We are looking at slides from *HTML and CSS: Design and Build Websites* by Jon Duckett.

Slides are posted on the class website, protected by a password written on the (virtual) board.

Take roll for second and last time.

Note: Exams are open-books, no cell phone or email use, proctored by Zoom, with private recording.

If you must take a phone call during an exam, tell the proctor what you’re doing.

---

**HEADINGS**

- `<h1>This is a Main Heading</h1>`
- `<h2>This is a level 2 heading</h2>`
- `<h3>This is a level 3 heading</h3>`
- `<h4>This is a level 4 heading</h4>`
- `<h5>This is a level 5 heading</h5>`
- `<h6>This is a level 6 heading</h6>`

---

**PARAGRAPHS**

- `<p>A paragraph consists of one or more sentences that form a self-contained unit of discourse. The start of a paragraph is indicated by a new line.</p>`

- `<p>Text is easier to understand when it is split up into units of text. For example, a book may have chapters. Chapters can have subheadings. Under each heading will be one or more paragraphs.</p>`
A paragraph consists of one or more sentences that form a self-contained unit of discourse. The start of a paragraph is indicated by a new line.

Text is easier to understand when it is split up into units of text. For example, a book may have chapters. Chapters can have subheadings. Under each heading there will be one or more paragraphs.

The moon is drifting away from the earth.

The Earth gets one hundred tons heavier every day due to falling space dust.

Note: <br> is the usual way to write the line break.
<p><strong>Beware</strong> pickpockets operate in this area.</p>

I think Ivy was the first.

I think <em>Ivy</em> was the first.

I think Ivy was the <em>first</em>. 

<p>STRONG & EMPHASIS</p>
LINKING TO OTHER SITES

```html
<a href="http://www.empireonline.com">Empire</a>
```

RESULT

Empire

---

LINKING TO OTHER PAGES ON THE SAME SITE

```html
<a href="index.html">Home</a>
<a href="about.html">About</a>
<a href="movies.html">Movies</a>
<a href="contact.html">Contact</a>
```

RESULT

- Home
- About
- Movies
- Contact

---

RELATIVE URLS

- examplearts
  - index.html
  - movies
    - index.html
    - listings.html
    - reviews.html
  - cinema
    - index.html
    - listings.html
    - reviews.html
  - dvd
    - index.html
    - listings.html
    - reviews.html
  - music
    - index.html
    - listings.html
    - reviews.html
  - theater
    - index.html
    - listings.html
    - reviews.html

RELATIVE URLS

SAME

- reviews.html

---
RELATIVE URLs

SAME
reviews.html

CHILD
music/index.html

GRANDCHILD
movies/dvd/index.html

PARENT
../index.html

RELATIVE URLS

SAME
reviews.html

CHILD
music/index.html

GRANDCHILD
movies/dvd/index.html

PARENT
../../index.html

EMAIL LINKS

<a href="mailto:jon@example.org">Email Jon</a>

HTML

EMAIL LINKS

<a href="mailto:jon@example.org">Email Jon</a>

HTML
Film-Making Terms

Arc Shot
A shot in which the subject is photographed by an enclosing or moving camera

Interlude
A brief, intervening film scene or sequence, not specifically tied to the plot, that appears within a film

Prologue
A speech, preface, introduction, or brief scene preceding the main action or plot of a film; contrast to epilogue

Interlude
A brief, intervening film scene or sequence, not specifically tied to the plot, that appears within a film

Prologue
A speech, preface, introduction, or brief scene preceding the main action or plot of a film; contrast to epilogue

Film-Making Terms

Arc Shot
A shot in which the subject is photographed by an enclosing or moving camera

Interlude
A brief, intervening film scene or sequence, not specifically tied to the plot, that appears within a film

Prologue
A speech, preface, introduction, or brief scene preceding the main action or plot of a film; contrast to epilogue
CHOOSING IMAGES FOR YOUR SITE

Images can set the tone for a site in less time than it takes to read a description.

Images are subject to copyright but there are stock photography sites where you can buy them.
CHOOSING IMAGES FOR YOUR SITE

If a page shows several photos of products or members of a team, keep them consistent.

