1. **Write a program to demonstrate JavaScript loops, operators and conditions?**

   Output:

2.

```
▶ 174  Third-party cookie will be blocked. Learn more in the Issues tab.

> // Function to check if a number is prime
  function isPrime(num) {
      if (num <= 1) {
          return false;
      }
      for (let i = 2; i <= Math.sqrt(num); i++) {
          if (num % i === 0) {
              return false;
          }
      }
      return true;
  }

  // Main program
  function main() {
      // Loop from 1 to 20
      for (let i = 1; i <= 20; i++) {
          if (i % 2 === 0) {
              console.log(i + " is even");
          } else {
              console.log(i + " is odd");
          }

          if (isPrime(i)) {
              console.log(i + " is prime");
          } else {
              console.log(i + " is not prime");
          }

          if (isPrime(i)) {
              console.log(i + " is prime");
          } else {
              console.log(i + " is not prime");
          }
      }
  }

  // Call the main function
  main();
1 is odd                                                    VM32:21
1 is not prime                                              VM32:27
2 is even                                                   VM32:19
2 is prime                                                  VM32:25
3 is odd                                                    VM32:21
3 is prime                                                  VM32:25
4 is even                                                   VM32:19
4 is not prime                                              VM32:27
5 is odd                                                    VM32:21
5 is prime                                                  VM32:25
6 is even                                                   VM32:19
6 is not prime                                              VM32:27
```

2. Write a program to demonstrate different array and string methods in JavaScript?

Output:

```
> // Array methods
let fruits = ['Apple', 'Banana', 'Cherry', 'Date'];

// 1. Push: Add an element to the end of the array
fruits.push('Elderberry');
console.log("After push:", fruits);

// 2. Pop: Remove the last element from the array
let poppedFruit = fruits.pop();
console.log("After pop:", fruits);
console.log("Popped fruit:", poppedFruit);

// 3. Shift: Remove the first element from the array
let shiftedFruit = fruits.shift();
console.log("After shift:", fruits);
console.log("Shifted fruit:", shiftedFruit);

// 4. Unshift: Add an element to the beginning of the array
fruits.unshift('Apricot');
console.log("After unshift:", fruits);

// 5. Concat: Concatenate two arrays
let moreFruits = ['Fig', 'Grape'];
let allFruits = fruits.concat(moreFruits);
console.log("Concatenated array:", allFruits);
```

```
// String methods
let str = "Hello, world!";

// 1. Length: Get the length of the string
console.log("Length of the string:", str.length);

// 2. ToUpperCase: Convert the string to uppercase
console.log("Uppercase:", str.toUpperCase());

// 3. ToLowerCase: Convert the string to lowercase
console.log("Lowercase:", str.toLowerCase());

// 4. Substring: Extract a part of the string
console.log("Substring:", str.substring(7, 12));

// 5. Split: Split the string into an array of substrings
console.log("Split:", str.split(','));

// 6. Trim: Remove whitespace from both ends of the string
let paddedStr = "   Hello, world!   ";
console.log("Trimmed string:", paddedStr.trim());
```

```
After push: ▶ (5) ['Apple', 'Banana', 'Cherry', 'Date', 'Elderberry']

After pop: ▶ (4) ['Apple', 'Banana', 'Cherry', 'Date']

Popped fruit: Elderberry

After shift: ▶ (3) ['Banana', 'Cherry', 'Date']
```

```
// 6. Trim: Remove whitespace from both ends of the string
let paddedStr = "   Hello, world!   ";
console.log("Trimmed string:", paddedStr.trim());
```

```
After push: ▶ (5) ['Apple', 'Banana', 'Cherry', 'Date', 'Elderberry']   VM36:6
After pop: ▶ (4) ['Apple', 'Banana', 'Cherry', 'Date']   VM36:10
Popped fruit: Elderberry   VM36:11
After shift: ▶ (3) ['Banana', 'Cherry', 'Date']   VM36:15
Shifted fruit: Apple   VM36:16
After unshift: ▶ (4) ['Apricot', 'Banana', 'Cherry', 'Date']   VM36:20
Concatenated array: ▶ (6) ['Apricot', 'Banana', 'Cherry', 'Date', 'Fig', 'Grape']   VM36:25
Length of the string: 13   VM36:31
Uppercase: HELLO, WORLD!   VM36:34
Lowercase: hello, world!   VM36:37
Substring: world   VM36:40
Split: ▶ (2) ['Hello', ' world!']   VM36:43
Trimmed string: Hello, world!   VM36:47
```

