HTML5 and CSS3 Complete

Second Edition

Unit I

Implementing Responsive Design

Objectives

Unit I

- Assess responsive design
- Construct a multipart media query
- Test layouts with an emulator
- Add a column with a media query
- Create a widescreen layout
- Create responsive navigation

Objectives (continued)

Unit I

- Implement adaptive content
- Use progressive enhancement

Assess Responsive Design

Unit I

- Web pages can be viewed at a range of screen sizes
- Responsive design: allows a web developer to specify different CSS rules for some/all elements depending on width of screen

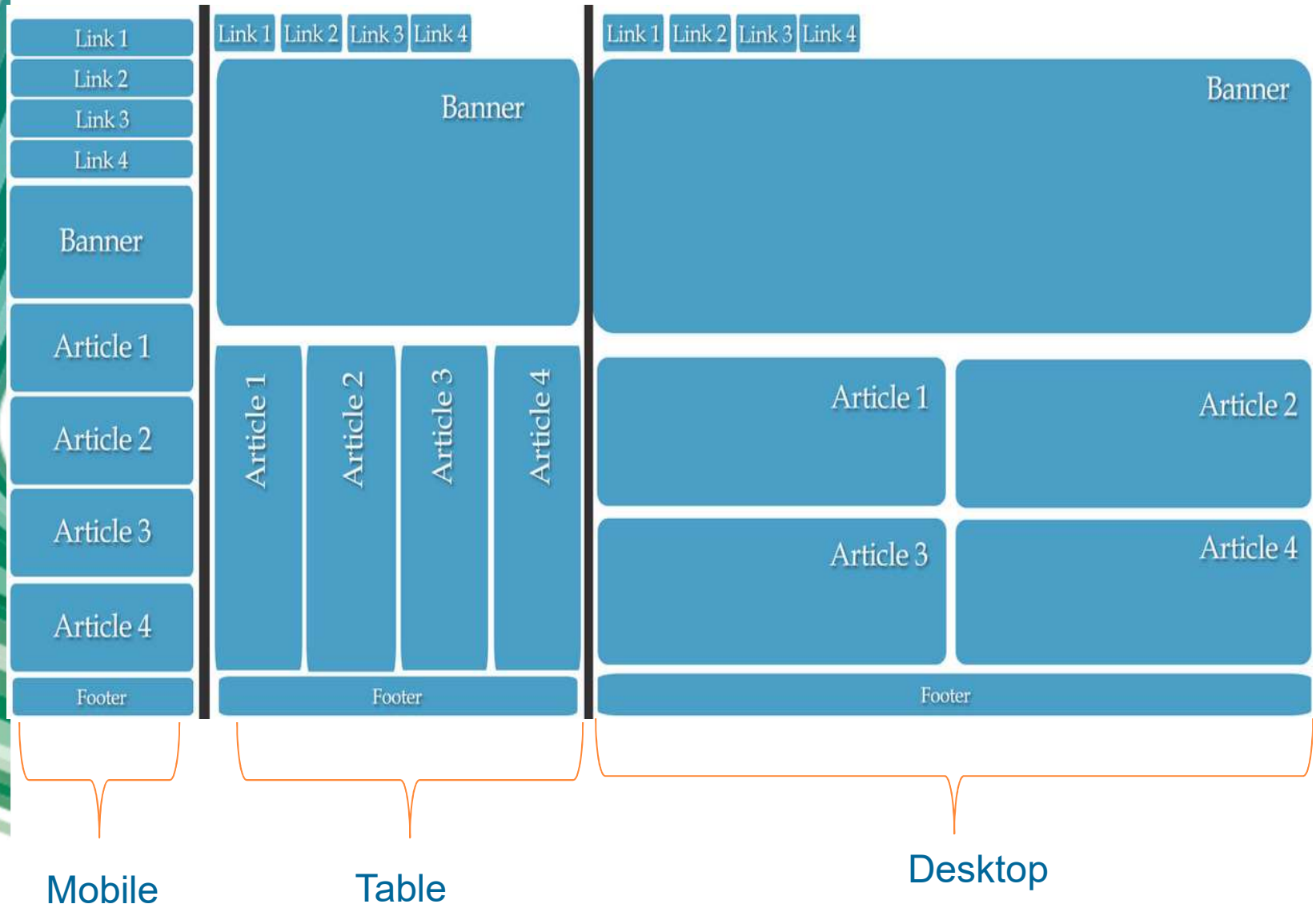
Mobile

Tablet

Desktop

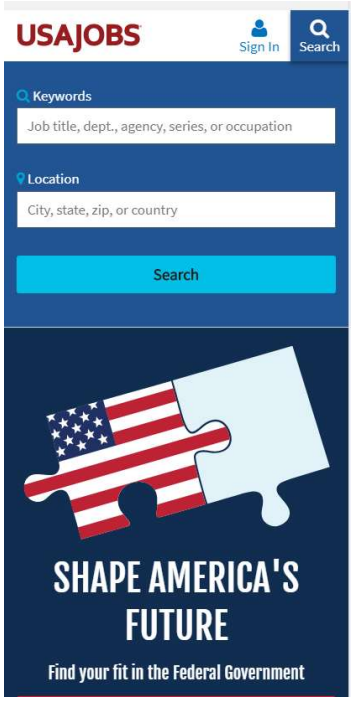
Example of Screen Sizes

Unit I

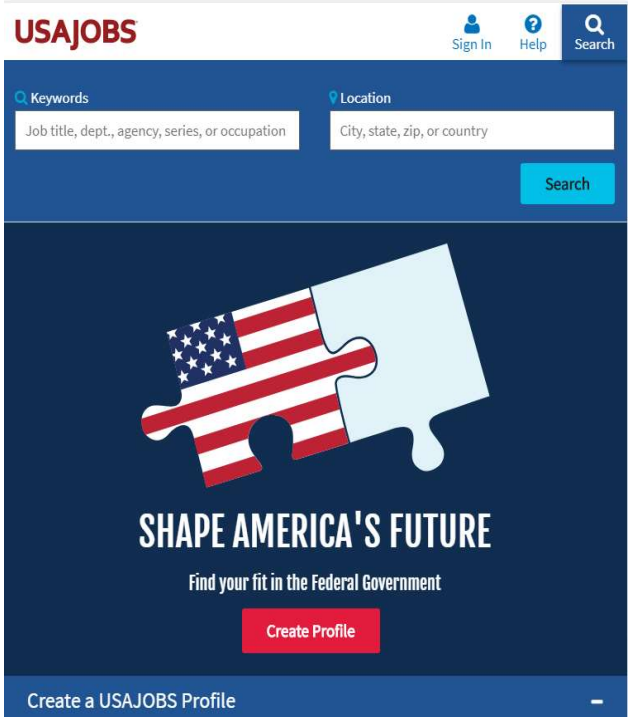


Screen sizes using emulator

Unit I

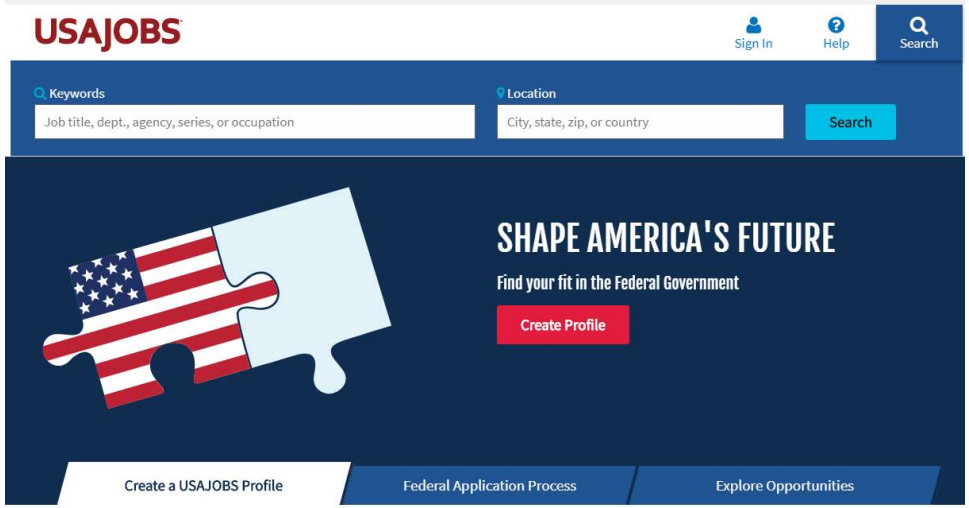


mobile



Tablet

Desktop



Create a USAJOBS Profile

Assess Responsive Design (continued)

Unit I

- Sizing with percentages
  - Elements sized with percentages maintain their sizing relative to each other while preserving their layout
  - Example: two images side by side, each with width of 45%, with remaining 10% for margin, padding, and border

Assess Responsive Design (continued)

Unit I

- Identifying breakpoints
  - Start with layout for smallest or largest screen size you want to support
  - View layout at different widths
  - Breakpoint: width at which the layout no longer looks good, or at which you decide to move elements, or add or remove content
- 320px iPhone, 768x iPad, 1024x laptop

Google Chrome Emulator

Assess Responsive Design (continued)

Unit I

- Creating multipart media queries
  - Using first breakpoint, use `@media` keyword to create media query
  - Media features: conditions media must satisfy for rules in query to be applied
  - Media query includes the `screen` media type followed by one or more media features
  - `min-width` and `max-width` media features most common in responsive design

Assess Responsive Design (continued)

