Comp 125 - Visual Information Processing

Spring Semester 2018 - week 6 - friday

Dr Nick Hayward

Fun exercise - using HTML and JavaScript

- create a HTML and JavaScript based **Random Greeting Generator**
  - *create a HTML file for the application*
  - *create a JavaScript file for the application's JS code and logic*
  - *add a `<script>` reference to the JS file in the HTML, and load the JS*
- HTML file should include the following minimum markup,
  - *metadata in the `<head>` element - e.g. `title`...*
  - *application header and heading*
  - *brief description of application and usage*
  - *form with input and submit button - allows a user to enter their **name** &c.*
  - *main content for rendering name, greeting, and any other necessary output*
  - *footer with details about the developer - e.g. copyright, year, name &c.*
- JavaScript file should include the following minimum logic,
  - *event listener for user click on form submit button*
  - *get text value from form's input text field*
  - *generate random **greeting message** with concatenated user's input text*
  - *render greeting message to HTML after user has clicked submit button*

Share your working application, including completed HTML file and JavaScript file, with the course TA, Catherine.

Submission options include,

- send files attached to private message on Slack
  - *attach files to an email to TA*
  - *share repository on GitHub*
  - ...

HTML & JavaScript - create a game - restart game

reset game and load new game word

- need to reset the game after **GAME OVER**
  - *player wins or loses...*
- game requires reloading, resetting of variables, data structures...
  - *might use simple browser refresh*
  - *better option is to dynamically reset game logic*
- need to abstract code to **functions**...

HTML & JavaScript - create a game

work left to complete

- code is **too** verbose
- code needs **abstraction**
- need to introduce **functions** for better code structure and reuse
- **reset** option necessary for **GAME OVER**
- hangman figure needs to be drawn to HTML document
- small updates to usability
  - *clear letter in input field after `guess` button pressed*
  - *add event listener for **return** key press in input field*
  - *add autofocus to input field*

HTML & JavaScript - create a game - quick updates

update usability on input field

- update event listener for mouse click on `guess` button
- reset value for input field after click event
  - *use empty string to clear input field*
  - *placeholder text will then be shown in input field*

```
// reset input field
document.getElementById('guess').value = "";
```

- reset focus on input field after click event

```
// reset focus on input field
document.getElementById('guess').focus();
```

JavaScript - functions - intro

- game code needs **LOTS** of abstraction and refactoring
- functions are a great way to help such abstraction and reuse
- a **function** is a common and useful option for grouping code
  - *organise for reuse within an application*
- reuse of functions also helps provide better abstraction of logic
- group and store functionality in JS functions
  - *use repeatedly by calling the same function*
- functions also help us organise our code and application logic
  - *providing better structure and design to our code*
- functions help us test our application code more easily
  - *creating manageable chunks of code and logic*
- we may also define accepted parameters for a functio
  - *enabling customisation and broader usage of contained code and logic*
- return values for a given function may be customised
  - *relative to passed arguments as we call a function*

JavaScript - functions - basic structure

- basic structure for function syntax

```
function () {
    ...code to excute...
}
```

- we can extend this syntax
  - *add a **name** for the function*
  - *define accepted **parameter** (or parameters)*
  - *use and return code from a function...*

JavaScript - functions - basic usage - part 1

define function with name and parameter

- add a custom name for a function
  - *this function will log a string to the console...*

```
function sayHello () {
    console.log('Hello...');
}
```

- execute this code by calling the function's name
  - *add parentheses to denote name as function*

```
sayHello();
```

JS Functions - name and call

add a custom function name and call...

JS - function call 1

JavaScript - functions - basic usage - part 2

define function as value of variable

- also assign a function as the value of a named variable

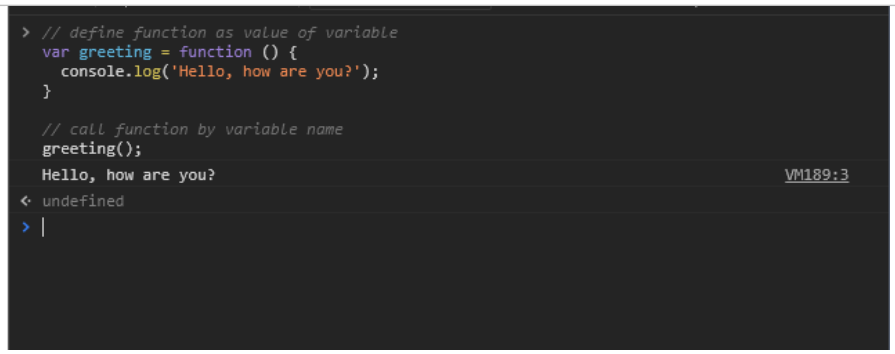
```
var greeting = function () {
    console.log('Hello, how are you?');
};
```

- then call this function using the same pattern

```
greeting();
```

JS Functions - name and call - example 2

add a custom function name and call as value of variable...



JS - function call 2

JavaScript - functions - basic usage - part 3

`return` value

- previous examples included a `return` value of `undefined`
- `return` value is value that a function will actually output
  - *for reuse elsewhere in the application*
- `console.log()` returns its own value
  - *not value for custom function*
- return value will always be `undefined`
  - ***unless*** *we specify a `return` value for the function*

JavaScript - functions - basic usage - part 4

parameters and arguments

- custom functions may also be modified by defining accepted **parameters**
  - *parameter values may be used in the executed logic*
- parameters allow a developer to pass values into the function
  - *may be used to modify the logic and executed code*
- parameters are always defined between a function's parentheses
- as we call the function, we pass the required values as **arguments**
  - *also specified between the parentheses for the function call*

JavaScript - functions - basic usage - part 5

using parameters and arguments - example

- structure for a function with parameter

```
function (parameter) {
    // test output of parameter
    console.log("function parameter = " + parameter);
}
```

- example usage might be as follows

```
function sayHello(name) {
    // output greeting to person
    console.log('Hello' + name + ', how are you?');
}
```

- then call this function
  - *passing an argument for the required function parameter*

```
sayHello('Amelia');
```

JS Functions - parameters and arguments - example

add a custom function with a parameter, and call function with passed argument...

JS - function call 3

References

- W3Schools
- JS - conditionals
- JS - For loop
- JS - functions