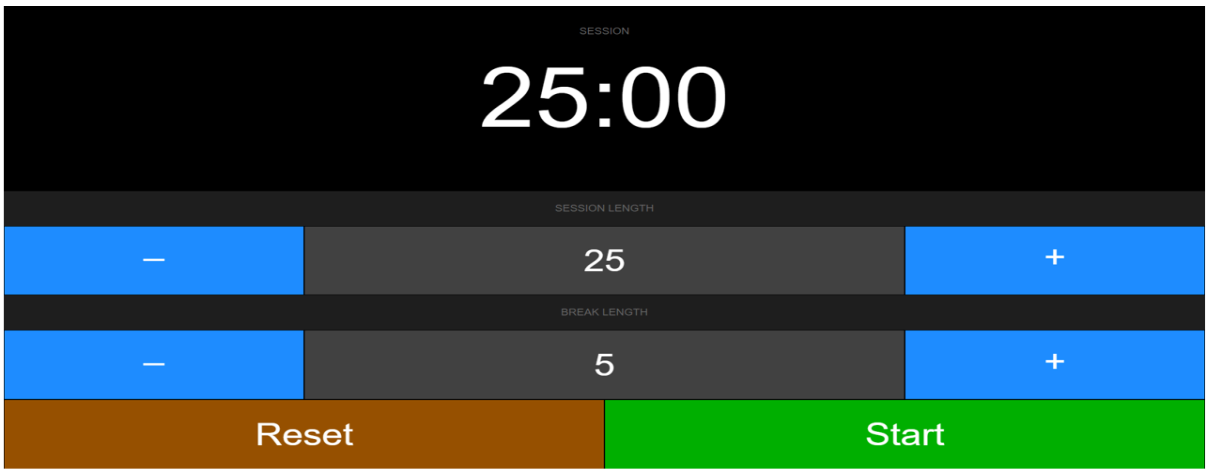
3. Write a program to show different ways to create a function in JavaScript?

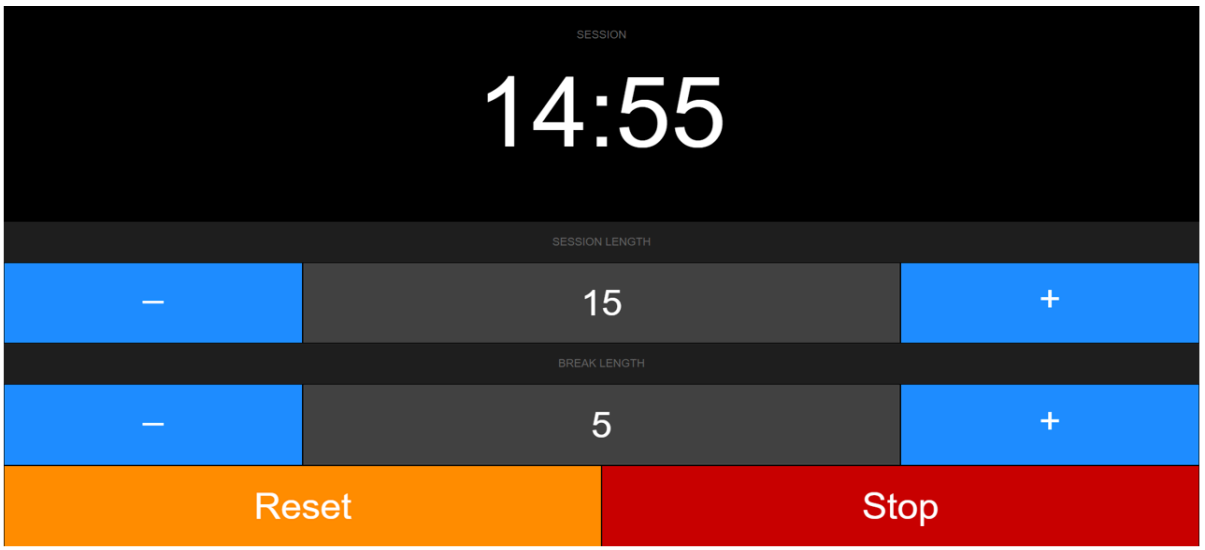
Output:

```
> // 1. Function Declaration
  function greet1(name) {
      return "Hello, " + name + "!";
  }

  // 2. Function Expression (Anonymous Function)
  let greet2 = function(name) {
      return "Hello, " + name + "!";
  };

  // 3. Function Expression (Named Function)
  let greet3 = function greetWithName(name) {
      return "Hello, " + name + "!";
  };

  // 4. Arrow Function Expression
  let greet4 = (name) => {
      return "Hello, " + name + "!";
  };

  // 5. Arrow Function Expression (Shortened)
  let greet5 = name => "Hello, " + name + "!";

< undefined

> console.log(greet1("Alice"));
  console.log(greet2("Bob"));
  console.log(greet3("Charlie"));
  console.log(greet4("David"));
  console.log(greet5("Eve"));
```

