Práctica - MiracleLab

Libro de Laboratorios

April 18, 2023

Acerca de este documento

Este documento se genero a partir de fuentes en LaTeX en https://github.com/bootlin/training-materials.

Correcciones, sugerencias, contribuciones son bienvenidas!

Introdución a Git

Objetivos: familiarizarse con los comandos básicos de Git.

Ejercicio 1

Vamos a crear nuestro primer repositorio de código, para ello:

1. Crearemos una nueva carpeta
2. Crearemos un nuevo archivo `README.md` dentro del mismo
3. Le diremos a git que es un repositorio de código y que tiene que estar al tanto de los cambios
4. Escribiremos dentro del archivo `README.md` una breve presentación
5. Añadiremos el archivo `README.md` al índice de git
6. Consultaremos qué cambios se han producido en el repositorio
7. Añadiremos al final del archivo `README.md` la hora actual
8. Consultaremos de nuevo el estado del repositorio
9. Volveremos a añadir estos cambios al índice de git
10. Guardaremos todos los cambios que hemos hecho en el repositorio

Ejercicio 2

Es hora de que nos conozcamos un poco más. Para ello vamos a colaborar todos en el repositorio del curso `https://github.com/labmiracle/miraclelabs2022v1.git`. Si es la primera vez que usamos `Github`, tendremos que registrarnos y activar nuestra cuenta. Además, tendremos que solicitar al docente que nos de acceso como colaboradores en el repositorio del curso.

1. Clonar el repositorio del curso en nuestra maquina
2. Crear una rama dentro del repositorio. Dicha rama se deberá llamar de la siguiente manera: `feature/<apellido-nombre>`
3. Cambiar a la rama que acabamos de crear
4. Dentro de la carpeta `alumnos`, crear otra carpeta con nuestras iniciales en minúsculas todas juntas. (p.e. `ldd`)
5. Crear un archivo `README.md` dentro de la carpeta que acabamos de generar
6. Abrir este archivo en el editor de código favorito y presentarse contando lo siguiente:
    - Nombre y apellidos y cómo prefieres que te llamen. Si eres Peter Parker, ya conocemos tu secreto :)
    - De donde vienes a donde vas. Cual es tu conocimiento previo
    - Qué hobbies tienes. En qué inviertes tu tiempo
    - ¿Qué es lo último que has aprendido? No necesariamente tiene que ser algo técnico

7. Guardar el contenido del archivo y luego de agregarlo al índice, confirmar los cambios
8. Subir el cambio al repositorio remoto
9. Crear una nueva `Pull Request` en el repositorio de Github. Esta `Pull Request` debe ir desde la rama que has creado hasta la rama `master`

Más información:
https://training.github.com/downloads/github-git-cheat-sheet.pdf
https://github.com/git-learning-game/oh-my-git
https://learngitbranching.js.org/

JavaScript

Ejercicio 1

- En el juego de golf, cada hoyo tiene un par, cuyo significado es el número promedio de golpes que se espera que haga un golfista para meter la bola en un hoyo y así completar el juego. Dependiendo de qué tan por encima o por debajo del par estén tus golpes, existe un apodo diferente.

Escribe una función que reciba los argumentos `par` y `strokes` y devuelve la cadena correcta de acuerdo a la siguiente tabla que enumera los golpes en orden de prioridad; de arriba (más alto) a abajo (más bajo):

```
Strokes  Return
1              "Hole-in-one!"
<= par - 2     "Eagle"
par - 1        "Birdie"
par            "Par"
par + 1        "Bogey"
par + 2        "Double Bogey"
>= par + 3     "Go Home!"
```

```
const names = [
  "Hole-in-one!",
  "Eagle",
  "Birdie",
  "Par",
  "Bogey",
  "Double Bogey",
  "Go Home!",
];

function golfScore(par, strokes) {
  return "Implementar!";
}

golfScore(5, 4);
```

Ejercicio 2

- Escribe una función recursiva `sum(arr, n)` que retorne la suma de los primeros `n` elementos del arreglo `arr`

```
function sum(arr, n) { }
```

Ejercicio 3

- Cada día una planta crece en metros según su `velocidadCrecimiento`. Cada noche, dicha planta decrece en metros en base a su `velocidadDecrecimiento` debido a la falta de sol. Cuando nace, mide exactamente `0` metros. Queremos saber los días que tardará una planta en alcanzar cierta `alturaDeseada`. Crear una función que reciba `velocidadCrecimiento`,

