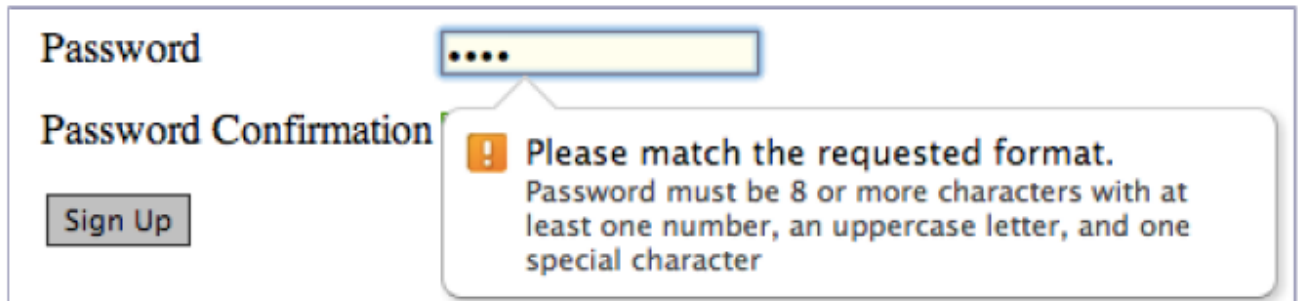
  <input id="last_name" type="text"
    required
    name="last_name" placeholder="'Smith'">
</li>
<li>
  <label for="email">Email</label>

  <input id="email" type="email"
    required
    name="email" placeholder="user@example.com">
</li>
```

Problem Statement 9: Validation with Regular Expressions

The pattern attribute lets us specify a regular expression against which we can validate the user data. The browser already knows how to validate email addresses and URLs, but we have specific rules for the Password field. Validate the password as per below condition using pattern attribute

- Password must be 8 or more characters.
- It should contain at least one number, an uppercase letter and one special character.
- Password and confirm Password must be same.



The screenshot shows a web form with two input fields. The first field is labeled 'Password' and contains four dots. The second field is labeled 'Password Confirmation'. Below the 'Password' field, there is a 'Sign Up' button. A validation error message is displayed next to the 'Password' field, stating: 'Please match the requested format. Password must be 8 or more characters with at least one number, an uppercase letter, and one special character'.

1. Create a HTML5 form **html_validation/index.html** to validate the password using pattern attribute and regular expression as shown below.

html5_validation/index.html

```
<li>
  <label for="password">Password</label>
  <input id="password" type="password" name="password" value=""
    autocomplete="off" placeholder="8-10 characters"
    pattern="^(?=.*{8,})(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&*])(?=.*[0-9]).*$"
    title="Password must be 8 or more characters with at
      least one number, an uppercase letter, and one special character"
  />
</li>
```

2. Use the same pattern for Password Confirmation field as below

html5_validation/index.html


```
<li>
  <label for="password_confirmation">Password Confirmation</label>
  <input id="password_confirmation" type="password"
    name="password_confirmation" value=""
    autocomplete="off" placeholder="Type your password again"
    pattern="^(?=.*{8,})(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&*])(?=.*[0-9]).*$"
    title="Password confirmation must be 8 or more characters with at
      least one number, an uppercase letter, and one special character"
  />
</li>
```

Problem Statement 10: Styling the Fields

Use just a little CSS to provide instant feedback to users using pseudo classes **:valid** and **:invalid** in some style declarations for the input fields created in Problem Statement 8 and 9 as shown below.

First Name	<input type="text" value="John"/>
Last Name	<input type="text" value="'Smith'"/>
Email	<input type="text" value="user@example.com"/>
Password	<input type="text" value="8-10 characters"/>
Password Confirmation	<input type="text" value="Type your password again"/>
<input type="button" value="Signup"/>	

First Name	<input type="text" value="John"/>
Last Name	<input type="text" value="Smith"/>
Email	<input type="text" value="Smith"/>
Password	
Password Confirmation	
<input type="button" value="Signup"/>	

 Please include an '@' in the email address. 'Smith' is missing an '@'.

1. Create a CSS **html_validation/stylesheets/style.css** to provide instant feedback to users using pseudo classes **:valid** and **:invalid** and link it to the index.html created in Problem Statement 9.

html5_validation/stylesheets/style.css

