Comp 125 - Visual Information Processing

Spring Semester 2019 - Week 6 - Monday

Dr Nick Hayward

Fun exercise - using objects

- create an object or objects with information about an archive
  - *include name and location of the archive*
- use a combination of arrays and objects to store information about books in the archive - minimum five books
  - *include author's name, book title, date of publication, number of pages...*
- output to the document all of the names of the books in the archive
  - *output to the document all information for at least one book in the archive*

Output answers to the document with link breaks between results.

HTML & JavaScript - create a game

- common first game to create with many languages is **Hangman**
  - *a word-guessing game*
  - *one player picks a secret word*
  - *the second player tries to guess*
  - *a word is chosen with a known length, e.g.* ***WALDZELL***
  - *8 letters in the word expressed using empty characters*

```
--------
```

- as second player guesses a correct letter
  - *we can add it to the output, e.g.*

```
  L ZE
--------
```

- good test of JavaScript usage and structure
  - *data usage*
    - interaction and input
    - output and updates...

HTML & JavaScript - create a game - basic HTML page

v0.1

- start by creating a basic HTML page for the game
  - *add header for page*
    - text input for player guess
    - render hangman data to document

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <!-- gaming title -->
    <title>Hangman Game</title>
  </head>
  <body>
    <header>
      <h3>Waldzell Gaming - Hangman</h3>
    </header>
    <main>
      <section>
        <header>
          <h4>play a game</h4>
        </header>
      </section>
      <section>
        <header>
          <h4>game updates</h4>
        </header>
      </section>
      <aside>
        <!-- add some game instructions... -->
      </aside>
    </main>
  </body>
</html>
```

HTML & JavaScript - create a game - game logic

- JavaScript - game logic includes
  - *player picks a random word for the game*
  - *logic needs to accept a player's guess*
  - *check if guess is a valid letter*
  - *record correct letters chosen by player*
  - *record counter of incorrect letters chosen by player*
  - *output game progress to player*
  - *finish the game*
    - either the player guesses the word correctly
    - or the player guesses incorrectly too many times...

HTML & JavaScript - create a game - add JS file

- create new JavaScript file for game logic
  - *e.g. game.js*
  - *add standard reference to JS file in* `index.html`

```
  <head>
  <meta charset="UTF-8">
  <!-- gaming title -->
  <title>Hangman Game</title>
      <!-- script files -->
      <script src="./assets/js/game.js"></script>
</head>
```

- we'll move this script element later in the development

HTML & JavaScript - create a game - game logic

part 1 - random word

- use JS built-in `Math` object
  - *use `random` method to get value*
  - *round the value down with `floor` method*

```
// random words for game
var gameWords = [
  "dragon",
  "wizard",
  "eagle",
  "hobbit",
  "earth",
  "planets",
  "geography"
];

// pick a random word for a new game
var gameWord = gameWords[Math.floor(Math.random() * gameWords.length)];

// check random word in console
console.log('game word = ' + gameWord);
```

- W3Schools - Math object

HTML & JavaScript - create a game - game logic

part 2 - array for the answers

- create initial empty array for characters in random word
  - *get length of random word*
  - *use string `length` property*
- use `for` loop to add underscore per character
  - *index `i` used to add value to `answers` array*
  - *`lettersToGuess` value decremented*
  - *decrement by 1 for each correctly guessed letter*

```
// define empty array for characters in random word
var answers = [];

// set value for letters to guess from random word
var lettersToGuess = gameWord.length;

