

# Objectives

- Review CSS
- Discuss: How Google Search Works
- Usability: Responsive design and Bootstrap
- HTML Forms

# CSS Review

- Why CSS?
- What is the syntax of a CSS rule?
- What is the order of applying rules in the cascade?
- What is your favorite styling property?
- When should you use a class? an id?

# Identify the Errors

```
<style>
body {      background-color: white
h1, {      gray;      font-family: "sans-serif"; }
h2, p {      color: }
<em> {      font-style: italic; }
</style>
```

# Fixed CSS

```
<style>
body {      background-color: white; }
h1, {      color: gray;      font-family: sans-serif; }
h2, p {    color: blue; }
<em> {     font-style: italic; }
</style>
```

Could be other ways to fix the errors

# Reflection

- Why do we have two languages: HTML and CSS?
  - Why do we typically store them in separate files instead of having both in one file?
- Bonus: Use software design principles/patterns to describe

Google SEARCH

# Google Search Discussion

- How does Google's search work?
- What are some of its novel features?
- What are some recommendations for web masters?
  - Any surprises?
- Why is search engine optimization important?
  - It's so important that it has its own acronym: SEO
- What are Google search's limitations?
  - How can it be manipulated?
- What features would you like to add to Google's searching technique (or other search engine)?
- What is your preferred search engine? Why?




# Google Search Discussion

- Knowing more about Google search, will you change your queries any?
  - How can you make them more effective?
  - How can you save yourself time?
- Now that you're an author of HTML documents, will you change them to get a higher rank on Google?
  - How could you do that?



# USABILITY

# Approaches to Software Design

- **Inside-out**  Traditional CS Courses are almost entirely inside-out
  - Develop a system
  - Add an interface
- **Outside-in**  Modern systems need to be designed outside-in to be effective.
  - Develop the interface  Web sites especially need to be usable.
  - Then build the system to support it
- When design decisions are made, either the developer must conform to the user, or the user must conform to the developer.

# User-Friendly

- The term **user-friendly** is over-used and under-defined
  - What is “friendly” to one person may be trite, tedious, or confusing to another
- “User appropriate” is a much more meaningful term
  - But we have to know the user

# Usability

- Engineering principles for designing and building software interfaces that are
  - Fast to learn
  - Speedy to use
  - Avoid user errors
- How to recognize and articulate the difference between “this page sucks” and “I can improve this page by changing X,Y, and Z”
- Life-long habits for engineering usable products

# Usability So Far

- Layout
  - Clear visual hierarchy
    - Headings - break up pages
    - Nesting content
  - More important something is, the more prominent it is
- Links
  - Make them ancillary to text
  - Visited in different color
- Images
  - Use the alt attribute

# Usability

- Undercurrent throughout the class
- Right now: responsive design

# USABILITY: RESPONSIVE DESIGN

# Responsive Design

- Recall: Design a web page so that it looks good, easy to use regardless of screen size



# Bootstrap

- I included Bootstrap as a stylesheet in the example HTML file I gave you
  - But, we didn't talk about it much

# Bootstrap

- Popular framework for creating responsive web sites quickly and easily
  - CSS (and JavaScript)
- Started at Twitter but now an independent project
  - Original goal: *consistent* look of internal tools
- Defines CSS classes
  - You can use as is OR override or supplement Bootstrap's CSS (a win for cascading!)
  - Learn what the classes do, then add those classes to your HTML

# Common Development Workflow

- Goal: I want to make my page look like X or I want to add a Y to my page
  - Examples: navigation bar, alerts, “cards”, carousels, buttons
- Check: does Bootstrap have a way to do this?
  - If so, apply relevant classes/HTML to your web page

## Recall: cLass Attribute

- The **class** attribute specifies one *or more* class names for an element.
  - To specify multiple classes, separate the class names with a space
- *All* classes' CSS will apply to that element

# Bootstrap Nav Bar Example

Navbar Home Features Pricing Disabled

Navbar



```
<nav class="navbar navbar-expand-lg bg-body
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-targ
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
      <div class="navbar-nav">
        <a class="nav-link active" aria-current="page" href="#">Home</a>
        <a class="nav-link" href="#">Features</a>
        <a class="nav-link" href="#">Pricing</a>
        <a class="nav-link disabled">Disabled</a>
      </div>
    </div>
  </div>
</nav>
```

