

# HTML5 and CSS3 Complete

Second Edition

## Unit I

### Implementing Responsive Design

# Objectives

- 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)

- 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



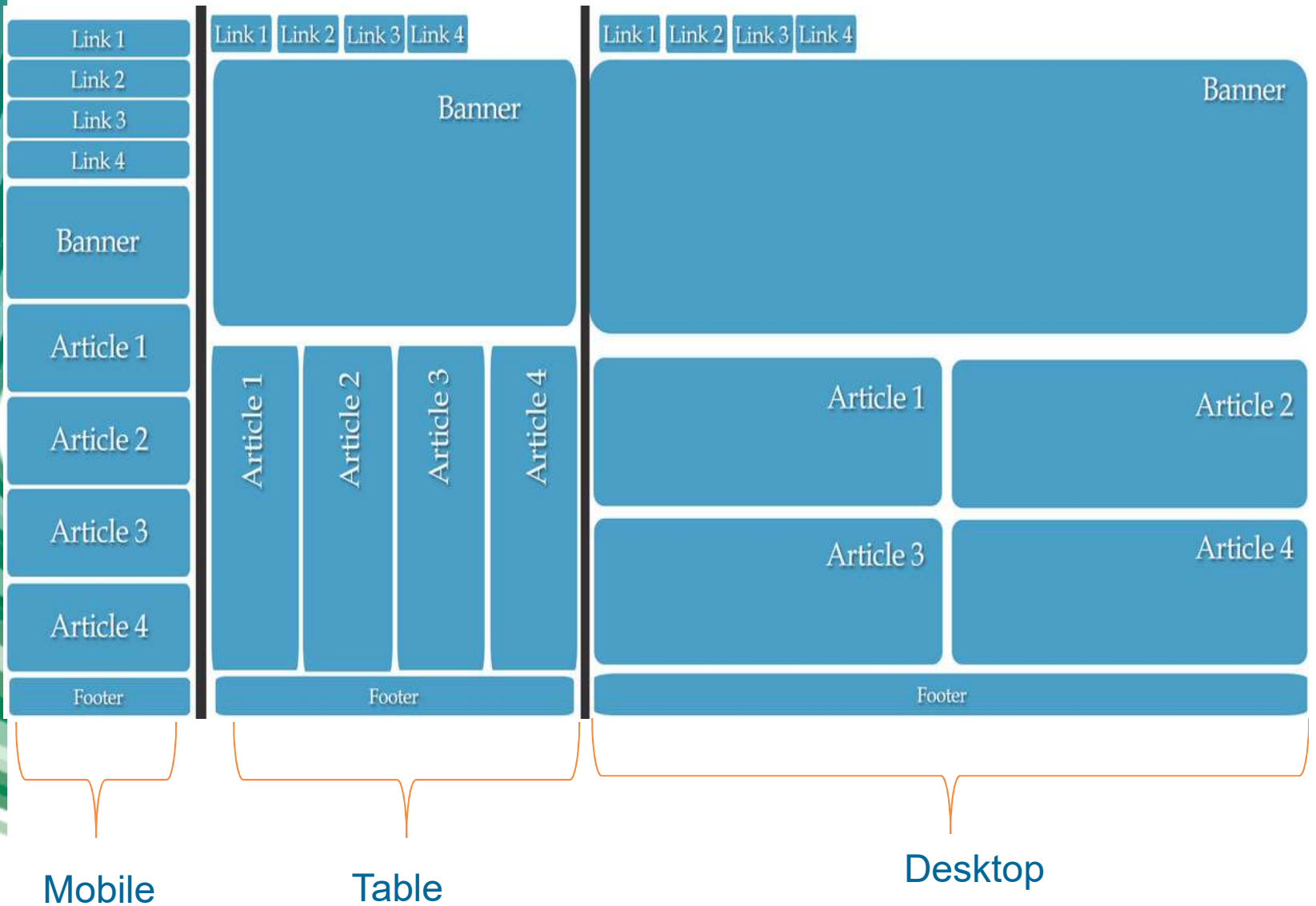
Tablet



Desktop

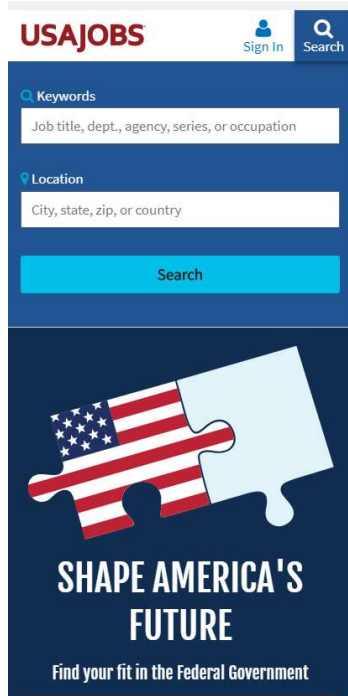
# Example of Screen Sizes

Unit I

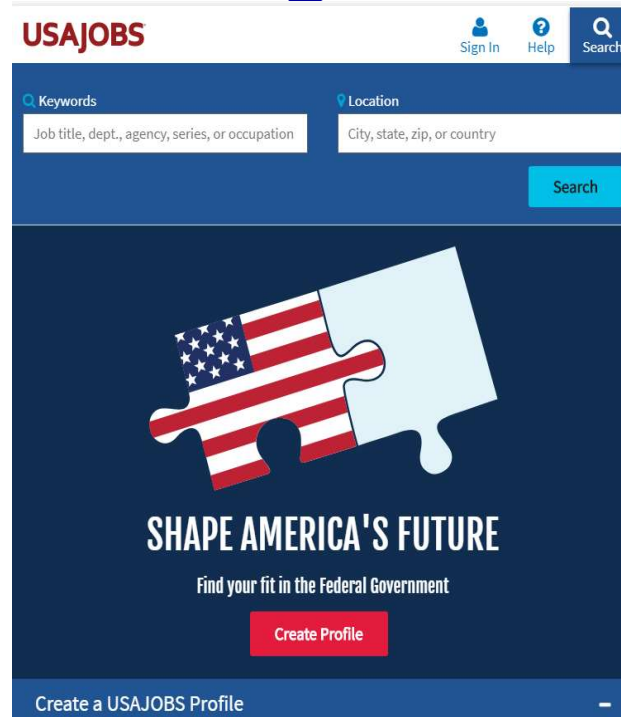




# Screen sizes using emulator

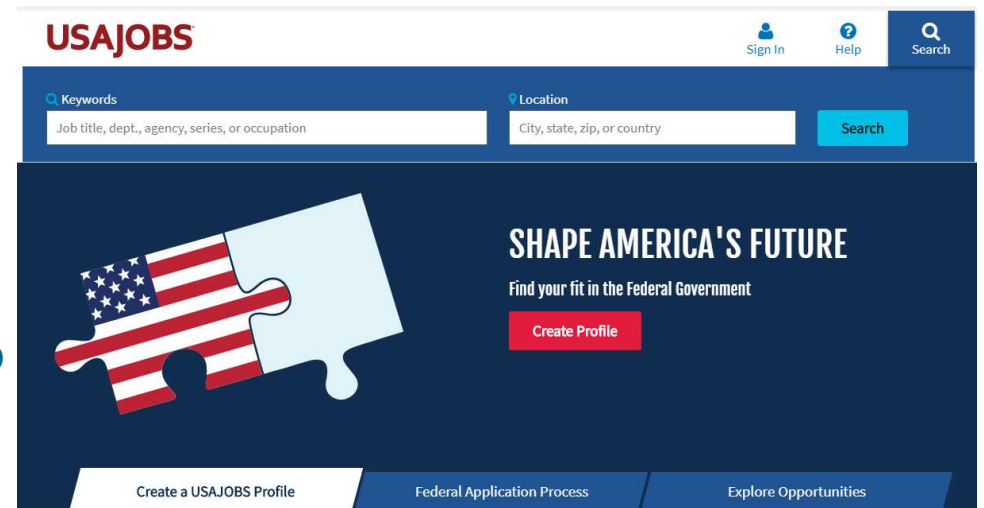