```
input[required]:invalid, input[pattern]:invalid{
    border-color: #A5340B;
}

input[required]:valid, input[pattern]:valid{
    border-color: #0B9900;
}
```

Problem Statement 11: Styling Tables with Pseudo classes

AwesomeCo uses a third-party billing and invoicing system for products it ships. You see, one of AwesomeCo's biggest markets is conference swag, such as pens, cups, shirts, and anything else you can slap your logo on. You've been asked to make the invoice more readable. Right now, the developers are producing a standard HTML table as shown below

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

1. Create `css3_advanced_selectors/index.html` to create a table as shown below.

`css3_advanced_selectors/index.html`

```
<table >
  <tr>
    <th>Item</th>
    <th>Price</th>
    <th>Quantity</th>
    <th>Total</th>
  </tr>
  <tr>
    <td>Coffee mug</td>
    <td>$10.00</td>
    <td>5</td>
    <td>$50.00</td>
  </tr>
  <tr>
    <td>Polo shirt</td>
    <td>$20.00</td>
    <td>5</td>
    <td>$100.00</td>
  </tr>
  <tr>
    <td>Red stapler</td>
    <td>$9.00</td>
    <td>4</td>
    <td>$36.00</td>
  </tr>
```



```
<tr>
  <td colspan="3">Subtotal</td>
  <td>$186.00</td>
</tr>
<tr>
  <td colspan="3">Shipping</td>
  <td>$12.00</td>
</tr>
<tr>
  <td colspan="3">Total Due</td>
  <td>$198.00</td>
</tr>
</table>
```

2. Create a new file called `stylesheets/style.css` and link it up as shown below.

`css3_advanced_selectors/index.html`

```
<link rel="stylesheet" href="stylesheets/style.css">
```

3. Create a new file called `stylesheets/style.css` as shown below

`css3_advanced_selectors/stylesheets/style.css`

```
table{
  border-collapse: collapse;
  width: 600px;
}
```

```
th, td{ border: none; }
```

```
th{
  background-color: #000;
  color: #fff;
}
```

4. Apply that style, and the table looks like this

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

Problem Statement 12: Striping Rows with :nth-of-type

Stripe every other row of the table created in Problem Statement 12 using :nth-of-type pseudo classes as show below.

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

1. update a file `stylesheets/style.css` created in Problem Statement 12 with below code

```
css3_advanced_selectors/stylesheets/style.css
```

```
tr:nth-of-type(even){  
    background-color: #F3F3F3;  
}  
tr:nth-of-type(odd) {  
    background-color:#ddd;  
}
```

Problem Statement 13: Bolding the Last Row with :last-child

The invoice is looking for pretty good right now, but one of the managers would like the bottom row of the table to be bolder than the other rows so it stands out as shown below.

Item	Price	Quantity	Total
Coffee mug	\$10.00	5	\$50.00
Polo shirt	\$20.00	5	\$100.00
Red stapler	\$9.00	4	\$36.00
Subtotal			\$186.00
Shipping			\$12.00
Total Due			\$198.00

1. Update a file **stylesheets/style.css** created in Problem Statement 13 with below code.

css3_advanced_selectors/stylesheets/style.css

```
tr:last-child{
    font-weight: bolder;
}

td:last-child{
    font-weight: bolder;
}

tr:last-child td:last-child{
    font-size:24px;
}
```

Problem Statement 14: Making Links Printable with :after and content

AwesomeCo is working up a new page for its forms and policies, and one of the members of the redesign committee insists on printing out a copy of the site each time the committee meets. He wants to know exactly where all the links go on the page so that he can determine whether they need to be moved. With just a little bit of CSS as shown below.

Forms and Policies

- [Travel Authorization Form \(travel/index.html\)](travel/index.html)
- [Travel Reimbursement Form \(travel/expenses.html\)](travel/expenses.html)
- [Travel Guidelines \(travel/guidelines.html\)](travel/guidelines.html)

1. Create `css3_print_links/index.html` with links as shown below.

`css3_print_links/index.html`