// loop through answers array - add underscore for each letter in gameWord
for (var i = 0; i < lettersToGuess; i++) {
    answers[i] = "_";
}
```

HTML - better markup

- web standards are crucial for understanding markup
  - *markup that goes beyond mere presentation*
- improved usage and structure, accessibility, integration...
- with standards, maintenance and extensibility becomes easier
- improved page structure and styling
  - *helps web designers and developers update and augment our code*
- poor markup usage
  - *to achieve a consideration and rendering of pure design*
  - *e.g. nesting tables many levels deep*
  - *adding images and padding blocks for positioning...*
- support for web standards continues to grow in popular browsers
- gives developers option to combine markup and styling
  - *HTML with CSS to achieve greater standards-compliant design*

HTML - markup and standards

- many benefits of understanding and using web standards, e.g.
- *reduced markup*
  - *less code, faster page loading*
  - *less code, greater server capacity, less bandwidth requirements...*
- *separation of concerns*
  - *content, structure, and presentation separated as needed*
  - *CSS used to manage site's design and rendering*
  - *quick and easy to update efficiently*
- *accessibility improvements*
  - *web standards increase no. of supported browsers & technologies...*
- *ongoing compatibility*
  - *web standards help improve chances of compatibility in the future...*

HTML - better structure

- consider *semantic* or *structured* markup
  - *within the context of app usage and domain requirements*
- trying to impart a sense of underlying meaning with markup
  - *correct elements for document markup*
- for a list
  - *use correct list group with list items - e.g.* `ul`, `li`...
- for a table
  - *consider table for data purposes*
  - *structure table & then consider presentation...*
- *semantic* markup helps create *separation of concerns*
  - *separate content and presentation*
  - *improves comprehension and usage*

Semantic HTML - intro

- importance of web standards
  - *and their application to HTML markup and documents*
- standards help drive a consideration of markup, e.g. HTML
  - *usage for what they mean*
  - *not simply how they will look...*
- semantic instead of purely presentational perspective
  - *introduction of meaning and value to the document*
- when pages are processed
  - *impart structure and meaning beyond mere presentation*
- a core consideration for usage of markup languages
- issues persist with HTML element usage
  - *e.g. inline elements such as `<b>` and `<i>`*

Semantic HTML - a reason to care

- Semantic HTML - opportunity to convey meaning with your markup
  - *meaning may be explicit due to the containing element*
  - *implicit due to a structured grouping of elements*
- markup makes it explicit to the browser
  - *underlying meaning of a page and its content*
- notion of meaning and clarity also conveyed to search engines
  - *fidelity with query and result...*
- semantic elements provide information beyond page rendering and design
- use semantic markup correctly
  - *create more specific references for styling*
  - *greater chance of rendering information correctly*

HTML & JavaScript - create a game - HTML

update game page

- update HTML for game
  - *add `id` attributes with unique reference values*
    - values act as unique selectors for elements

```
<section id="play">
    <header>
        <h3>play a game</h3>
    </header>
</section>
<section id="updates">
    <header>
        <h3>game updates</h3>
    </header>
    <p id="wordStatus"></p>
</section>
```

- add unique `id` references for each section

HTML & JavaScript - create a game - game update

output start of game

- output game word to player in the `updates` section of HTML

```
// output game progress to player
var lettersOutput = answers.join(" "); // create string from answers array
document.getElementById('wordStatus').innerHTML = 'guess word: ' + lettersOutput;
```

- use `join()` method to create string from `answers` array
  - *use paragraph with ID `wordStatus`*

HTML & JavaScript - create a game - user input

add input for letter guess

- add a text input field
  - *allows player to guess a letter in the random word*
  - *add useful attributes to `input`*
  - *`placeholder` - sets default text for input (helper text)*
    - `maxlength` - sets maximum characters permitted in input

```
<section id="play">
    <header>
    <h3>play a game</h3>
    </header>
    <form>
      <input name="guess" placeholder="guess a letter" type="text" maxlength="1"
    </form>
</section>
```

- W3Schools - HTML Form Attributes

HTML & JavaScript - create a game - guess a letter

add button to make a guess

- add a simple button
  - *player may submit letter in input field as their **guess***

```
<form id="">
    <input name="guess" placeholder="guess a letter" type="text" maxlength="1" id
    <button type="button" id="guessBtn">guess</button>
</form>
```

- W3Schools - HTML Form Elements

Semantic HTML - example usage

```
<!-- incorrect element chosen -->
<div id="code">
document.addEventListener('click', function () {
  console.log('Click received...');
});
</div>
```

```
<!-- correct element chosen -->
<code>
document.addEventListener('click', function () {
  console.log('Click received...');
});
</code>
```

- semantic example usage