Unit I

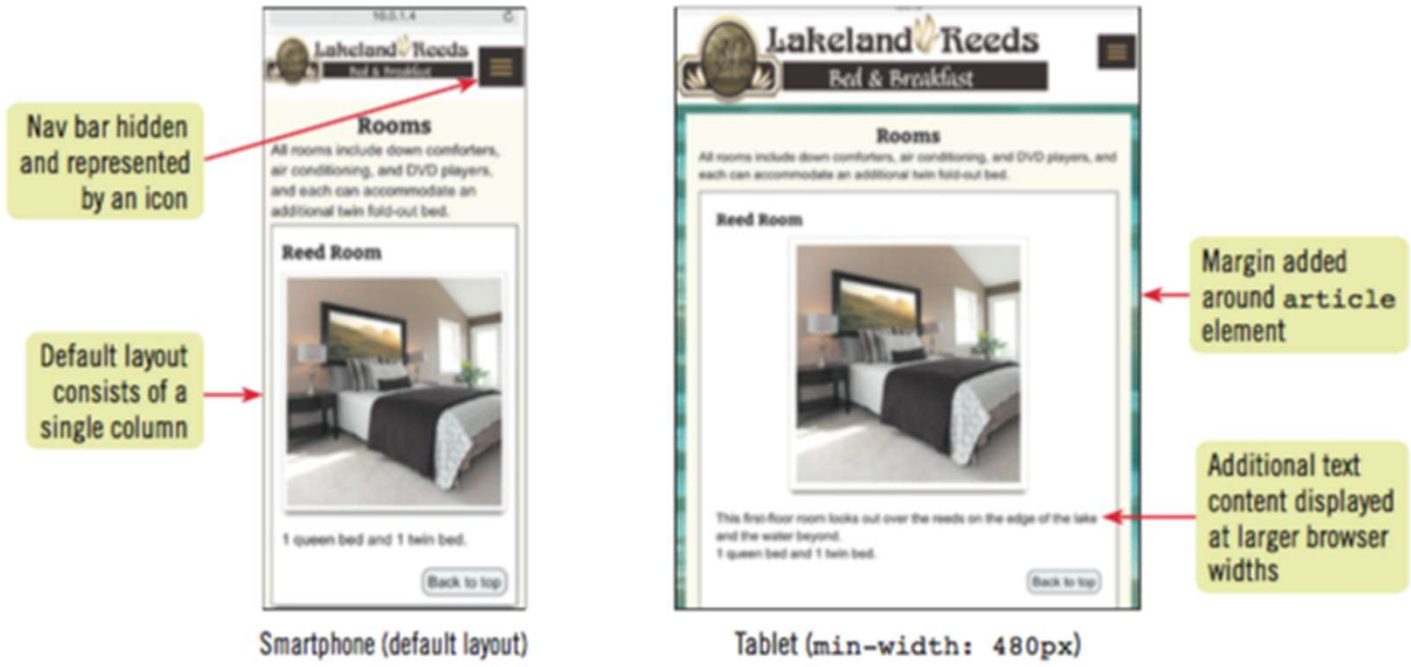
- Creating multipart media queries (continued)
  - Example:

```
@media screen and (min-width:
800px) {
    style rules
}
```

Assess Responsive Design (continued)

Unit I

Figure I-1: Web page displayed on different devices using different media queries



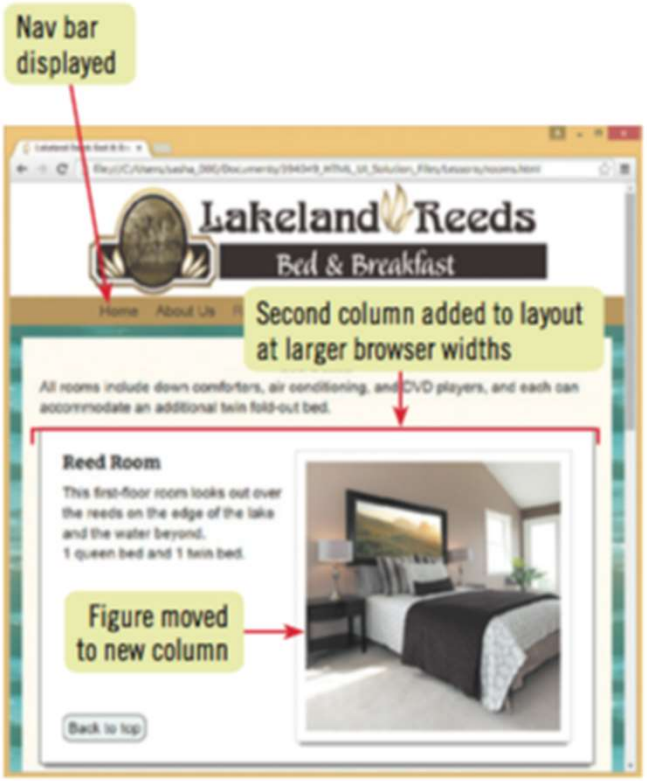
Smartphone (default layout)

Tablet (`min-width: 480px`)

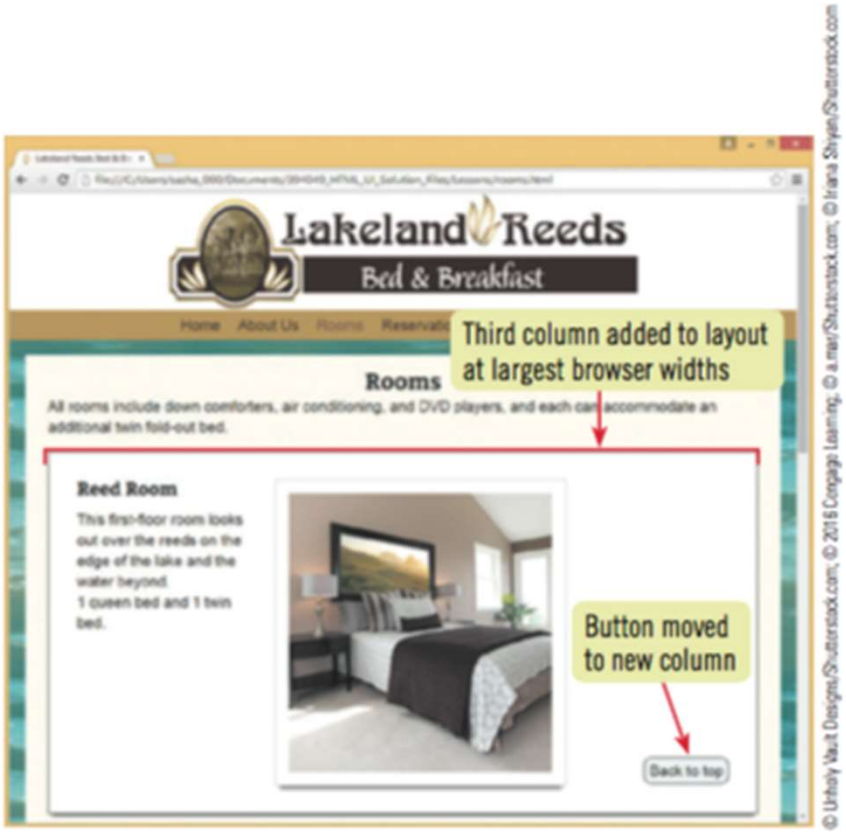
Assess Responsive Design (continued)

