12 present

- [ ] Comments in html, css and js are:
  <!-- comment -->
  /* like 'C' or Java */
  /* like 'C' or Java */ and // like bash to end of line

- [ ] Interesting lick:
  <button onclick='alert(eval(prompt("text")))' ... >

- [ ] Stacking and testing libraries
  See:
  https://stackoverflow.com/questions/950087/how-do-i-include-a-javascript-file-in-another-javascript-file

  This isn't super clean, there are several mechanisms,
  and they are all incomplete. They include:
  HTML: <script src='lib.js'></script>
  lib.mjs:
  ```
  export function hello(){} // module.mjs
  import { hello } from 'module' // main.mjs
  ```
  Dynamic Imports in Browsers
  Node.js require
  jQuery (nice):
  ```
  $.getScript("my_lovely_script.js", function()
  {
      alert("Script loaded but not necessarily executed.");
  });
  ```

- [ ] paragraphs (p), divisions (divs), spans and headers (h1-h4) can all have on onclick() function that activates when they are clicked, or mouse-overed and so forth. Huge!!

- [ ] Elements with ID's can be written to and read from

- [ ] <scripts> in <head> are deferred
  <scripts> at end of <body> are run immediately
  <scripts> at beginning of <body> aren't as predictable

- [ ] Factoring the {html, css, js} trifecta into modules.
  I did this and created utilities for managing them.
  This greatly eased deploying homework assignments for the course.
  This was done by the following simplifications:
  1) used zip to archive {html, css, js} triples
  2) created 'gat' utility to create the triple
  3) created 'ungat' utility to unpack the triple
  4) created 'pack' utility to 'gat' all triples
  5) created 'unpack' utility to 'ungat' all triples
  6) created 'find,js' to unpack and grep all triples for a construct
     or commonly used idiom to save having to recreate it.

- [ ] Try writing to class versus writing to id
  Try writing to tag vs writing to id