mobile



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

Responsive ▼ 768 × 862 100% ▼ Online ▼



# 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)

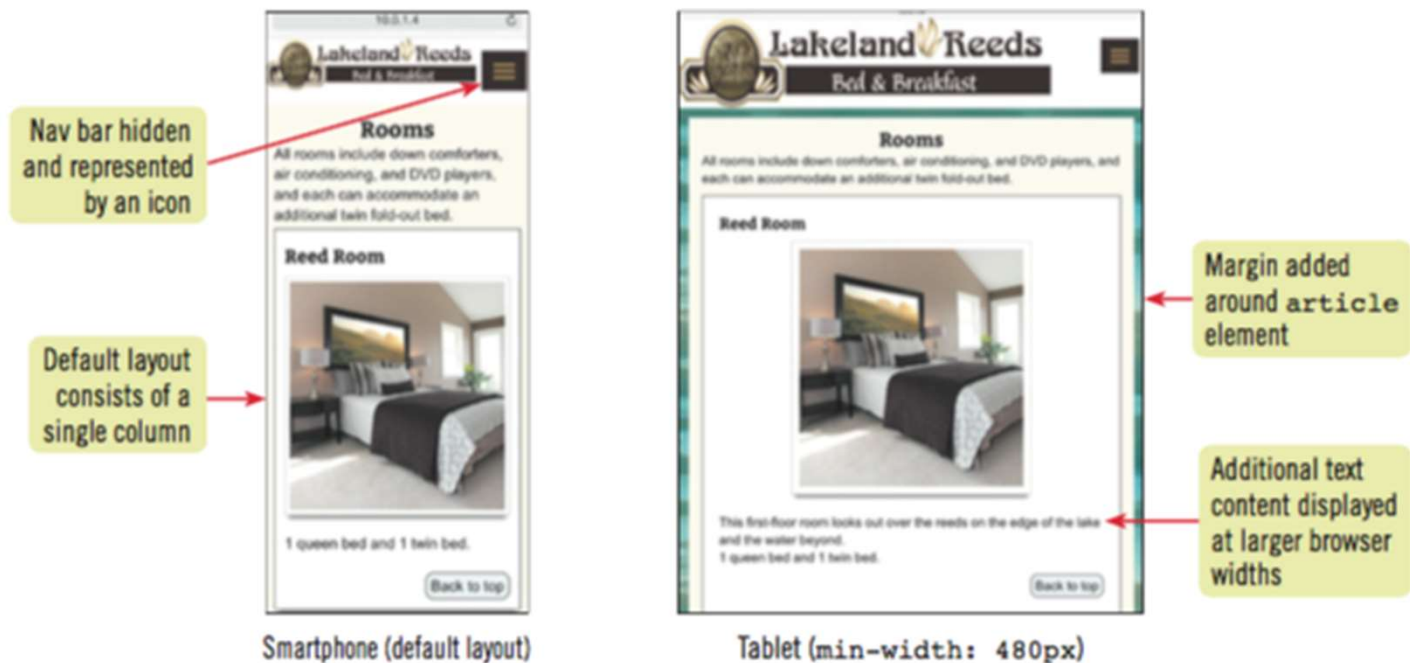
- Example:

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

# Assess Responsive Design (continued)

## Unit I

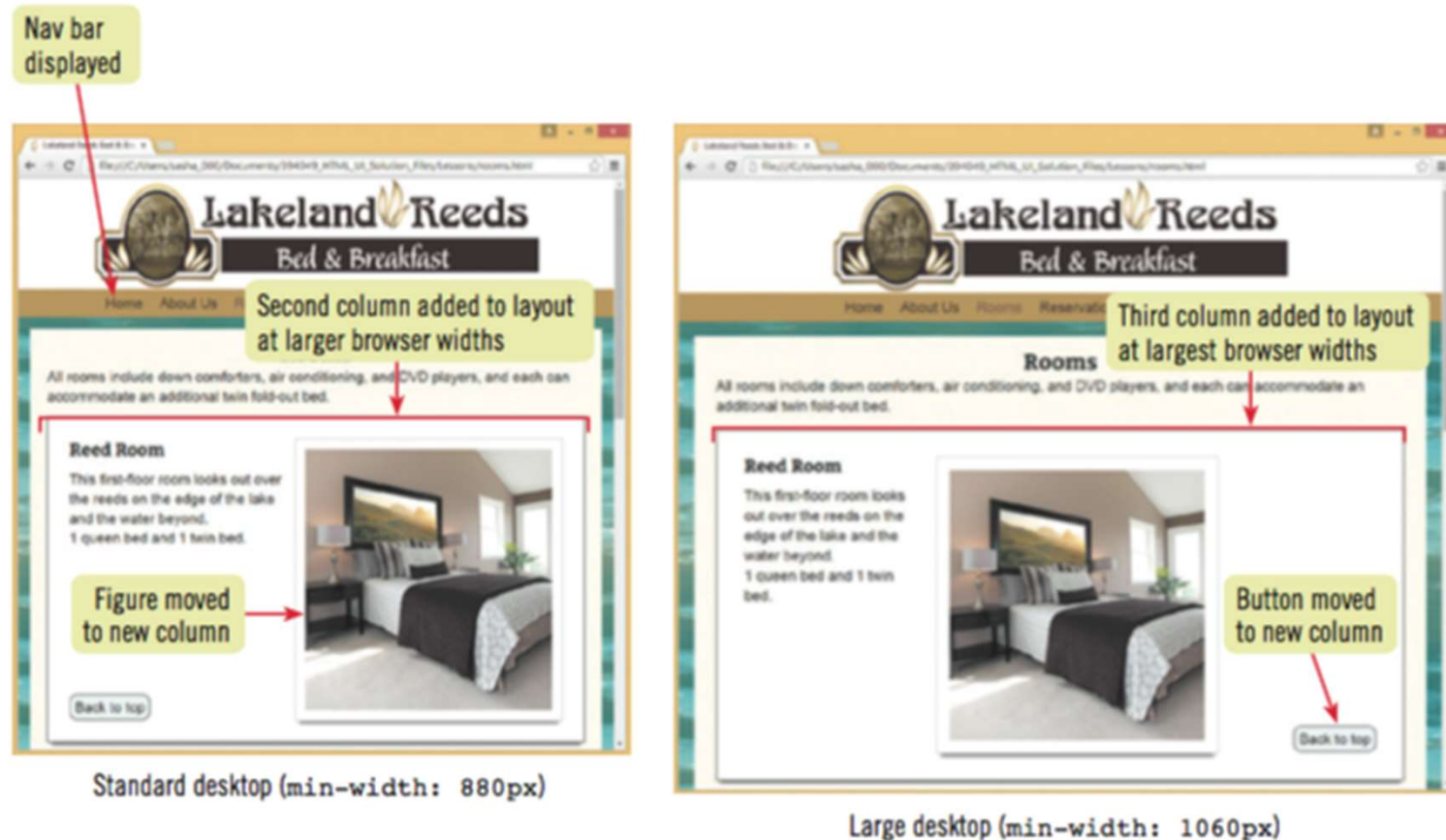
Figure I-1: Web page displayed on different devices using different media queries



# Assess Responsive Design (continued)

## Unit I

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





# 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.

smart phone - Google Search Ordered/Unordered List X + -

file:///D:/OneDrive/Schools/ITUs/ITU\_HTML\_Fall2017/week8/demos\_h/labs.htm

To see favorites here, select ☆ then ☆, and drag to the Favorites Bar folder. Or import from another browser. [Import favorites](#)

## Ordered/Unordered List, Tables, Debugging, Responsive Design

Unordered List	Ordered List	Description List	Navigation	Nested List	Tables
Table Format (rowspan, colgroup)		Table Format CSS			

[Yahoo!](#)

### UNORDERED LIST

Places to travel...

- France
- Britain
- Russian
- India
- Honkong

```
<ul>
  <li>France</li>
  <li>Britain</li>
  <li>Russian</li>
  <li>India</li>
  <li>Honkong</li>
</ul>
```

"list-style-type" property defines the style of the list item marker.

disc - lists item marker to a bullet (default)



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

file:///D:/OneDrive/Schools/ITUs/ITU\_H

To see favorites here, select ☆ then ☆, and drag to the Favorites Bar folder. Or import from another browser

## Ordered/Unordered List, Tables, Debugging, Responsive Design

Unordered List	Ordered List	Description List	Navigation
Nested List	Tables	Table Format (rowspan, colgroup)	
Table Format CSS			

Places to travel...

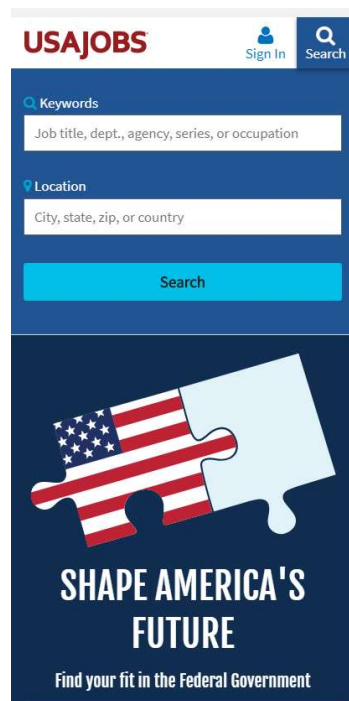
- France
- Britain
- Russian
- India
- Honkong

```
<ul>
  <li>France</li>
  <li>Britain</li>
  <li>Russian</li>
  <li>India</li>
  <li>Honkong</li>
</ul>
```