4. Write a program to implement pomodoro using JavaScript DOM?

Output:



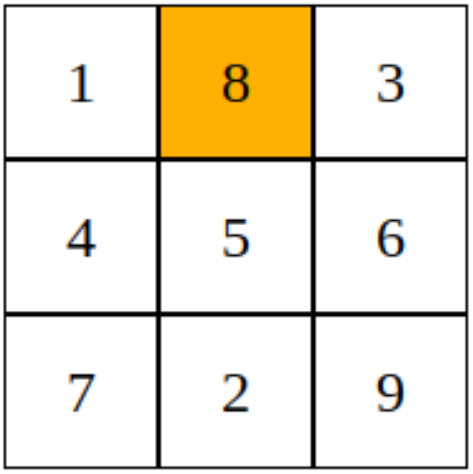


5. Write a program to implement swap 1 to 9 numbers using drag and drop?

Without Drag

1	2	3
4	5	6
7	8	9

After Drag



6. Demonstrate all ES6 concepts with examples.

1. let and const:

```
> let x = 5;
  x = 10; // Valid

  const PI = 3.14159;
  PI = 3; // Invalid, attempting to reassign a constant
```

2. Arrow Functions:

```
> // Traditional function
  function add(a, b) {
      return a + b;
  }

  // Arrow function
  const add = (a, b) => a + b;
```

3. Template Literals

```
> const name = 'Alice';
  console.log(`Hello, ${name}!`);
```

```
Hello, Alice!
```

4. Destructuring Assignment:

```
> // Array destructuring
  const [a, b] = [1, 2];

  // Object destructuring
  const { x, y } = { x: 1, y: 2 };
```

5. Spread Operator

```
> const arr1 = [1, 2, 3];
  const arr2 = [4, 5, 6];
  const combined = [...arr1, ...arr2]; // [1, 2, 3, 4, 5, 6]

  const obj1 = { a: 1, b: 2 };
  const obj2 = { c: 3, d: 4 };
  const merged = { ...obj1, ...obj2 }; // { a: 1, b: 2, c: 3, d: 4 }
```

6. Classes

```
> class Animal {
      constructor(name) {
          this.name = name;
      }

      speak() {
          console.log(`${this.name} makes a noise.`);
      }
  }

  class Dog extends Animal {
      speak() {
          console.log(`${this.name} barks.`);
      }
  }

  const dog = new Dog('Buddy');
  dog.speak(); // "Buddy barks."
```

```
Buddy barks.
```

7. Promises

```
> const fetchData = () => {
      return new Promise((resolve, reject) => {
          setTimeout(() => {
              resolve('Data fetched successfully');
          }, 2000);
      });
  };

  fetchData()
      .then(data => console.log(data))
      .catch(error => console.error(error));
```

```
<- ▶ Promise {<pending>}
```

```
Data fetched successfully
```