STORING IMAGES ON YOUR SITE

ADDING IMAGES

```html
<img src="images/quokka.jpg" alt="A family of quokka" title="The quokka is an Australian marsupial that is similar in size to the domestic cat" />
```

Book examples: this page online
After right-click (not on image), View Source we can see the HTML for the page

This is using Chrome, but this can be done in any desktop browser.

Chrome on smartphone: need to use URL view source: `http://whatever`

Can carefully edit URL with pencil that shows up with tap on URL.

---

**HEIGHT & WIDTH OF IMAGES**

```html
<img src="images/quokka.jpg"
    alt="A family of quokka"
    title="The quokka is an Australian marsupial that is similar in size to the domestic cat"
    width="600"
    height="450" />
```

---

**Resizing**

- Although browsers will resize images as specified by width and height, it's not a great idea to use it.
- Can end up with distorted or fuzzy images.
- Better to make another right-size image using a graphics editor and use that.

---

**How a page with an image gets displayed**

- In this case, the user selects the page as usual
- The browser does a GET request to the server
- The server sends the HTML page with the `<img>`...
- The browser receives the HTML, parses the `<img>` URL, and issues another GET request for the image data (no user involvement here)
- The server sends the image data back
- The browser shows the completed page to the user
- This means two complete “request cycles” to the server
- We can make a chart showing this communication…

---

**Communications Diagram: page with `<img>`**

(time flows down in diagram)

1. User: requests `.html` page via browser
2. Server: sees GET `/…page.html`, Returns HTML on same connection
3. Browser: parses page, sees `<img>`, requests image data
4. Server: sees GET `/...quokka.jpg`, return its data in same connection
5. User: sees finished page
Chrome can show details on the two requests.

Use right-click, inspect, choose Network tab, reload page.

One page can involve many HTTP requests (from last class).

From https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

WHERE TO PLACE IMAGES IN YOUR CODE

There are around 10,000 living species of birds that inhabit different ecosystems from the Arctic to the Antarctic. Man species undertake long distance annual migrations, and many more perform shorter irregular journeys.

WHERE TO PLACE IMAGES IN YOUR CODE

There are around 10,000 living species of birds that inhabit different ecosystems from the Arctic to the Antarctic. Many species undertake long distance annual migrations, and many more perform shorter irregular journeys.
<img> is “inline”

- Now we look at cases where the <img> element lies inside the <p> element.
- <p> is a block element, starts a new line
- <img> Image is an inline element, keeps going on the current line
- Other block elements: <h1>, <ul>, <li>, ...
- Other inline elements: <em>, <a>, <strong>

WHERE TO PLACE IMAGES IN YOUR CODE

<p>There are around 10,000 living species of birds that inhabit different ecosystems from the Arctic to the Antarctic. Many species undertake long distance annual migrations, and many more perform shorter irregular journeys.</p>

“OLD CODE”

- In the next slide and in Duckett, “OLD CODE” means pre-HTML5
- In HTML5, we use CSS to arrange non-default positioning
- Skip “OLD CODE” for this class, but note it could be useful for working with older HTML.
- We should revisit these examples once we’ve tackled CSS.

OLD CODE: ALIGNING IMAGES HORIZONTALLY (we’re skipping)

<p><img src="images/bird.gif" alt="bird" width="100" height="100" align="left" /> There are around 10,000 living species of birds that inhabit different ecosystems from the Arctic to the Antarctic. Many species undertake long distance annual migrations, and many more perform shorter irregular journeys.</p>

Similarly skip align="right" inside <img>
Example image

- See [http://htmlandcssbook.com/code-samples/chapter-05/adding-images.html](http://htmlandcssbook.com/code-samples/chapter-05/adding-images.html)

- HTML:
  ```html
  <body>
    <img src="images/quokka.jpg" alt="Quokka (Setonix brachyurus)" />
  </body>
  ```

  - Note how it stays the same size while you resize the page. It shows in 600x450 screen pixels.
  - On my laptop, 600 px width = 50% of screen width
  - Right-click on image in Chrome, Inspect, to see its dimensions.

Pixels

- Each digital image has a grid of pixels, written as width x height, for example 600x450
- Each screen has “resolution” width x height pixels
  - My old laptop: 1280x800, new one 1920x1080
  - My desktop (20” "two-page" monitor): 1920x1080
  - iPhone 5: 640x1136, iPhone X: 1125x2436, iPhone XS: 1125x2436
  - Samsung S7, Note 9: 1440x2960, ... S9: 1440x2960
  - Many old phones: 854x480
- We can expect about 1000 pixels across to work with
- In particular, the 600x450 image fits on any of these
- See pp. 377-378 for more examples.