Unit I

Figure I-1: Web page displayed on different devices using different media queries (cont'd)



Standard desktop (`min-width: 880px`)



Large desktop (`min-width: 1060px`)

Construct a Multipart Media Query

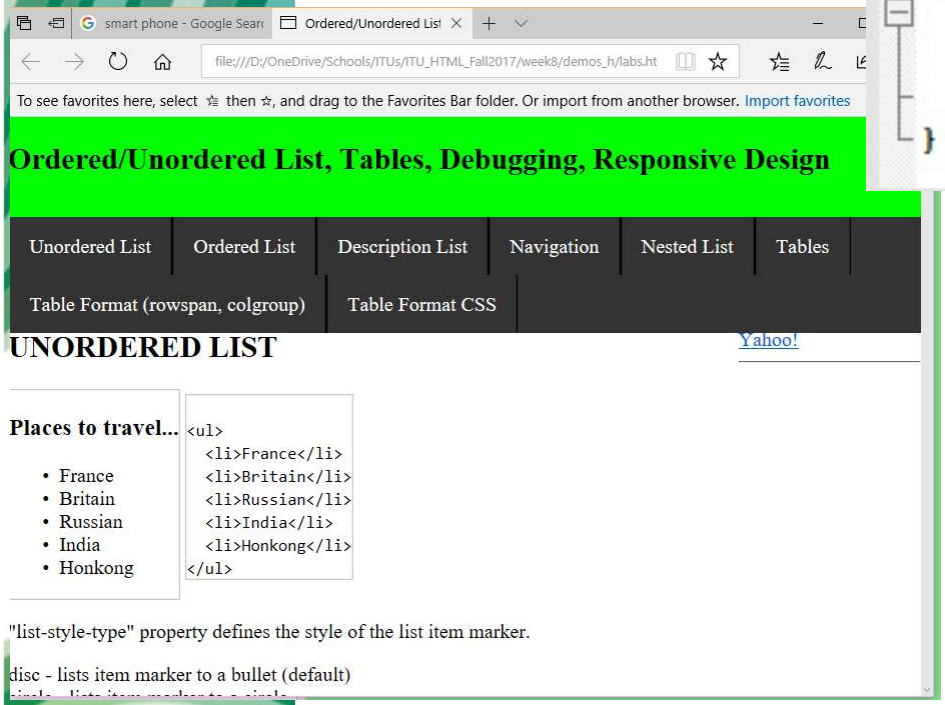
Unit I

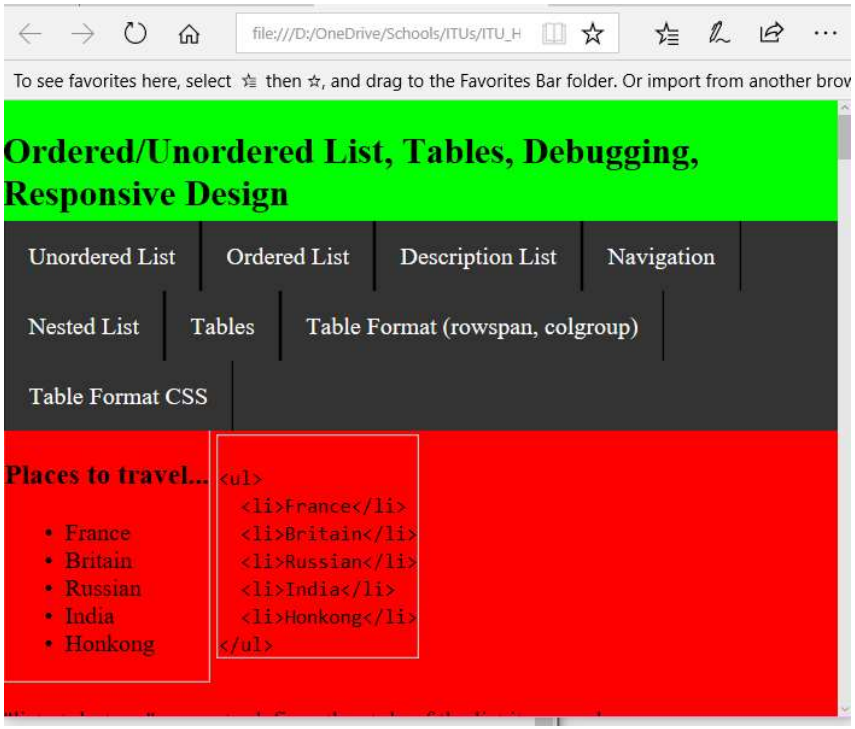
- Implementing responsive design:
  - 1. Create a default layout for smallest or largest browser width to support
  - 2. Create a multipart media query
    - `screen` media type
    - media feature (usually `min-width` or `max-width`)

Demo – use labs.htm, labs.css and add the code below.