<https://getbootstrap.com/docs/5.3/components/navbar/>

# Bootstrap Nav Bar Example

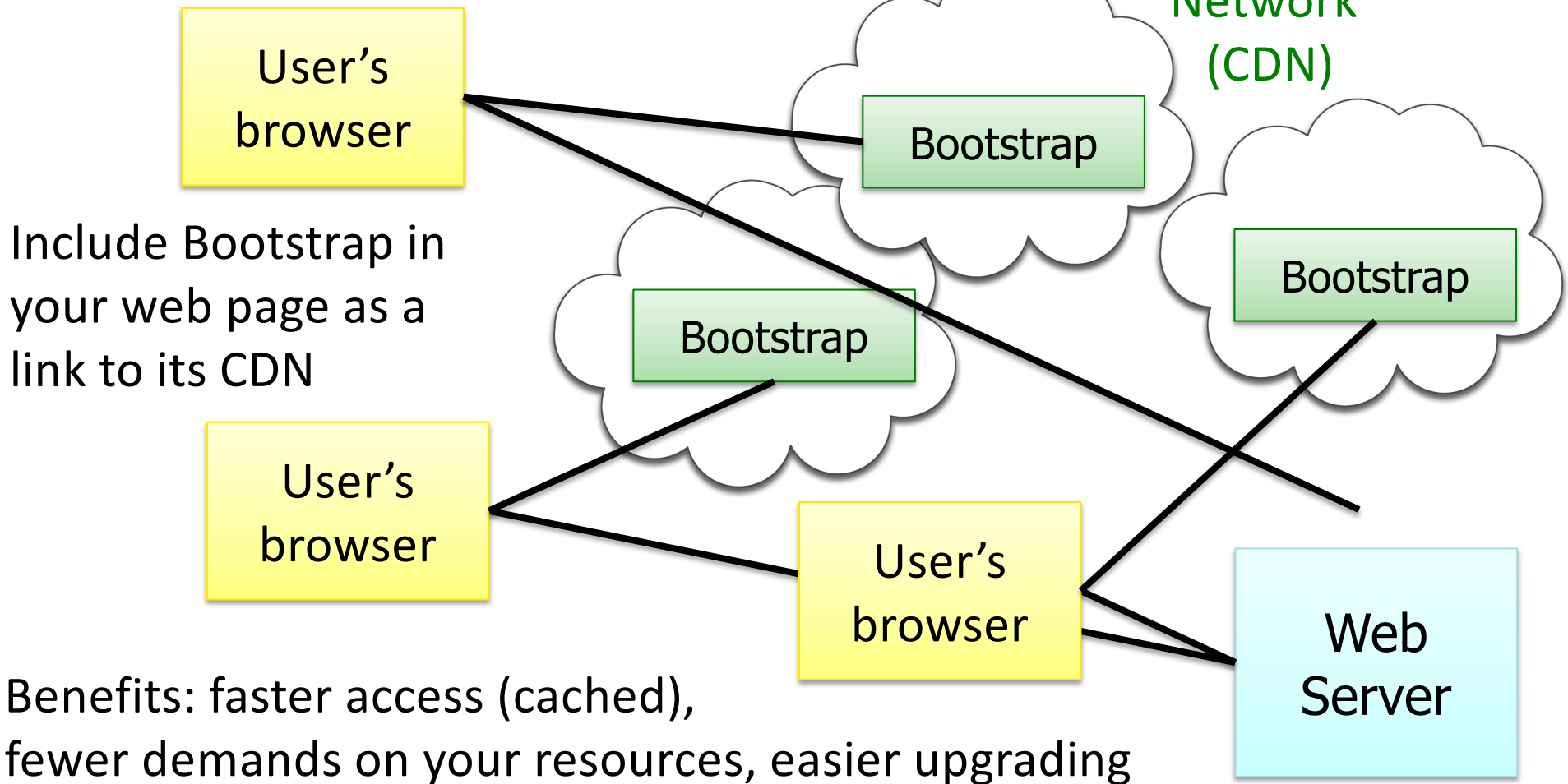
- Collapse class: for responsive design
  - If the width of the browser is narrow, then collapse the menu into a “hamburger” menu
- You’d fill in the menu options, the pages they link to, the brand, etc.
- Various classes to style the element (e.g., navbar-dark)

# A Lot More to Bootstrap

- But we have a lot more to cover
- Process
  - Look up what you need
  - Understand examples
  - Adapt!