Image Dimensions

- Create each image the same width and height as you would like it to appear on your website, using total width = 1000 px.
- Check it on various devices.
- More on this later.

HTML5: Figure & Figure Caption (HTML5)

```html
<figure>
  <img src="images/otters.gif" alt="Photograph of two sea otters floating in the water" />
  <br />
  <figcaption>
    Sea otters hold hands when they sleep so that they don’t drift away from each other.
  </figcaption>
</figure>
```

HTML5: Figure & Figure Caption

```html
<figure>
  <img src="images/otters.gif" alt="Photograph of two sea otters floating in the water" />
  <br />
  <figcaption>
    Sea otters hold hands when they sleep so that they don’t drift away from each other.
  </figcaption>
</figure>
```
HTML5: FIGURE & FIGURE CAPTION

```html
<figure>
  <img src="images/otters.gif" alt="Photograph of two sea otters floating in the water" />
  <br />
  <figcaption>Sea otters hold hands when they sleep so that they don’t drift away from each other.</figcaption>
</figure>
```

WHAT’S A TABLE?

```
<table>
  <tr>
    <td>15</td>
    <td>15</td>
    <td>30</td>
  </tr>
  <tr>
    <td>45</td>
    <td>60</td>
    <td>90</td>
  </tr>
</table>
```
### Saturday Sunday

**Tickets sold:** 120 135
WHY FORMS?

FORM CONTROLS

ADDING TEXT:
- Text input (single-line)
- Password input
- Text area (multi-line)

MAKING CHOICES:
- Radio buttons
  - Rock
  - Pop
  - Jazz
- Checkboxes
  - [ ] Class A
  - [ ] Class B
  - [ ] Class C
- Drop-down boxes

SUBMITTING FORMS:
- Submit buttons
- Image buttons

UPLOADING FILES:
- File upload

HOW FORMS WORK

1: User fills in form and presses button to submit info to server
### How Forms Work

#### 1: Form

**Vote For Your Favorite Jazz Musician Of All Time**

- **Username:** Ivy
- **I vote for:**
  - Ella Fitzgerald
  - Herbie Hancock
  - John Coltrane
  - Miles Davis
  - Thelonius Monk

**Submit**

#### 2: Name of each form control sent with value user entered

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Ivy</td>
</tr>
</tbody>
</table>

#### 3: Server processes information using programming language

Thank you, Ivy!

You voted for Herbie Hancock.

#### 4: Server creates new page to send back to the browser based on info received

**username=Ivy**

<table>
<thead>
<tr>
<th>NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>Ivy</td>
</tr>
</tbody>
</table>
Name-value pairs sent to server

- **HTTP GET:** in URL query string
  
  GET /webapp/program?username=Ivy HTTP/1.0
  … headers

- **HTTP POST:** in the body of the request:
  
  POST /webapp/program
  … headers
  … encoded name-value pairs

Note: HTTP POST is the usual way to send in form data, as we will see.

**Communications Diagram: simple form handling**
(time flows down in diagram)

1. User: requests form page via browser

2. Server: sees GET /...form.html, returns form.html on same connection

3. User: fills in form
   
   Browser: puts user input into params in POST request

4. Server: sees POST /.../doit.php
   
   Get user input from params, do requested action, compose response, return it in same connection.

5. User: sees response

**FORM STRUCTURE**

```html
<form action="http://example.com/join.php" method="get">
  This is where the form controls will appear.
</form>
```

**TEXT INPUT**

```html
<form action="http://example.com/join.php">
  <input type="text" name="username" size="15" maxlength="30" />
</form>
```

**PASSWORD**

```html
<p>Username:
  <input type="text" name="username" size="15" maxlength="30" />
</p>

<p>Password:
  <input type="password" name="password" size="15" maxlength="30" />
</p>
```
Your favorite genre:

- [ ] Rock
- [ ] Pop
- [ ] Jazz

Your favorite music service:

- [ ] iTunes
- [ ] Last.fm
- [ ] Spotify
DROP DOWN LIST BOX

```
<select name="devices">
  <option value="iPod" selected="selected">iPod</option>
  <option value="radio">Radio</option>
  <option value="PC">Computer</option>
</select>
```

MULTIPLE SELECT BOX, select with additional attribute (red)

```
<select name="devices" size="4" multiple="multiple">
  <option value="guitar" selected="selected">Guitar</option>
  <option value="drums">Drums</option>
  <option value="keys" selected="selected">Keyboard</option>
  <option value="bass">Bass</option>
</select>
```

FILE INPUT BOX: we won't need this.

```
<form action="http://eg.com/upload.php" method="post">
  <p>Upload your song in MP3 format:</p>
  <input type="file" name="user-song" />
  <input type="submit" value="upload" />
</form>
```
Upload your song in MP3 format:

<form action="http://eg.com/email.php">
  <p>Subscribe to our email list:</p>
  <input type="text" name="email" />
  <input type="submit" value="Subscribe" />
</form>

SUBMIT BUTTON

Substitute to our email list:

<form action="http://eg.com/email.php">
  <p>Subscribe to our email list:</p>
  <input type="text" name="email" />
  <input type="image" src="images/subscribe.jpg" width="100" height="20" />
</form>

IMAGE BUTTON (does submit)

Substitute to our email list:

<form action="http://eg.com/add.php">
  <button>
    <img src="images/add.gif" alt="add" width="10" height="20" />
  </button>
</form>

BUTTONS: Don't use <button> inside a form!

This submits the form, like <input type="submit"> or <input type="image">. Tip from www.w3schools.com: If you use the <button> element in an HTML form, different browsers may submit different values. Use <input> to create buttons in an HTML form.

• Let's follow this rule.
HIDDEN FORM CONTROLS
And fix pg. 162: replace <button> with <input>

```html
<form action="http://eg.com/add.php">
  <button> <img src="images/add.gif" alt="add" width="10" height="20" /></button>
  <input type="image" src="images/add.gif" alt="add" width="10" height="20" />
  <input type="hidden" name="bookmark" value="lyrics" />
</form>
```

RESULT
(Plus hidden effect: bookmark=lyrics goes back to server)

Communications Diagram: form handling
Case of form having hidden control “bookmark”

1. User: requests form page via browser
2. Server: sees GET /...form.html, returns form.html on same connection
3. User: clicks button
   Browser: puts bookmark=lyrics in GET request
4. Server: sees
   GET /add.php?bookmark=lyrics
   Gets bookmark value, does requested action, composes response, returns it in same connection.
5. User: sees response

LABELLING FORM CONTROLS: two ways...

```html
<form action="http://eg.com/email.php">
  <label>
    Age:
    <input type="text" name="Age" />
  </label>
  Gender:
  <input id="female" type="radio" name="gender" value="f" />
  <label for="female">Female</label>
  <input id="male" type="radio" name="gender" value="m" />
  <label for="male">Male</label>
</form>
```

GROUPING FORM ELEMENTS

```html
<fieldset>
  <legend>Contact details</legend>
  <label>Email:<br />
    <input type="text" name="email"></label>
<br />
  <label>Mobile:<br />
    <input type="text" name="mobile"></label>
<br />
  <label>Telephone:<br />
    <input type="text" name="tel"></label>
  </fieldset>
```
HTML5: FORM VALIDATION
Add required attribute

```html
<label for="username">Username:</label>
<input type="text" name="username" required="required" />

<label for="password">Password:</label>
<input type="password" name="password" required="required" />

<input type="submit" value="Submit" />
```

HTML5: DATE INPUT

```html
<label for="date">Departure date:</label>
<input type="date" name="depart" id="date" />

<input type="submit" value="Submit" />
```

HTML5: EMAIL & URL INPUT

```html
<input type="email" name="email" />
<input type="url" name="website" />
```
Whenever you want to collect information from visitors you will need a form, which lives inside a `<form>` element. Note: this is not strictly true, since it is possible to use `<input>` outside a form along with scripts that specify resulting behavior on user actions. It is true for simple PHP websites (they have no scripts).

Information from a form is sent in name/value pairs.

Each form control is given a name, and the text the user types in or the values of the options they select are sent to the server.
SUMMARY

HTML5 introduces new form elements which make it easier for visitors to fill in forms.