Desarrollo de Software V

Laboratorio # 15 – Prof. Regis Rivera

Objetivo: Hashing con JavaScript y su manejo con elementos multimedia

Aplicando Hash con SHA 256 para textos

Lab151.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Aplicando Hash de tipo SHA-256</title>
    <script src="sha256.js">
    </script>
    <script>
        function aplicarSHA256( str ){
            var hash = SHA256( str );
            document.getElementById('resultado').innerHTML = hash;
        }
    </script>
  </head>
  <body>
    <input type="text" id="texto_entrada">
    <input type="button" value="Aplicar Hash" onclick="aplicarSHA256(document.getElementById('texto_entrada').value)">
    <label id="resultado"></label>
  </body>
</html>
```

Este código ejecuta una función que esta dentro del archivo sha256.js adjunto a este laboratorio

Aplicando Hash con SHA 256 para imágenes, videos y archivos en general

Lab152.html

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Aplicando Hash de tipo SHA-256 para Imagenes y archivos en general</title>
    <script src="sha256.js">
    </script>
    <script>
        manejarArchivos = (files) => {
          console.log(files[0]);
          // 'files' será una matriz de archivos, incluso si solo se selecciona un archivo
          Object.keys(files).forEach((item, index) => {
            const file = files[index];
            // inicia una nueva instancia de 'FileReader'
            const reader = new FileReader();

            // proporciona 'onload callback' para esta instancia de 'FileReader'
            // esto se llama una vez reader.readAsArrayBuffer() esta terminado
            reader.onload = () => {
              const fileResult = reader.result;

              crypto.subtle.digest('SHA-256', fileResult).then((hash) => {
                var sha256result = hex(hash);
                // esto debería contener su valor hash sha-256
                console.log(sha256result);
                document.getElementById('result').innerHTML = sha256result;
              });
            };
```

```
30
31            // llamar a 'reader.readAsArrayBuffer' y proporcionar un 'file', debería activar la devolución de llamada anterior
32            // tan pronto comoreadAsArrayBuffer este completado
33            reader.readAsArrayBuffer(file);
34          });
35
36        }
37
38        function hex(buffer) {
39          var hexCodes = [];
40          var view = new DataView(buffer);
41          for (var i = 0; i < view.byteLength; i += 4) {
42            // El uso de 'getUint32' reduce la cantidad de iteraciones necesarias (procesamos 4 bytes cada vez)
43            var value = view.getUint32(i)
44            // 'toString(16)' dará la representación hexadecimal del número sin padding
45            var stringValue = value.toString(16)
46            // Usamos concatenación y corte para relleno (padding)
47            var padding = '00000000'
48            var paddedValue = (padding + stringValue).slice(-padding.length)
49            hexCodes.push(paddedValue);
50          }
51
52          // Une todas las cadenas hexadecimales en una
53          return hexCodes.join("");
54        }
55    </script>
56  </head>
57  <body>
58    <label><input type="file" onchange="manejarArchivos(this.files)"><span title="Seleccionar Archivo"></span></label>
59    <div>
60        <p id="result">Ninguna imagen ha sido seleccionada</p>
61    </div>
62  </body>
63 </html>
```