Mark Gutierrez 02/20/2020

Homework part 3

2. Describing what the addEventListener() method is doing:

```
//Add
const addNums = () => {
   let num1 = document.getElementById("num1").value;
   let num2 = document.getElementById("num2").value;
   let sum = parseInt(num1) + parseInt(num2);
   document.getElementById("add_result").innerHTML = sum;
};
//Add event listener
const btnAdd = document.getElementById("add");
btnAdd.addEventListener("click", addNums);
```

I declare a "const" variable named "btnAdd" and assigned "document.getElementById("add")". This variable is now getting the element "button" by the Id of "add." Lastly, I added an "addEventListener" method to "btnAdd"; the "addEventListener()" method can accept up to 3 parameters. I am using the first parameter to do something when the user "clicks" the button. For the second parameter I am referencing the function "addNums." To summarize, I am using the "addEventListener" to run the "addNums" function when the user "clicks" the button.

3. Referenced function vs invoked/called function

a. Reference functions are stored and used later. They are often used with "addEventListener()" method in order to use them when the user does something; like click a button.

   Reference function code example:

```
//Add
const addNums = () => {
        let num1 = document.getElementById("num1").value;
        let num2 = document.getElementById("num2").value;
        let sum = parseInt(num1) + parseInt(num2);

        document.getElementById("add_result").innerHTML = sum;

    };

    //Add event listener

    const btnAdd = document.getElementById("add");

    btnAdd.addEventListener("click", addNums);
```

In this code I am referencing “addNums” via the “addEventListener()” method. “AddNums” is stored and only used when the user clicks the button with the class “add.” In this case the user can change the input and click the button to keep running the function for different addition output.

b. Call/invoke functions are used immediately when the website loads. If you want something to happen now use a call function. A call function uses “()” at the end; like “callMeNow()”.

   Call function code example:

```
    let alertUser = () => {

alert("CALL FUNCTION: You are in Mark's week 3 homework. Enjoy!");

    }

    alertUser();
```

In this call function I am using an alert to give the user a message as soon as the website loads. Without “alertUser()” the function would not run. This function only runs once. If I had 3 “alertUser()” it would run the function 3 times.

4. Named functions vs function expression

The main difference between function declaration and function expression is the function name. A function expression can omit the name of the function and make it into an anonymous function.

Named functions are hoisted like var. They load before any code is executed.

Named function code example:

```
    add(3, 3);
```

```
function add(number1, number2) {
return number1 + number2;
}
console.log(add);
ƒ add(number1, number2) {
return number1 + number2;
}
```

Function expression are loaded only when the interpreter reaches that line.

Function expression code example:

```
subtract(3, 1);
var subtract = function (sub1, sub2) {
return sub1 - sub2;
}
console.log(subtract);
```

This doesn’t work. Subtract is not a function.

5. Arrow functions, named functions, and anonymous functions

a. Arrow functions:

Arrow functions are a compact alternate of a function.

Code example:

```
const addNums = () => {
    let num1 = document.getElementById("num1").value;
    let num2 = document.getElementById("num2").value;
    let sum = parseInt(num1) + parseInt(num2);
    document.getElementById("add_result").innerHTML = sum;
};
```

b. Anonymous functions:

Anonymous functions do not have a name. They are assigned to a variable; also known as a function expression. They are not bound to "this", but rather bound to the parent scope. This type of function has a block scope.

Code example:

```
const addNums = () => {
    let num1 = document.getElementById("num1").value;
    let num2 = document.getElementById("num2").value;
    let sum = parseInt(num1) + parseInt(num2);
    document.getElementById("add_result").innerHTML = sum;
};
```

c. Named functions:

Just like "var" a named function is hoisted into the browser.

Code example:

```
function namedExample() {
alert("Named function");
}
namedExample();
```

6. https://marklegit.github.io/COMD_3663/mark_gutierrez_toggle_circle/

7. window.location.reload()

The "reload()" method reloads the document, just like clicking the reload button on your browser.

8. Local git command

After you create a folder(directory).

a. Use "pwd" to check that you are inside the right folder.
b. "git init" initializes a local git repository on your computer.
c. Once you added files inside, type "git status" on the Terminal and hit return keyboard.
d. When you want to "commit" changes type "git add ." and hit return; if you want to commit all your files and folders
e. You can "commit" individual files by typing "git add index.html"

f. “git commit” can be done in 2 ways, either ‘git commit –m “comment”’ or “git commit”
g. “git remote add” you can push into the remote origin, which is the remote hosting service that host your repository remotely.
h. “git remote add origin remoteurl” this allows you to push local commits for the first time onto remote. Make sure to have a remote repository to push on.
i. To create a remote repository, go to github and click your profile, click Repositories, and click the button new and this will take you to create a new repository.
j. Create a repository by filling repository name. Click the “create repository” button.
k. “git push –u” using this you can push your local repository into remote origin. In the Terminal type “git push –u" and click return.
l. You will get a master branch, which is a default branch when you initialize.
m. “cd” can change the directory. Type “cd name_of_folder” in the Terminal to do so.

9. Local Git repository

The local repository is the .git/ subdirectory inside the working directory

10. What else signals initialized Git?

The master branch.