Comp 125 - Visual Information Processing

Spring Semester 2019 - Week 6 - Wednesday

Dr Nick Hayward

HTML & JavaScript - create a game - guess a letter

get letter from input field

- add **event listener** to the `guess` button
  - *listener is attached to the `guess` button*
  - *logic is executed each time a player clicks on this button*
  - *get the value of the letter entered by the player*
  - *value of input field for `guess a letter`*
    - log value to console for initial testing

```
// listen for user click on `guess` button
var guessBtn = document.getElementById('guessBtn');
guessBtn.addEventListener('click', function() {
    // get letter from input field
    var letter = document.getElementById('guess').value;
    console.log('letter = ' + letter);
    // check letter against
}, false);
```

HTML & JavaScript - create a game - HTML

update game page

- update HTML for game
  - *add <p> for letter guess by player*

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

- Hangman Game - v0.1

HTML & JavaScript - create a game - guess a letter

output letter from input field

- use **guess** letter from input field
  - *output value to HTML for player*

```
// listen for user click on `guess` button
var guessBtn = document.getElementById('guessBtn');
guessBtn.addEventListener('click', function() {
    // get letter from input field
    var letter = document.getElementById('guess').value;
    // output guess letter
    console.log('letter = ' + letter);
    document.getElementById('guessLetter').innerHTML = `guess letter: ` + letter;
    // check letter against
}, false);
```

- get element with ID `guessLetter`
  - *set HTML to player's current **guess** letter*

HTML & JavaScript - create a game - check guess letter

check letter against game word - part 1

- use `includes()` method with `gameWord` string
  - *initial check that guess letter is in game word*

```
// check letter against  game word
if (gameWord.includes(letter) === true) {
  console.log('letter has been found...');
}   else {
    console.log('letter not found...');
    document.getElementById('guessLetter').innerHTML = 'letter not found - please
}
```

- log results of conditional statement to console
  - *update player if guess letter not found in game word*

HTML & JavaScript - create a game - check guess letter

check letter against game word - part 2

- loop through game word
  - *check guess letter against each character in game word*
  - *e.g.* `letter` *in* `gameWord`
  - *if guess letter found in game word*
  - *add guess letter to matching index position in* `answers` *array*
  - *update string from* `answer` *array*
  - *output update guess word for player*

```
for (i = 0; i < gameWord.length; i++) {
  if (gameWord[i] === letter) {
    console.log('letter = index ' + i);
    answers[i] = letter;
    // update game progress to player
    var lettersOutput = answers.join(" "); // create string from answers array
    document.getElementById('wordStatus').innerHTML = 'guess word: ' + lettersOut
    }
}
```

HTML & JavaScript - create a game - check guess letter

check letter against game word - part 3

```
// select guess button in document
var guessBtn = document.getElementById('guessBtn');

// listen for user click on `guess` button
guessBtn.addEventListener('click', function() {
  // get letter from input field
  var letter = document.getElementById('guess').value;
  // output guess letter
  console.log('letter = ' + letter);
  document.getElementById('guessLetter').innerHTML = 'guess letter: ' + letter;
  // check letter against game word
  if (gameWord.includes(letter) === true) {
    console.log('letter has been found...');
    for (i = 0; i < gameWord.length; i++) {
      if (gameWord[i] === letter) {
        console.log('letter = index ' + i);
        answers[i] = letter;
        // update game progress to player
        var lettersOutput = answers.join(" "); // create string from answers arra
        document.getElementById('wordStatus').innerHTML = 'guess word: ' + letter
      }
    }
  } else {
  console.log('letter not found...');
  document.getElementById('guessLetter').innerHTML = 'letter not found - please t
  }
}, false);
```

- Hangman Game - v0.2

Semantic HTML - correct usage

- need to ensure elements convey their correct meaning
  - *i.e. the meaning expected for the contained content*
- e.g. often see the following elements mis-used and applied incorrectly for markup,
  - *<p> - paragraphs*
  - *<ul - unordered list*
  - *<h1> to <h6> - headings*
  - *<blockquote> - blockquote*
- using <blockquote> to simply help indent text
  - *instead of CSS margins...*
- or the perennial mis-use of a <p>
  - *simply add extra space between elements*

```
<p> <p>
```

HTML - structure & validation - example

Using lists correctly...

```
<li>nice</li>
<li>cannes</li>
<li>menton</li>
```

- list markup looks OK
  - *still fails validation for an obvious reason*
  - *missing structural grouping for list items*
  - *not valid markup...*
- semantics of the overall list are missing
- example - basic list items

HTML - a semantic point of view

```
<ul>
  <li>nice</li>
  <li>cannes</li>
  <li>menton</li>
</ul>
```

- from the perspective of semantics
  - *meant to act as a group of items that belong together*
- denote such groupings with correct semantic markup
- structuring items to clearly denote their meaning and purpose
- consider global attributes
  - *https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes*
- example - basic group

HTML - benefits of structure & validation

- define and create a meaningful structure for required markup
  - *improves usage and flexibility as project develops*
  - *provides extensible structure for project*
- for example, benefits include
  - *helps increase ease of CSS styling*
  - *creates properly structured documents*
  - *improves general management of updates to markup*
  - ...
- easier to understand and easier to maintain and update
- structured, valid markup aids in repurposing data
  - *into various representations of information*

HTML - benefits of structure & validation - example 1

e.g. a standard list

```
<ul>
  <li>nice</li>
  <li>cannes</li>
  <li>menton</li>
  <li>antibes</li>
  <li>grasse</li>
</ul>
```

- example - basic group style

HTML - benefits of structure & validation - example 2

e.g. lists for navigation, menus, tabs...

```
<ul id="menutabs">
  <li><a href="nice">nice</a></li>
  <li><a href="cannes">cannes</a></li>
  <li><a href="menton">menton</a></li>
  <li><a href="antibes">antibes</a></li>
  <li><a href="grasse">grasse</a></li>
</ul>
```

- example - basic menu tabs