Unit I

```
@media only screen and (max-width: 600px) {
    body {
        background-color: red;
    }
}
```

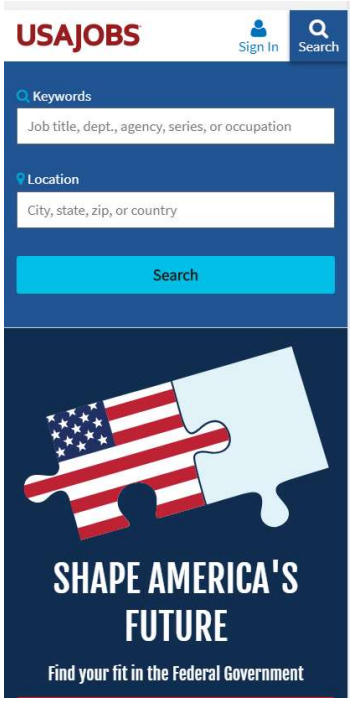




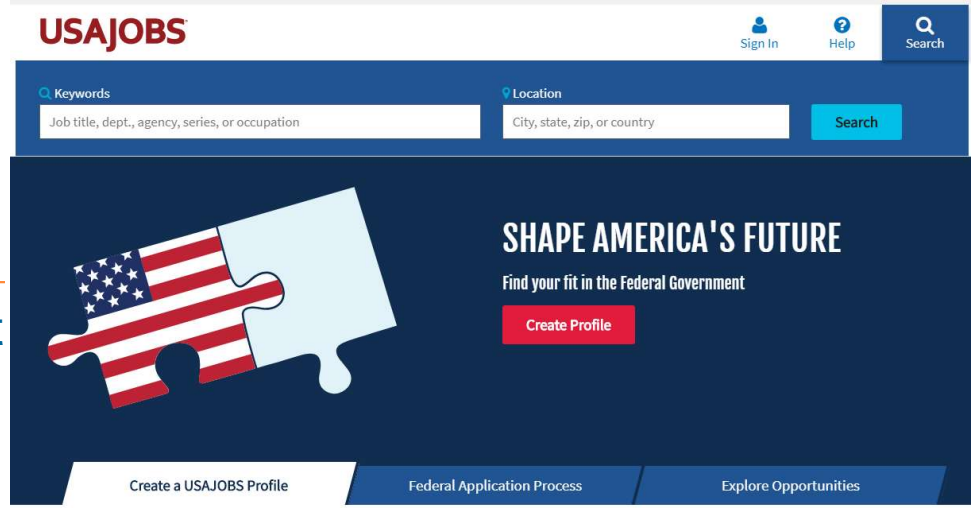
Construct a Multipart Media Query

Unit I

- Implementing responsive design:
  - 3. Identify the first breakpoint



Breakpoint

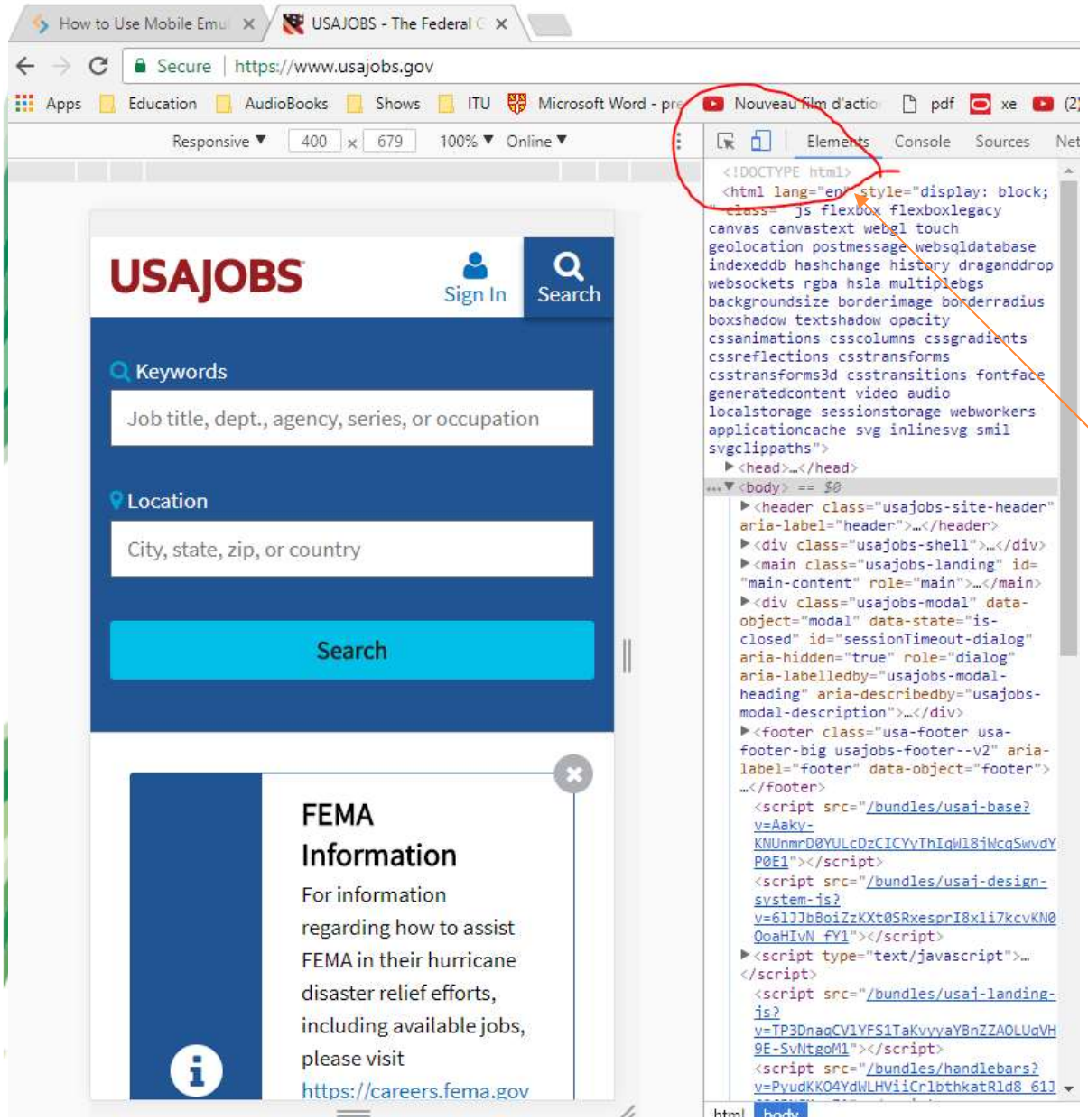


Create a USAJOBS Profile

```
@media only screen and (max-width: 768px) {
  /* For mobile phones: */
  /* define your layout here */
}
```

Chrome F12

Unit I



Start Chrome, navigate to the web page you want to test and open the **Developer Tools** (Menu > Tools > Developer Tools, Cmd+Opt+I on Mac or F12 / Ctrl+Shift+I on Windows and Linux).

You can now enable the browser emulator by clicking the **Toggle device toolbar** icon in the top-left:

Part II: Continuation

Unit I

```
@media(min-width:900px){p{color:red;}}
```

1. @media – query announcement
2. (min-width:900px) – conditional
3. {p{color:red;}} – What should I do if the condition is met?

Purpose:

1. build modern websites and blogs
2. Allow different css instructions to be used for different size screens.

In a nutshell.

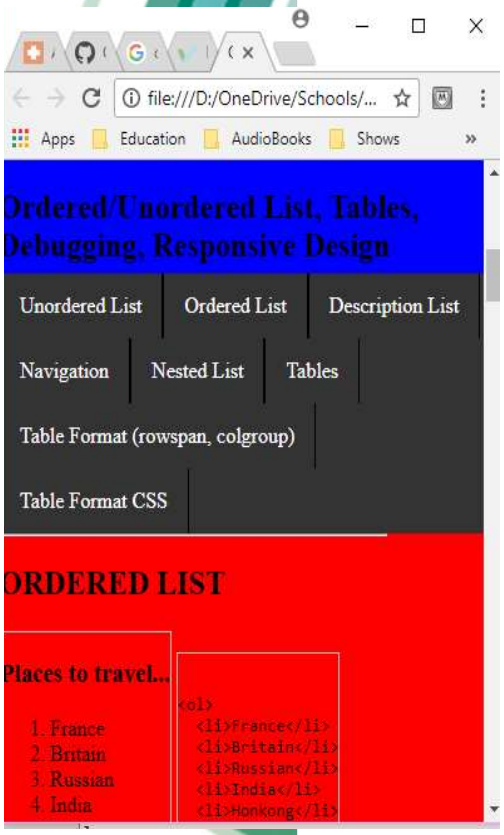
Unit I

How do media queries work?

1. To find out how big a screen is
2. To apply CSS for the size of the screen found.

Analysis: Example

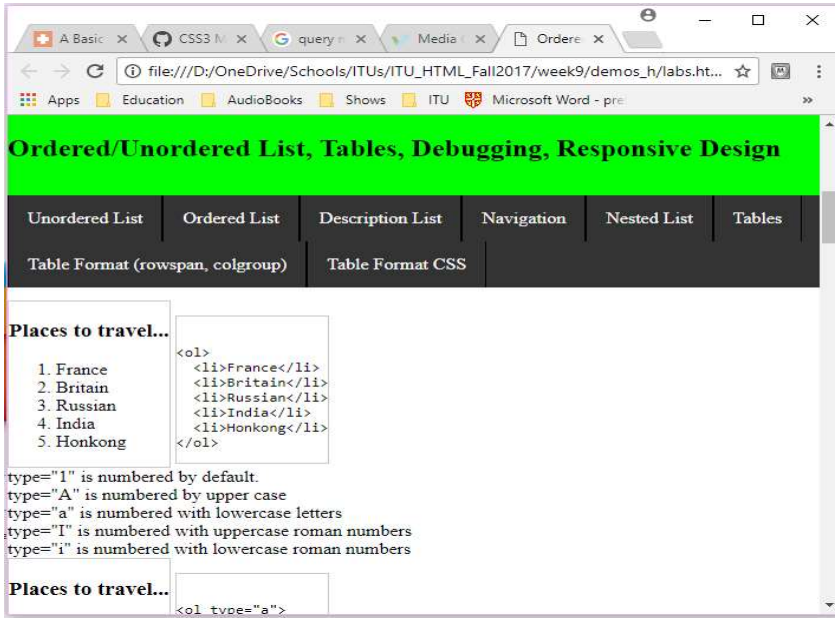
Unit I



Resize by Media Query

```
@media only screen and (max-width: 600px) {
body{background-color: red;}
header {background-color:blue;
}}
```