# Using Bootstrap

Content Distribution Network (CDN)



Include Bootstrap in your web page as a link to its CDN

Benefits: faster access (cached), fewer demands on your resources, easier upgrading



# Bootstrap: Pros and Cons?

# Bootstrap: Pros and Cons

- Pros

- It looks good!
- Has a lot of stuff!
  - Supports **Accessible** Rich Internet Applications
- Commonly used – lots of documentation, resources
- Handles responsive design


- Cons

- Learning curve
- *Everyone* uses it! Everyone's sites look the same!
- Blurs line between structure and style
  - Makes maintenance a little tricky
- Additional learning curve

User Interface

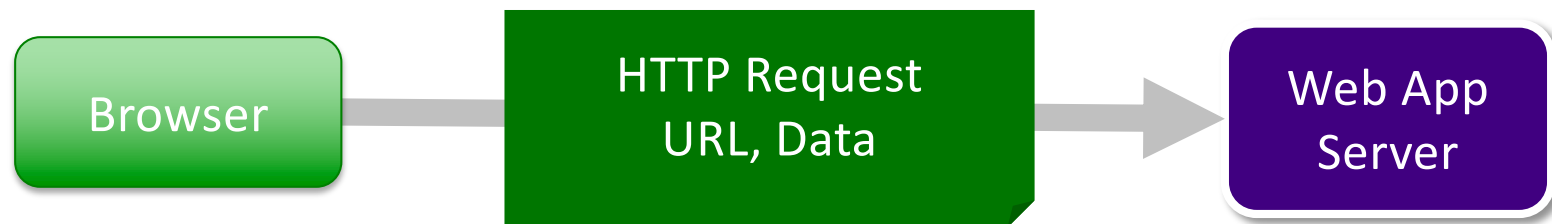
# FORMS

# Web Interfaces

- Menus
  - Sidebars, links, navigation
- GUIs
- Forms  Today's focus

# HTML form Tag

- Used to provide web application input from the user
- Contains various ***controls*** or ***widgets*** (sources of input) and labels for controls
- Must have a **submit** button that transmits all input data to server as a query string



# Examples of Forms/Input



Google Search I'm Feeling Lucky

Explore our planet through the years with Google Earth

**Member Login**

E-Mail:

Password:

Remember Me

[Forgot your password?](#)

**Friends are coming over for dinner. You wow them with...**

- Your amazing cooking and homemaking skills. Julia Childs weeps with pride. ↻
- Your comedic skills. Pass out the tissues. ↻
- Your wisdom and masterful conversation. ↻
- Your ability to throw together a memorable event with seemingly little effort. ↻
- Your ability to be scathingly sarcastic while still loveable. ↻
- Your amazing music collection and kick ass stereo system. ↻
- Your beautiful smile. ↻

# HTML form Tag

- form attributes:

- **action** (required): URL to send request to

- Relative or absolute

- **method**: `get` or `post`

- Default: Get
- More on differences later...

```
<form action="/search">  
  <!-- input fields, etc. -->  
</form>
```

# Input From Forms

- Types of input data
  - Text
  - Checkbox
  - Radio buttons
  - Select from list
  - Password
- Data is identified by a **name**, has a **value**
  - Specified by input fields' **name** attributes



# Simplified Google Search Form

```
<form action="/search">
  <p>
    <input type="text" name="q" size="55"/>
    <input type="submit" value="Google Search"
      name="btnG"/>
  </p>
</form>
```

- Form is submitted to Google's `/search` page with data `q` set to whatever user entered in box, e.g.,
  - `https://www.google.com/search?q=user_query`

# Another Way to Invoke A Web Application

- Example: Google
- Create a link to
  - <https://www.google.com/search?q=web+applications>
- Browser sends a **GET** request to the server's **search** page
  - 1 Parameter:
    - name is **q**, which has value **web+applications**
      - The + is the encoding for a space

# Web Server Receives Request

- Receives the request
- Responds to the request
- Sends back a response
- For example, Google performs the search for “web applications” and returns an HTML document with the search results.

# Discussion: Query Strings

- Do you always want your input data to show up in the URL?
- Example:  
`https://www.google.com/search?q=web+applications`

# get vs post

- **get** passes parameters to server as a *query string*
  - Limited to *browser's* URL length
- **post** embeds the parameters in *HTTP request body*
  - **Not** in the URL

# post

## Advantages

- Information is more private (not shown in URL)
- Can't be bookmarked

## Disadvantages

- Can't be bookmarked
- Browser can't easily go back (POSTDATA error)

# input Tag

- Used to create many of the widgets
- **type** attribute specifies the type of widget
- Must be inside a block-level element
- Contains attributes
- Examples:
  - text, checkbox, radio
- Often requires **name** attribute
  - Names the data that will be sent to the Web application

# Text input

Enter Text:

- A horizontal box that the user can input text into

```
<input type="text" name="name" size="25"/>
```

## Attributes:

<b>size</b>	Width of text box in characters; scrolls if user types more
<b>maxlength</b>	Maximum number of characters browser accepts in a box
<b>value</b>	Provide a default value

Examples of when to use value?



