Siobhan Lynch
COMD6336
Homework 3 Part 2

1)
```
const btnDivide = document.getElementById("divide");
btnDivide.addEventListener("click", divideNums);
```

Const is a way to define a variable, it cannot be reassigned
btnDivide the btn is the object that represents the HTML <button> tag and the the Divide is its name.
= equals, very self explanitory
Document.getElementByID(“divide”) document is an object that represents the HTML document while .getElementByID is the method and (“divide”) are the parameters along with the ID’s name inside it.
; ends that line
btnDivide again is the call to the <button> tag and its name is Divide
.addEventListener is a method that is an built in function in JavaScript which takes the event to listen for, and a second argument to be called whenever the described event gets fired. In this case the (“click”, divideNums) are the events and the divideNums is name of one of the const variables defined earlier in the js code that lets the EventListener know that thats the code it wants to fire on “click”. Oh and “click” and divideNums are parameters within the .addEventListener too.

2) Referenced functions are stored and used later. They are sometimes used with methods. This way they can be later called. Here I’ve referenced the function subtractNums and then I called with the method addEventListener() with in its parameters.
Example:
```
const subtractNums = () => {
    const num9 = document.getElementById("num9").value;
    const num10 = document.getElementById("num10").value;
    const subtract = num9 - num10;
    document.getElementById("result5").innerHTML = subtract;
};

const btnSubtract = document.getElementById("subtract");
btnSubtract.addEventListener("click", subtractNums);
```

Invoked functions are functions that execute when they are called. Basically and Invoke/call mean the same thing here. Here I define a function named hello. Then I invoke it by typing it into the console and it prints, “how are you”.
Example:
```
function hello(){
```

```
  console.log('how are you');
}

hello();
// how are you
```

3) A Named function is a functions that has been named after it. The only way to name a functions is to type in anything after the word (function) but before you put the parentheses which represent the parameters. This helps with referencing it later. If it is not named, it is an anonymous function.
Example:

```
function iHaveAName(name) {
    return `My name is ${name}!`
}
iHaveAName(`Tim`)
```

Function Expressions is the function keyword being used to define a function inside an expression. You can also define functions using the Function constructor and a function declaration. It is very similar to and has almost the same syntax as a function declaration. The main difference between a function expression and a function declaration is the function name, which can be omitted in function expressions to create anonymous functions.
Example:

```
console.log(notHoisted) // undefined
//  even though the variable name is hoisted, the definition isn't. so it's undefined.
notHoisted(); // TypeError: notHoisted is not a function

var notHoisted = function() {
   console.log('bar');
};
```

4) (This is a repeated explanation)A Named function is a functions that has been named after it. The only way to name a functions is to type in anything after the word (function) but before you put the parentheses which represent the parameters. This helps with referencing it later. If it is not named, it is an anonymous function.
Example:

```
function iHaveAName(name) {
    return `My name is ${name}!`
```

```
}
iHaveAName(`Tim`)
```

Function Expressionfdjsfksjlfs;lfs

An anonymous function is a function that was declared without any named identifier to refer to it. As such, an anonymous function is usually not accessible after its initial creation.
Example:

```
var anon = function() {
  alert('I am anonymous');
}
anon();
//
```

An Arrow function is a compact alternate of a regular function. Arrow function expressions aren’t made to be a good replacement to a method, and they cannot be used as constructors.
Example:

```
const materials = [
  ‘Sugar’,
  ’Spice’,
  ‘Everything’,
  ‘Nice’
];

console.log(materials.map(material => material.length));
// Array [5, 5, 10, 4]
```

5) https://github.com/catwoman1199/homework/tree/master/Siobhan-Lynch-wk-three-hw/Siobhan-Lynch-toggle-circle

6)window.location.reload()

window
.location
.reload() method reloads the document

7)
create a directory aka a folder.
- Type “pwd” to check where you are and their you aren’t in the right

folder us the ~cd command to navigate yourself there. “cd” kinda means change directory.
- “git init” makes a local git repository on your computer.
- add some files inside the folder, then type “git status” into the Terminal and hit enter.
- “commit” changes and then type “git add .” and hit enter. This is if you want to commit all your files and folders. You can “commit” individual files by typing “git add index.html”
- There are 2 ways you can “git commit”. One, ‘git commit –m “comment”’ or “git commit”
- “git remote add” pushes to the remote origin. Origin is the remote hosting service that hosts your repository remotely.
- Create a remote repositoryby going to your github and click your profile, click Repositories, click “new” which will take you to create a new repository. Give it a name and you’re done with that.
- “git remote add origin remoteurl” this pushes your local commits to remote, But you need a remote repository to push onto first!!
- “git push –u” this pushes your local repository into remote origin. In the Terminal type “git push –u" and click enter.
- You’ll see you have a master branch, a default branch when initialized by you.
- Use “cd” to change directories by type “cd name_of_folder” in the Terminal.

8) Local repositories are the .git/ subdirectory inside the working directory

9) The master branch can also signals an initialized Git