Normal



Reminder: Viewport

Unit I

- Set the viewport when making responsive design:

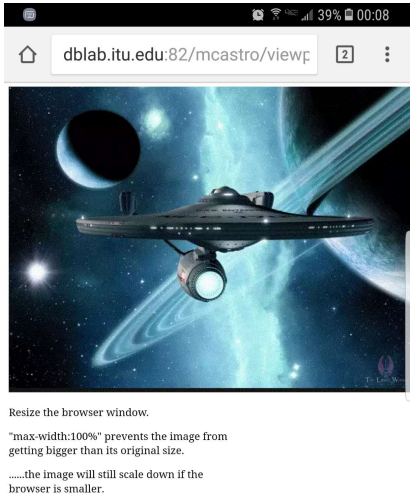
<meta name="viewport" content="width=device-width, initial-scale=1.0“>

WHY?

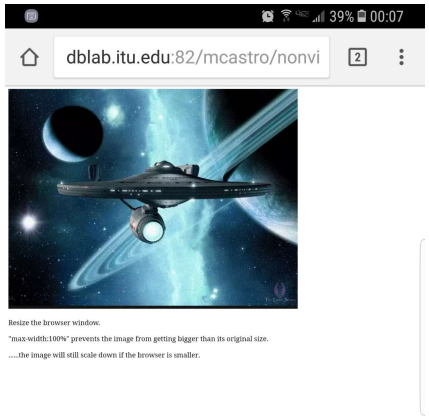
It gives the browsers’ instruction how to control the :

1. Page dimension
2. Scaling

Should be part of Adaptive design



With viewport



With none viewport

Add a column with a media query

Unit I

- It can be helpful to vary the number of columns in which content is displayed
  - Small screens = single column
  - Larger screens = multiple columns (sidebar, or placing elements side by side)
- Create columns in a media query using properties like `float`, `width`, and `position`

Create a Widescreen Layout

Unit I

- Larger screens provide extra width
- Common to add an additional column
- Can also specify a breakpoint at which content width stops increasing
  - Use `@media` rule instead of `max-width` property in default style for element so other developers can easily understand your code

Create Responsive Navigation

Unit I

- Can be useful to hide elements on smaller screens and let users show them
- Hamburger menu: a button showing 3 horizontal lines that is often used to replace the nav bar 
  - Click/touch button to show nav bar
  - Click/touch button again to hide nav bar
  - Uses JavaScript

Implement Adaptive Content

Unit I

- All content available on larger devices should also be available on smaller ones
- Adaptive content: limiting amount of content shown by default and making related information available only if user requests it

Implement Adaptive Content (cont'd)

Unit I

- Methods of implementing adaptive content
  - Change display of elements in media queries using class names
  - Use a structural pseudo-class

Implement Adaptive Content (cont'd)

Unit I

Table I-2: Structural pseudo-classes

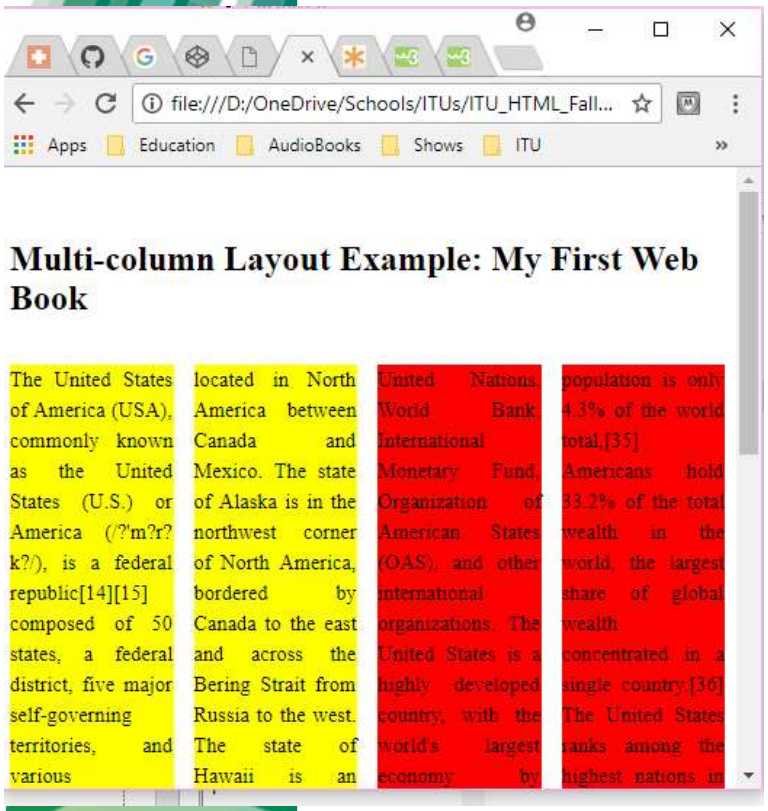
pseudo-class	description*	example	selects
`:first-child` `:last-child`	the first or last child element if it is of the specified type	`p:first-child`	first `p` child element if it is also the first child
`:first-of-type` `:last-of-type`	the first or last child element of the specified type	`p:last-of-type`	last `p` child element
`:nth-child(n)`	the *n*th child element of the specified type	`li:nth-child(3)`	the third `li` child element
`:nth-last-child(n)`	the *n*th from last child element of the specified type	`li:nth-last-child(3)`	the third from last `li` child element
`:nth-of-type(n)`	the *n*th occurrence of the specified child element	`p:nth-of-type(5)`	the fifth `p` child element
`:nth-last-of-type(n)`	the *n*th from last occurrence of the specified child element	`p:nth-last-of-type(5)`	the fifth from last `p` child element