# Password input

Password:

- A horizontal box that the user can input text into **but** the text displays as \*s or •s

```
<input type="password" name="mypassword"
      size="10" maxlength="10"/>
```

## Attributes:

<b>size</b>	Width of text box in characters; scrolls if user types more
<b>maxlength</b>	Maximum number of characters browser accepts in a box

# Labeling input fields: **Label**

- Label a control with the **Label** element
- Better than labeling with other text because
  - Can get separated during maintenance
  - **Label** has special presentation
  - Improved usability

```
<p>  
  <label>Password:  
  <input type="password"  
        name="mypassword" size="10"  
        maxLength="10"/>  
  </label>  
</p>
```

# Checkboxes and Radio Buttons

- How are they different? How are they used?

# Multiple Choice Input: **checkbox**

- Use when user has multiple choices for something and can choose  $\geq 1$

Milk  Bread  Eggs

- Only items that user checks are sent by the form to the action location

# Multiple Choice Input: **checkbox**

- All checkboxes in a "group" have the same **name**
- Checkbox requires a **value** attribute
  - **value** is submitted in the form data iff the checkbox is 'checked'
- To make a checkbox checked, add the **checked** attribute (which doesn't have a value)

```
<label>  
  <input type="checkbox" name="groceries"  
        value="milk" checked/>Milk  
</label>
```

# Multiple Choice Input: checkbox

```
<label>  
  <input type="checkbox" name="groceries"  
        value="milk" checked/>Milk  
</label>  
<label>  
  <input type="checkbox" name="groceries"  
        value="bread"/>Bread  
</label>  
<label>  
  <input type="checkbox" name="groceries"  
        value="eggs"/>Eggs  
</label>
```

Notice order of label/input,  
Label and value are different

Milk  Bread  Eggs

# Multiple Choice Input: **checkbox**

- Discussion: When designing a form, when should a checkbox be **checked** by default?

```
<label>  
  <input type="checkbox" name="groceries"  
        value="milk" checked/>Milk  
</label>
```

# Multiple Choice Input: **checkbox**

- Discussion: When designing a form, when should a checkbox be **checked** by default?
  - Common value that people will always want
- Have you ever seen a checkbox turned on when you think it shouldn't be?

```
<label>  
  <input type="checkbox" name="groceries"  
        value="milk" checked/>Milk  
</label>
```



# When You Click Submit, What Happens?

- User checks whichever boxes they want
- All of the checked inputs are submitted as parameters
- For example, if milk and bread were checked, the URL would look like

`http://example.org/cart?groceries=milk&groceries=bread`

➤ Parameter name: groceries

➤ Parameter values: milk, bread

# Multiple Choice Input: **radio**

- *Only one* radio button in a group can be on or pressed
  - Groups of radio buttons are identified with the same **name**

Why is **radio** appropriate?

```
<label> <input type="radio" name="age"
           value="under20" checked="checked" />0-19
</label>
<label> <input type="radio" name="age"
           value="20-35" />20-35
</label> ...
<label> <input type="radio" name="age"
           value="over65" />&gt;65
</label>
```

Same  
name

0-19    20-35 ...    >65

# Alternative for Label

- Use **for** attribute to specify which control you're labeling
  - **for**'s value is the control's **id**

```
<label for="age.under18">0-17</label>  
<input id="age.under18" type="radio"  
      name="age" value="under18"/>
```

Important for usability

# Menus with `<select>`

- Displays large number of options more compactly
- Can emulate radio buttons (only one selection, default) or checkboxes (multiple selections)

<b>name</b>	Name of the data
<b>size</b>	# of items to display
<b>multiple</b>	Allows multiple selections if value is <b>multiple</b>

```
<select name="age">  
  ...  
</select>
```

# option tag

- Value options are in **option** tags, nested inside of **select** tags
- Can preselect an option with **selected** attribute set to “selected”

```
<select name="age">
  <option value="under18"
    selected>0-17</option>
  <option value="18-25">18-25</option>
  ...
  <option value="over65">&gt; 65</option>
</select>
```

What is your age?

