Hecho con 💛 por alumnos de Henry

Contenido de la clase

Tiempo de lectura
8 min

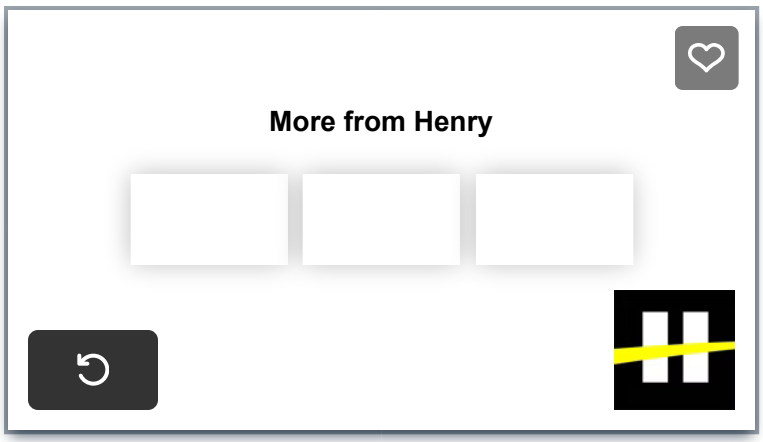
Homework 📝

JavaScript III

Continuación de bucles *for* y *arrays*

En esta lección cubriremos:

- **Introducción a los arrays**
- **Bucles *for* con arrays**



Introducción a los arrays (matrices/arreglos)

En la lección anterior discutimos los 3 tipos de datos básicos (cadenas/strings, números y booleanos) y cómo asignar esos tipos de datos a las

Dejanos tu feedback! 👍

Contenido de la clase

Dejanos tu feedback! 👍

ejemplo, ¿qué pasaría si quisiéramos hacer un seguimiento del nombre de cada estudiante en esta clase usando una sola variable, `nombresEstudiantes` . Podemos hacer eso usando Arrays. Podemos pensar en las matrices como contenedores de almacenamiento para colecciones de datos. Construir una matriz es simple, declarar una variable y establecerla en []. Luego podemos agregar al contenedor (separadas por coma) tantas cadenas, números o booleanos como queramos y acceder a esos elementos cuando lo deseemos.

```
const nombresEstudiantes = ['Martin', 'Antonio'
```

.length

Al igual que el tipo de dato *String* tiene un método incorporado `.length` , también lo hace la matriz. De hecho, la matriz tiene muchos métodos incorporados útiles (los discutiremos en lecciones posteriores). Al igual que la cadena `.length` cuenta los caracteres, la matriz `.length` devolverá el número de elementos en una matriz:

```
const nombresEstudiantes = ['Martin', 'Antonio'

console.log(nombresEstudiantes.length);  // 4
```

Acceso a elementos en una matriz

Podemos acceder a un elemento de una matriza en cualquier momento, solo necesitamos llamar al elemento por su posición en la matriz. Los elementos reciben una posición numérica (índice) de acuerdo con su ubicación en la matriz, en orden. El orden numérico de una matriz SIEMPRE comienza en 0, por lo que el primer elemento está en el índice 0, el segundo en el índice 1, el tercero en el 2, y así sucesivamente (esto puede ser

Contenido de la clase


```
const nombresEstudiantes = ['Martin', 'Antonio'
                                    0         1
```

Para acceder al elemento, escribiremos el nombre o la variable de matriz, seguidos de corchetes que contienen la asignación numérica.

```
const nombresEstudiantes = ['Martin', 'Antonio'

console.log(nombresEstudiantes[1]);  // 'Antoni
```

Para acceder dinámicamente al último elemento de la matriz, utilizaremos el método `.length`. En nuestra matriz `nombresEstudiantes`, la longitud es 4. Sabemos que el primer elemento siempre será 0, y cada elemento posterior se desplaza sobre un número. Entonces, en nuestro ejemplo, el último elemento tiene un índice de 3. Usando nuestra propiedad de longitud mostraremos cómo se hace cuando no sabemos el número de elementos en una matriz:

```
const nombresEstudiantes = ['Martin', 'Antonio'

console.log(nombresEstudiantes[nombresEstudiant
```

Asignación

Podemos asignar y reasignar cualquier índice en la matriz usando el paréntesis/índice y un "=".

Dejanos tu feedback! 👍

Contenido de la clase

Dejanos tu feedback! 👍

```
nombresEstudiantes[0] = 'Jorge';

console.log(nombresEstudiantes);  // ['Jorge',
```

.push y .pop

Otros dos métodos de matriz incorporados muy útiles son `.push` y `.pop`. Estos métodos se refieren a la adición y eliminación de elementos de la matriz después de su declaración inicial.

`.push` agrega un elemento al final de la matriz, incrementando su longitud en 1. `.push` devuelve la nueva longitud.

```
const nombresEstudiantes = ['Martin', 'Antonio'

nombresEstudiantes.push('Patricia');

console.log(nombresEstudiantes);  // ['Martin',
```

`.pop` elimina el último elemento de la matriz, disminuyendo la longitud en 1. `.pop` devuelve el elemento "reventado" (*popped*).

```
const nombresEstudiantes = ['Martin', 'Antonio'

nombresEstudiantes.pop();

console.log(nombresEstudiantes);  // ['Martin',
```

.unshift y .shift

`.unshift` y `.shift` son exactamente como `.push` y `.pop`, excepto que operan en el primer

Contenido de la clase

Dejanos tu feedback! 👍

de la matriz.

```
const nombresEstudiantes = ['Martin', 'Antonio'

nombresEstudiantes.unshift('Leo');

console.log(nombresEstudiantes);  // ['Leo', 'M

nombresEstudiantes.shift();

console.log(nombresEstudiantes);  // ['Martin',
```

Notas sobre las matrices

Debido a que Javascript no es un lenguaje fuertemente tipado, las matrices tampoco necesitan ser tipadas. Las matrices en Javascript pueden contener múltiples tipos de datos diferentes en la misma matriz.

Utilizando bucles *for* en arrays

La mayoría de las veces, los bucles for se utilizan para iterar sobre todos los elementos de una matriz. Usando la técnica de acceso al índice ("index access technique") podemos acceder a cada elemento de la matriz. Para hacer esto, usamos el método `.length` como punto de parada para el ciclo.

```
const nombresEstudiantes = ['Martin', 'Antonio'

for (let i = 0; i < nombresEstudiantes.length;
    console.log(nombresEstudiantes[i]);
}

// 'Martin'
```

Contenido de la clase

Recursos adicionales

- **MDN: Arrays**
- **MDN: for Loops**

Homework

Completa la tarea descrita en el archivo **README**

Si tienes dudas sobre este tema, puedes consultarlas en el canal *04_js-iii* de Slack

Dejanos tu feedback!👍