`velocidadDecrecimiento` y `alturaDeseada` como números enteros positivos y devuelva el número de días que tardará la planta en medir la `alturaDeseada`.

```
function calcularDiasCrecimiento(velocidadCrecimiento, velocidadDecrecimiento, alturaDeseada) {
}
```

Ejercicio 4

- Crea una función que reciba una frase como cadena y devuelva la palabra más larga:

  ```
  function palabraMasLarga(str) {
  }

  console.assert(palabraMasLarga('Erase una vez que se era') === 'Erase');
  ```

- Crea una función que reciba una cadena y lo devuelva con todas las palabras con su primera letra mayúscula

```
function primeraMayuscula(str) {
}

console.assert(primeraMayuscula('En un lugar de la Mancha') === 'En Un Lugar De La Mancha');
```

Ejercicio 5

- Crea una función que reciba una cadena y la devuelva en `camelCase`

```
function camelize(str) {
}
console.assert(camelize('Hola a todos que tal') === 'holaATodosQueTal');
```

Ejercicio 6

- Vamos a calcular el precio de un carrito de compra. Un carrito de compra tiene una propiedad `productos` que es una lista de los productos. Cada producto tiene las siguientes propiedades:
  - `nombre`: Nombre del producto (Papel higiénico, leche, ...)
  - `unidades`: Número de elementos que se van a comprar de dicho producto
  - `precio`: Precio unitario del producto

Además, tiene una propiedad `precioTotal` donde se va actualizando automáticamente el precio del producto. Crear el código necesario para soportar esta funcionalidad.

Datos de prueba:

```
const carrito = {
  productos: [{
    nombre: 'papel higienico',
    unidades: 4,
    precio: 5
  },
  {
    nombre: 'chocolate',
    unidades: 2,
    precio: 1.5
  }],
```



```
  get precioTotal() {

  }
}
```

Ejercicio 7

- En el juego de casino Blackjack, un jugador puede obtener una ventaja sobre la casa si lleva un registro del número relativo de cartas altas y bajas que quedan en la baraja. Esto se llama conteo de cartas.

Tener más cartas altas en el mazo favorece al jugador. A cada carta se le asigna un valor de acuerdo con la tabla siguiente. Cuando la cuenta es positiva, el jugador debe apostar alto. Cuando la cuenta es cero o negativa, el jugador debe apostar poco.

Escribe una función de conteo de cartas. Tendrá un parámetro `card`, que puede ser un número o una cadena, y aumentará o disminuirá la variable de conteo global de acuerdo con el valor de `card`. La función devolverá una cadena con el recuento actual y la cadena `Bet` si el recuento es positivo, o `Hold` si el recuento es cero o negativo. El recuento actual y la decisión del jugador (`Bet` o `Hold`) deben estar separados por un solo espacio

```
Count  Cards
+1  2, 3, 4, 5, 6
 0  7, 8, 9
-1  10, J, Q, K, A
```

Ejemplo:

```
-3 Hold
5 Bet
```

```
let count = 0;

function cc(card) {
  return "Implementar!";
}

cc(2);
cc(3);
cc(7);
cc("K");
cc("A");
```

Ejercicio 8

- Escriba una función que pueda tomar cualquier número de argumentos y devuelva la suma de todos los argumentos

Ejercicio 9

- Escriba una función llamada `addOnlyNums` que pueda aceptar cualquier cantidad de argumentos (incluidos números o cadenas) y devuelva la suma de solo los números

Ejercicio 10

- Escriba una función llamada `combineTwoArrays` que tome dos arreglos como argumentos y devuelva un solo arreglo que combine ambos (usando el operador `...`)

Ejercicio 11

- Escriba una función llamada `onlyUniques` que pueda tomar cualquier cantidad de argumentos y devuelva un arreglo de solo los argumentos únicos

Ejercicio 12

- Vamos a armar el juedo de "Adivinar el número". El juego consiste de los siguiente: La computadora genera un número aleatorio entre 1 y 10. Luego nos pide que adivinemos el número. Si el número que ingresamos es menor que el que genero la maquina nos muestra el mensaje "El número es mayor". Si el número que ingresamos es mayor nos muestra el mensaje "El número es menor". Cuando hayamos acertado el valor nos informa del fin del juego.
- Agregue al juego anterior niveles de dificultad.