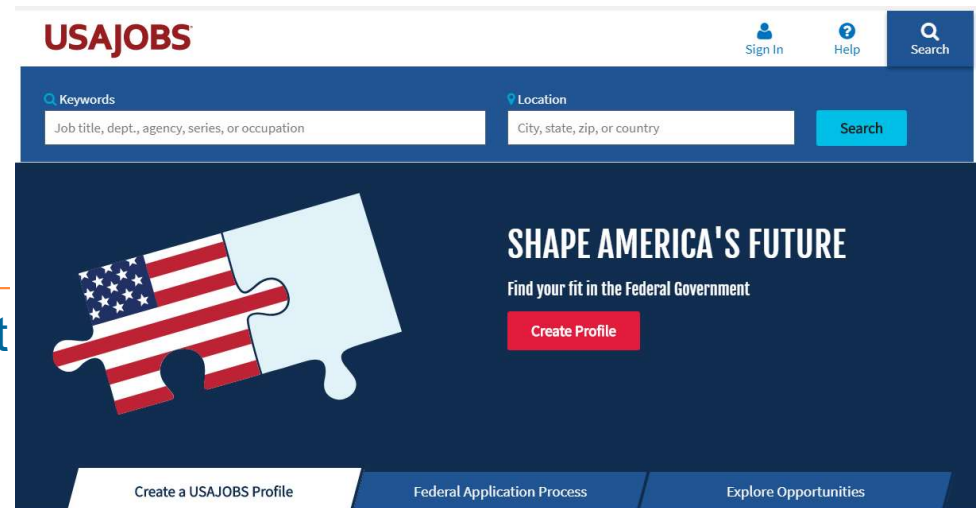
# Construct a Multipart Media Query

## Unit I

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



Breakpoint

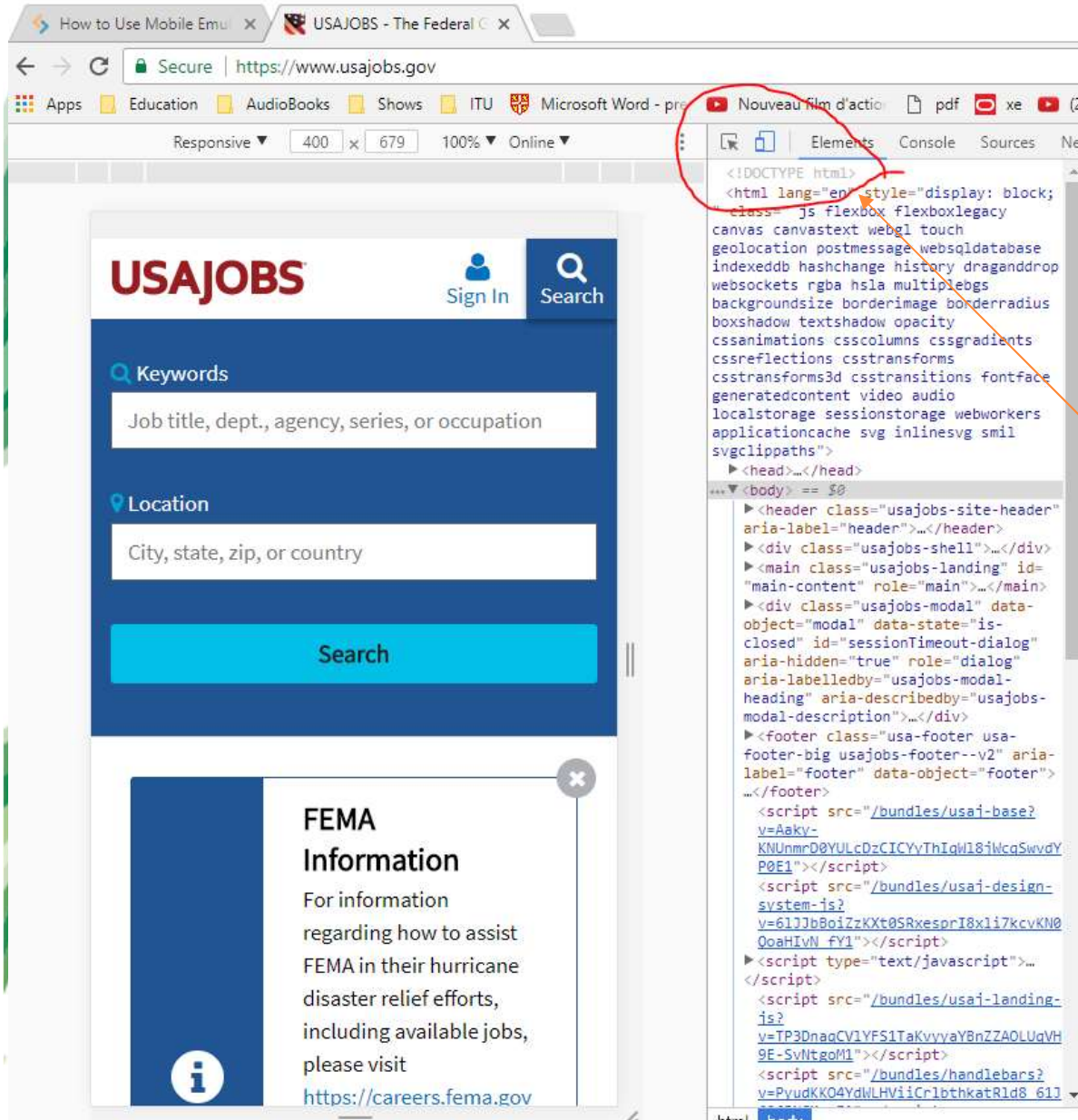


Create a USAJOBS Profile

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

# Chrome F12

## Unit I



The screenshot shows a Chrome browser window with the URL <https://www.usajobs.gov>. The browser is in mobile emulator mode, with a resolution of 400x679. The USAJOBS website is displayed, featuring a search bar, a location filter, and a search button. A FEMA Information banner is visible at the bottom. The developer tools are open, and the 'Elements' tab is selected, showing the HTML structure of the page. A red circle highlights the 'Elements' tab, and an orange arrow points from the text on the right to it.

```
<!DOCTYPE html>
<html lang="en" style="display: block;
  class= "js flexbox flexboxlegacy
  canvas canvastext webgl touch
  geolocation postmessage websql database
  indexeddb hashchange history draganddrop
  websockets rgba hsla multiplebgs
  backgroundsize borderimage borderradius
  boxshadow textshadow opacity
  cssanimations csscolumns cssgradients
  cssreflections csstransforms
  csstransforms3d csstransitions fontface