- ✓ 0-17
- 18-25
- 26-45
- 46-64
- > 64

# select Tag Example

- Emulating checkboxes

```
<select name="groceries" multiple="multiple">
  <option value="milk">Milk</option>
  <option value="bread">Bread</option>
  <option value="eggs">Eggs</option>
</select>
```

## Alternative Example:

Which types of films do you like to watch?

 Action  
 Comedy  
 Foreign

# Using `select`

- Any advantages or disadvantages to using `select` rather than radio buttons or checkboxes?

# Using `select`

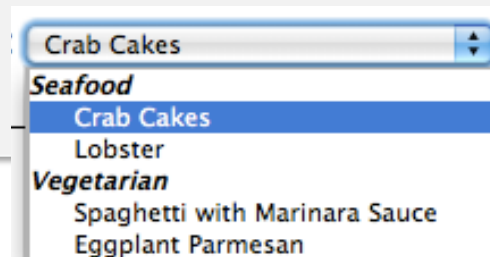
- Any advantages or disadvantages to using `select` rather than radio buttons or checkboxes?
  - `select` takes up less space when only one selection BUT user must click to see all options



# Option Groups: `optgroup`

- Tag used to group options with a label
  - Can also apply a style to label

```
<select name="entree">
  <optgroup label="Seafood">
    <option value="crabcakes">Crab Cakes</option>
    ...
  </optgroup>
  <optgroup label="Vegetarian">
    <option value="spaghetti">Spaghetti</option>
    ...
  </optgroup>
</select>
```

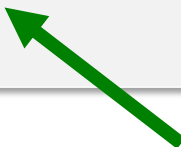


# textarea Tag

- Allows users to input multiple lines of text

<b>name</b>	Name of the data
<b>rows</b>	# of rows for text area
<b>cols</b>	# of characters wide for text area

```
<p>Please provide your yearbook memories:</p>  
<p><textarea name="memories" rows="3"  
cols="40"> (Be brief and concise.)  
</textarea></p>
```



Can't close the tag when opened.  
Needs content, even if empty.

Displayed by default in the  
text area.

# Grouping Input: `fieldset`, `legend`

- **`fieldset`**: groups related **`input`** fields
- **`legend`**: supplies an optional caption

```
<fieldset>
  <legend>Credit cards:</legend>

  <input type="radio" name="creditcards" id="visa" />
  <label for="visa">Visa</label> <br />

  <input type="radio" name="creditcards"
    id="mastercard" />
  <label for="mastercard">MasterCard</label> <br />

  <input type="radio" name="creditcards" id="amex" />
  <label for="amex">American Express</label><br />
</fieldset>
```

# submit and reset Buttons

- type = **submit**

- When clicked, browser sends parameters to the server
- Browser shows server's response

- type = **reset**

- when clicked, browser changes the controls back to their original state

```
<input type="submit" value="Submit Query"/>  
<input type="reset" value="Reset"/>
```

# Styling Forms with CSS Attribute Selectors

```
element[attribute=value] {  
    properties; ...  
}
```

```
input[type="text"] {  
    color: blue;  
    font-style: italic;  
    margin-bottom: 2em;  
}
```



*example text*

- CSS attribute selector affects an element only if it has the given attribute set to the given value
- Often used with forms because **input** element represents many different controls

# Buttons: <button>

- Button's text appears inside **button** tag

```
<button>  
My Button  
</button>
```

```
<button>  
  
</button>
```

# Comparing Buttons

## Input type="submit"

- No content in the element
- Submits a form

## Buttons

- Allows content in the element (e.g., text or an image)
- Can do other actions by setting its `onClick` event handler (typically JavaScript)

# More `<input>` fields

- Date
- Email
- Time
- File
- Color
- Hidden
- ...

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input>



# Considerations for Choosing Controls

How easy is it for the user to enter the input?

What is the range of possible values?

How many possible values are there?

How much flexibility does the user have?

- How much *should* they have?

Should the data be hidden in the browser?

# What **input** would you use?

Input Data	Input Type
Sensitive data	
Number of items to purchase	
Abstract for a paper	
Title for a song	
Household Income Bracket	

# Usability Considerations in Forms

- Input
  - Use appropriate types
  - Use mouse rather than type
    - Less error-prone
  - Use **Label** for inputs
- Checking options by default
  - Easier on user, but only when user wants them

# TODO

- Lab 3: HTML Forms
  - Due tonight at 11:59 p.m.
- Reading for Canvas discussion
- Prep for tomorrow:
  - If you want to use your own laptop, install Eclipse for Enterprise development
    - See course schedule page
  - Review Java: Java for Python Programmers textbook is linked on the schedule page