* where *n* is a number or a calculation in the form *an+b*; note that IE8 does not support the structural pseudo-classes listed in this table

See the sample for these properties along with the Media queries… next slides…

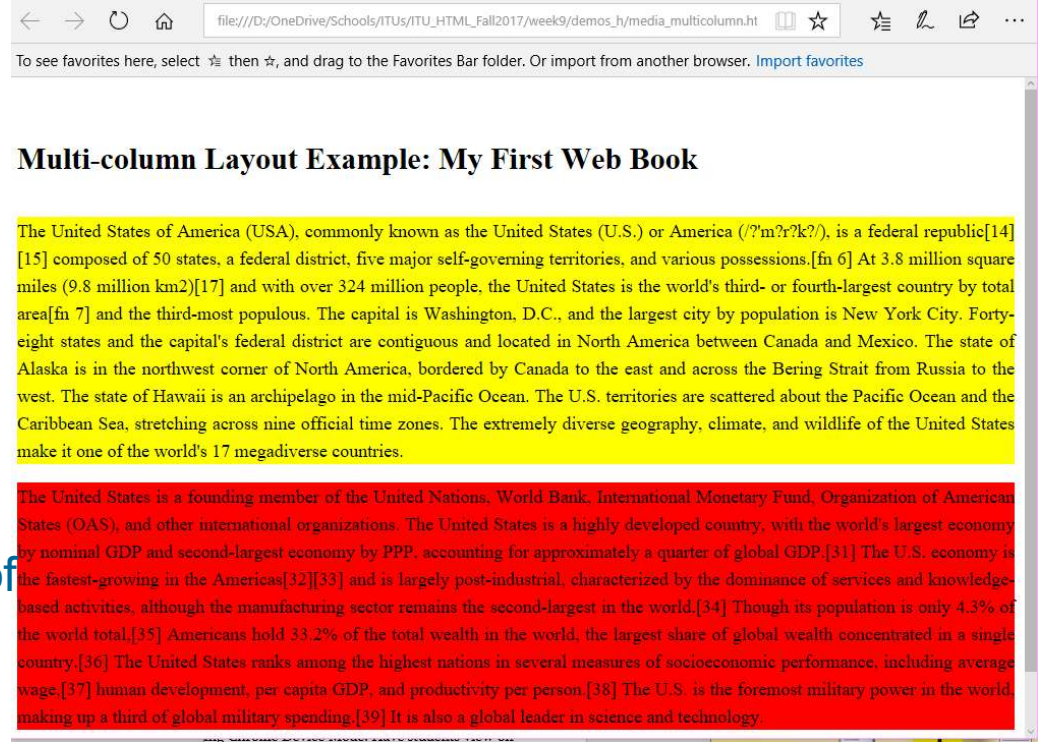
Demo: media queries/columns.

Unit I




```
p:first-child {
  margin-top: 0;
  background-color: yellow;
}

p:last-child {
  background-color:red;
}
```




```
@media (max-width: 800px) {
  .multi-column.long-columns {
    columns: 100px 7; /*width and # of columns*/
  }
}
```

Use Progressive Enhancement

Unit I

- Older browsers usually don't support newer CSS properties
- Shim: recreates a newer feature for older browsers using JavaScript
- Progressive enhancement: using newer features without affecting the meaning/functionality of content
  - Example: specifying background color as fallback for background image

Summary

Unit I

- Responsive design allows a web developer to specify different CSS rules for some/all elements depending on width of screen
- To implement responsive design, create a default layout for smallest or largest browser width to support, identify the first breakpoint, then create a multipart media query

Summary (cont'd)

Unit I

- While developing a website, you can use an emulator to test how your pages might function on a variety of devices
- You can use media queries to add columns to your layout for larger screens

Summary (cont'd)

Unit I

- You can create a layout for wide screens with an additional column and/or a maximum width for content
- Create responsive navigation by hiding elements such as nav bars on smaller screens and replacing them with items like hamburger menus that users click to display them

Summary (cont'd)

Unit I

- Change what content is visible at different screen sizes by using class names or structural pseudo-classes in media queries
- You can use shims to simulate newer features in older browsers and use progressive enhancement to ensure that newer features don't affect the user experience in older browsers