Apuntes clase 2

Operadores aritméticos (de asignación)

```
let num = 10; // 10
num += 10; // 20
num -= 10; // 10
num /= 2; // 5
num *= 2; // 10
```

Módulo

El operador aritmético para ver el resto de una división se llama módulo (`%`):

```
10 % 2; // 0
11 % 2; // 1
7 % 5; // 2

let num = 10;

if (num % 2 == 0) {
    // num es par
} else {
    // num es impar
}
```

Operadores lógicos

Tabla de verdad para *OR* (`||`)

A	B	A || B
false	false	false
false	true	true
true	false	true
true	true	true

Tabla de verdad para *AND* (`&&`)

A	B	A && B
false	false	false
false	true	false
true	false	false
true	true	true

Bucles

For

```
for (let i = 0; i < 3; ++i){
        console.log(i);
}

// lo que se ve en la consola:
// 0
// 1
// 2
```

Podemos dividir un `for` en 3 partes: `for (primera parte; segunda; tercera)`

- **primera parte:** se usa para crear variables. Es lo primero en ejecutarse y solo lo hace una vez.
- **segunda parte:** la condicion para que la ejecucion del `for` continue. De no cumplirse, este termina. Se evalua antes de cada ciclo.
- **tercera parte:** se utiliza para modificar las variables creadas en la primera parte. Se ejecuta luego del codigo dentro de las llaves del `for`, terminando cada ciclo.

While

Alternativa al `for`.

```
let i = 0;

while (i < 3){
        console.log(i++);
        // recordar que el ++ luego de la variable,
        // primero se usa como esta y luego suma 1
}

// 0
// 1
// 2
```

Existe una variante llamada `do while`. La diferencia es que se ejecuta antes de chequear la condicion, en cambio el `while` normal primero chequea y luego se ejecuta.

```
let i = 1;

do {
        console.log(i++);
} while (i < 1)

// 1
// en cambio

i = 1;
while (i < 1){
        console.log(i++); // no se llega a ejecutar
}
```

Arreglos (arrays)

```
// index:  0  1  2  3  4
let arr = [1, 2, 3, 4, 5];

arr[2] = 10; // [1, 2, 10, 4, 5]

// agrega un elemento al final del arreglo
arr.push(6); // [1, 2, 10, 4, 5, 6]

// elimina el ultimo elemento
arr.pop(); // [1, 2, 10, 4, 5]

let newArr = [] // arreglo vacio
newArr[0] = "hola"; // ["hola"]
newArr[1] = "mundo"; // ["hola", "mundo"]
```

Strings (cadenas de texto)

Recordar que podemos acceder a cada letra de un *string* al igual que con un *array*.

```
let str = "Hello World!";
str[0]; // "H"
str[str.length - 1] = "?"; // "Hello World?"

// Concatenacion
str += "I'm Tiny Rick!" // "Hello World? I'm Tiny Rick!"
```

```
let frase = str + "What's your name?"
// Hello World? I'm Tiny Rick! What's your name?
```