  generatedcontent video audio
  localstorage sessionstorage webworkers
  applicationcache svg inlinesvg smil
  svgclippaths">
  <head>...</head>
  <body>...
    <header class="usajobs-site-header"
      aria-label="header">...</header>
    <div class="usajobs-shell">...</div>
    <main class="usajobs-landing" id=
      "main-content" role="main">...</main>
    <div class="usajobs-modal" data-
      object="modal" data-state="is-
      closed" id="sessionTimeout-dialog"
      aria-hidden="true" role="dialog"
      aria-labelledby="usajobs-modal-
      heading" aria-describedby="usajobs-
      modal-description">...</div>
    <footer class="usa-footer usa-
      footer-big usajobs-footer--v2" aria-
      label="footer" data-object="footer">
      ...</footer>
    <script src="/bundles/usaj-base?
      v=Aakv-
      KNUnmrD0YULCDzCICyYThIqH18iWcqSwvdY
      P0E1"></script>
    <script src="/bundles/usaj-design-
      system-is?
      v=61J7bBoiZzKXt0SRxesprI8x1i7kcvKN0
      QoaHivN_fY1"></script>
    <script type="text/javascript">...
    </script>
    <script src="/bundles/usaj-landing-
      is?
      v=TP3DnaoCV1YF51TaKvyyaY8nZZAOLUqVH
      9E-SvNtgoM1"></script>
    <script src="/bundles/handlebars?
      v=PyudKKO4YdWLVHviCrlbthkatR1d8_611
```

