**Reporte de prácticas 1**

Fernando Julián Verján Vargas 2E

Introducción

Antes de empezar con la práctica como tal, se nos presentaron los equipos en los que trabajaríamos, ya que es un sistema operativo Mac Os y muchos de nosotros no teníamos claro cómo trabajan estas computadoras.

Iniciamos acomodando nuestra barra de aplicaciones de la parte inferior, mejor conocida como “Muelle”, Eliminamos todas las aplicaciones del muelle, excepto Finder, Safari, Terminal, y Configuraciones.

Una vez más claro todo, ejecutamos dos consolas (o terminales), analizamos ciertos comandos que enlisto a continuación:

Pwd (Muestra la ruta hasta la carpeta actual de trabajo)

Cd (Comando para cambiar la carpeta actual de trabajo, necesita especificar la carpeta a la que se desea acceder)

Mkdir (Crea un directorio nuevo)

Clear (Limpia la consola)

ls –l (Hace una lista del contenido del directorio)

nano \*nombre del archivo\* (abre el archivo solicitado, en caso de no existir lo crea)

Desarrollo

Para este momento ya teníamos una idea clara de cómo funcionaba el sistema operativo, además de conocer algunos comandos para navegar por la terminal, lo siguiente fue ejecutar JSC dándole un nombre con el siguiente comando:

*alias jsc='/System/Library/Frameworks/JavaScriptCore.framework/Versions/Current/Resources/jsc'*

Le dimos un alias al shell JSC, entonces cuando solicitáramos jsc en la terminal, directamente se nos abriría la ruta solicitada.

Con el uso de Nano creamos bastantes programas simples:

-holamundo.js (Imprime la frase “Hola JS”)

-minombre.js (Imprime información mía)

-mishobbies.js(Imprime 3 de mis hobbies)

-triangulo.js (Calcula el área de un triángulo)

-cuadrado.js (Calcula el área de un cuadrado)

-rectangulo.js (Calcula el área de un rectángulo)

-circulo.js (Calcula el área de un círculo)

-trapecio.js (Calcula el área de un trapecio)

También conocimos comandos para el propio jsc, como **quit()** que sirve para salir de jsc y **load()** que usamos para correr los programas que hicimos.

Sin embargo, todos los programas anteriormente mencionados eran muy simples, así que analizamos dos bifurcaciones que se utilizan en la programación, las cuales son **“if”** que realiza una u otra acción según sea el caso, y **“switch”** que funciona de manera similar a if, pero con muchos más casos.

Entonces nos dimos a la tarea de realizar programas un poco más complejos usando **if:**

-esPositivo.js (Te dice si el número que ingresaste es positivo)

-esNegativo.js (Te dice si el número que ingresaste es negativo)

-circuloCircunscrito.js (Calcula el Radio de un circulo circunscrito dentro de un triángulo)

-circuloCircunscribiendo.js (Calcula el Radio de un circulo circunscribiendo un triángulo)

También se nos explicó la ley de la tricotomía, y con esto realizamos un programa que analiza dos números, y te dice si el primero es mayor, si el segundo es mayor, o si son iguales.

Posteriormente creamos otros programas, pero ahora utilizando la bifurcación **switch:**

-switch.js (Pide un número, y te dice que día de la semana es ese número. Ej: 2=Martes)

-meses.js (Funciona igual que el anterior, pero ahora te dice que mes representa ese número)

El último programa realizado fue puntoCirculo.js que te pide un radio para una circunferencia, y dos coordenadas para ubicar un punto dentro del plano cartesiano. Una vez que tiene estos datos te regresa la información de si el punto ubicaste en el plano está dentro del círculo, fuera del circulo o exactamente en la línea de la circunferencia.

Conclusiones

Básicamente utilizamos archivos de texto “nano” para programar en JS, programas bastante básicos pero de los que se puede llegar a ciertos análisis, por ejemplo:

-Los programas más sencillos no necesitaron datos, simplemente fueron impresiones de texto. (Por ejemplo, “Hola JS)

-Para algunos programas lo único que se necesita son datos y fórmulas matemáticas. (Calcular áreas de figuras)

-También utilizamos bifurcaciones que nos ayudaron a tomar “decisiones” según fuera requerido. (Ejemplo: Tricotomía y Días de la semana)

Otro dato curioso es que JavaScript es un lenguaje débilmente tipado, es decir, que no tenemos que asignar un tipo de variable a los datos que usamos, el sistema se da cuenta de que tipo se trata y lo utiliza como tal.