```
<ul>
  <li>
    <a href="travel/index.html">Travel Authorization Form</a>
  </li>
  <li>
    <a href="travel/expenses.html">Travel Reimbursement Form</a>
  </li>
  <li>
    <a href="travel/guidelines.html">Travel Guidelines</a>
  </li>
</ul>
```

2. Create `css3_print_links/stylesheets/print.css` with below code.

`css3_print_links/stylesheets/print.css`

```
a:after {
  content: " (" attr(href) ) ";
}
```

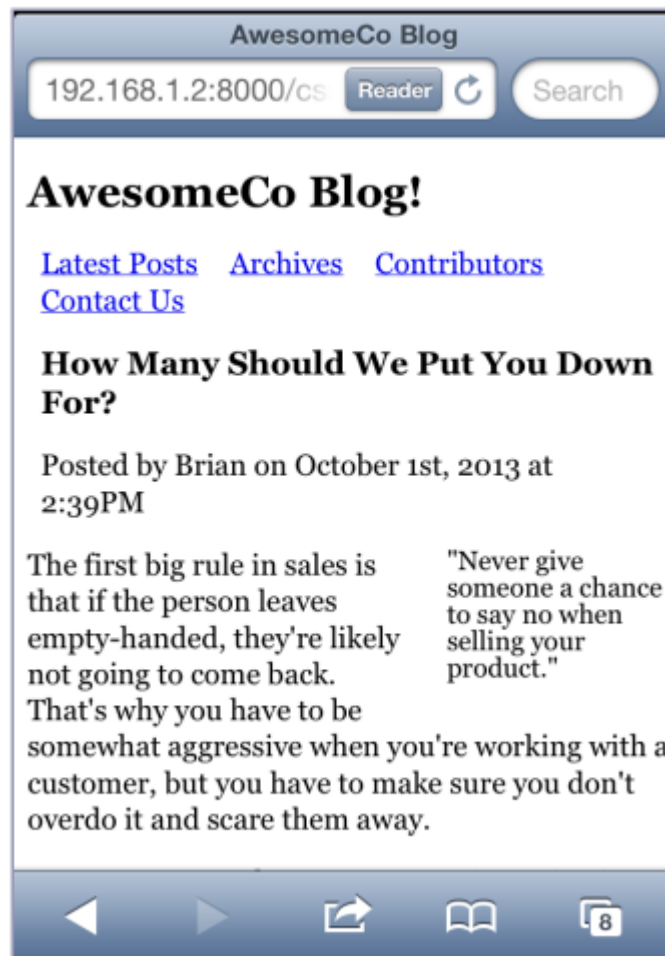
3. Link `print.css` in `index.html` as shown below.

`css3_print_links/index.html`

```
<link rel="stylesheet" href="print.css" type="text/css" media="print">
```

Problem Statement 15: Building Mobile Interfaces with Media Queries

AwesomeCo executive staff has finally gotten tired of hearing complaints from customers and employees about how web pages look like garbage on smartphones. The marketing director would love to see a mobile-ready version of the blog template we built in as shown below.



1. Create `css3_mediaquery/stylesheets/style.css` with below code and link it to the `index.html` created in Problem Statement 1 for blog.

`css3_mediaquery/stylesheets/style.css`

```
@media only screen and (max-device-width: 480px) {  
  body{ width:480px; }  
  nav, section, header, footer{ margin: 0 10px 0 10px; }  
  
  #sidebar, #posts{  
    float: none;  
    width: 100%;  
  }  
}
```

2. Create **stylesheets/mobile.css** with above code and link in html as below.

```
<link rel="stylesheet" type="text/css"  
      href="stylesheets/mobile.css" media="only screen and (max-device-width: 480px)">
```

3. With this our blog immediately becomes more readable on tiny screens, although the viewport is still zoomed way out. We can fix that by adding this tag right below the web page's <title> tag

css3_mediaquery/index.html

```
<meta name="viewport"  
      content="width=device-width, initial-scale=1, maximum-scale=1">
```