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

```
@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.

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

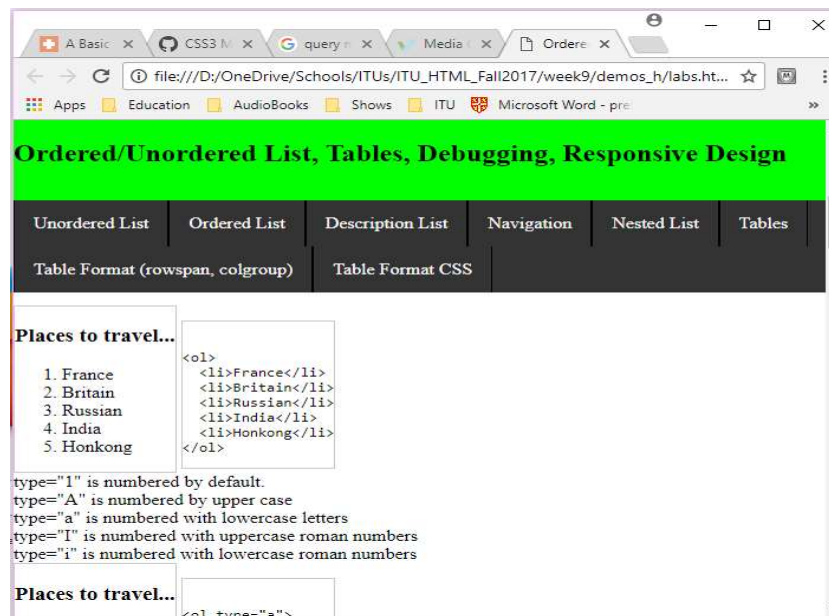
## Unit I

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



Resize by Media Query

Normal



# Reminder: Viewport

## Unit I

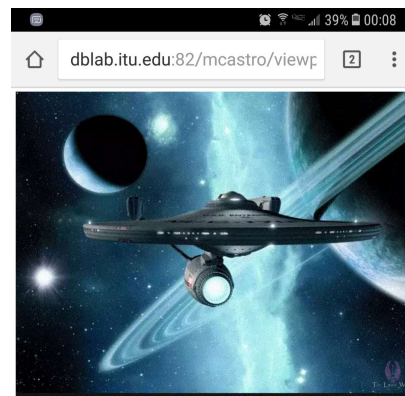
- Set the viewport when making responsive design:  
`<meta name="viewport" content="width=device-width, initial-scale=1.0">`

WHY?

It gives the browsers' instruction how to control the :

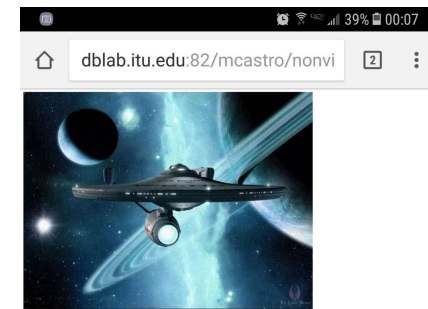
1. Page dimension
2. Scaling

Should be  
part of  
Adaptive  
design



Resize the browser window.  
"max-width:100%" prevents the image from getting bigger than its original size.  
.....the image will still scale down if the browser is smaller.

With viewport



Resize the browser window.  
"max-width:100%" prevents the image from getting bigger than its original size.  
.....the image will still scale down if the browser is smaller.

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

- 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)

- 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
<code>:first-child</code> <code>:last-child</code>	the first or last child element if it is of the specified type	<code>p:first-child</code>	first <i>p</i> child element if it is also the first child
<code>:first-of-type</code> <code>:last-of-type</code>	the first or last child element of the specified type	<code>p:last-of-type</code>	last <i>p</i> child element
<code>:nth-child(<i>n</i>)</code>	the <i>n</i> th child element of the specified type	<code>li:nth-child(3)</code>	the third <i>li</i> child element
<code>:nth-last-child(<i>n</i>)</code>	the <i>n</i> th from last child element of the specified type	<code>li:nth-last-child(3)</code>	the third from last <i>li</i> child element
<code>:nth-of-type(<i>n</i>)</code>	the <i>n</i> th occurrence of the specified child element	<code>p:nth-of-type(5)</code>	the fifth <i>p</i> child element
<code>:nth-last-of-type(<i>n</i>)</code>	the <i>n</i> th from last occurrence of the specified child element	<code>p:nth-last-of-type(5)</code>	the fifth from last <i>p</i> child element

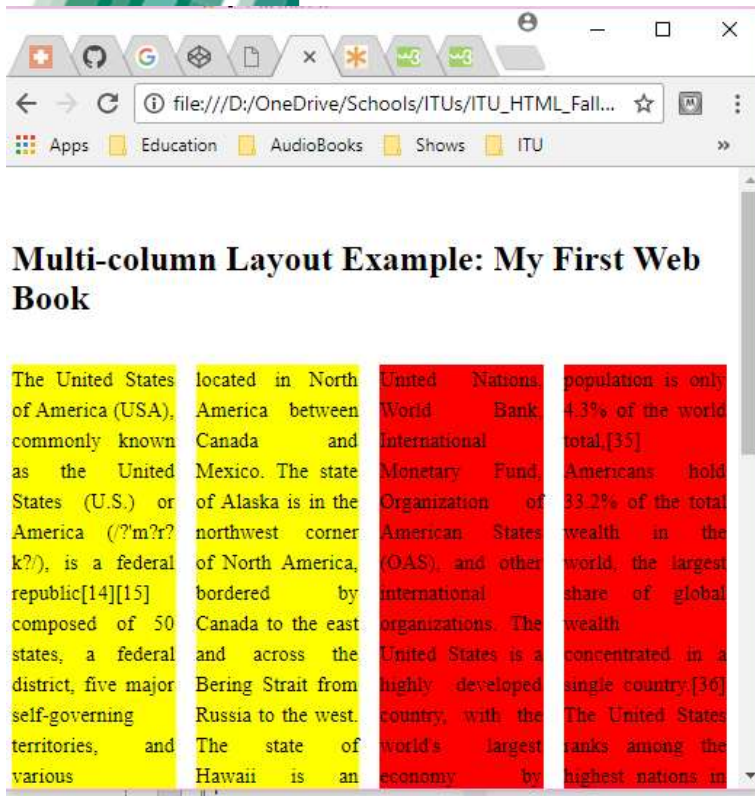
\* where *n* is a number or a calculation in the form *an+b*; note that IE8 does not support the structural pseudo-classes listed in this table

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



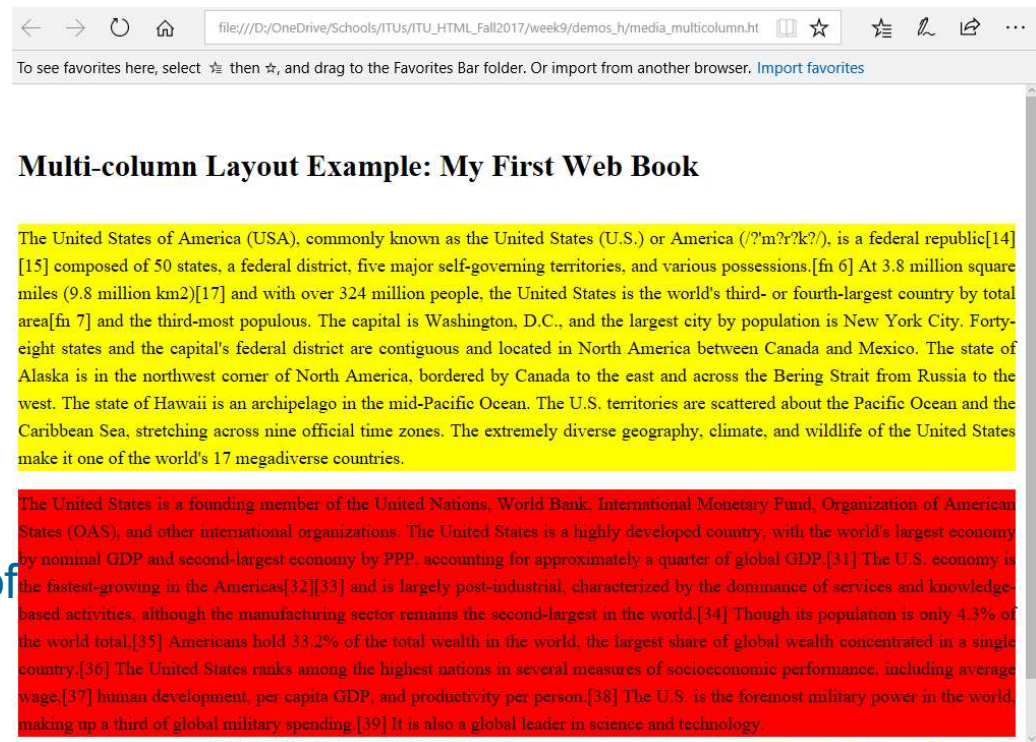
# Demo: media queries/columns.

## 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