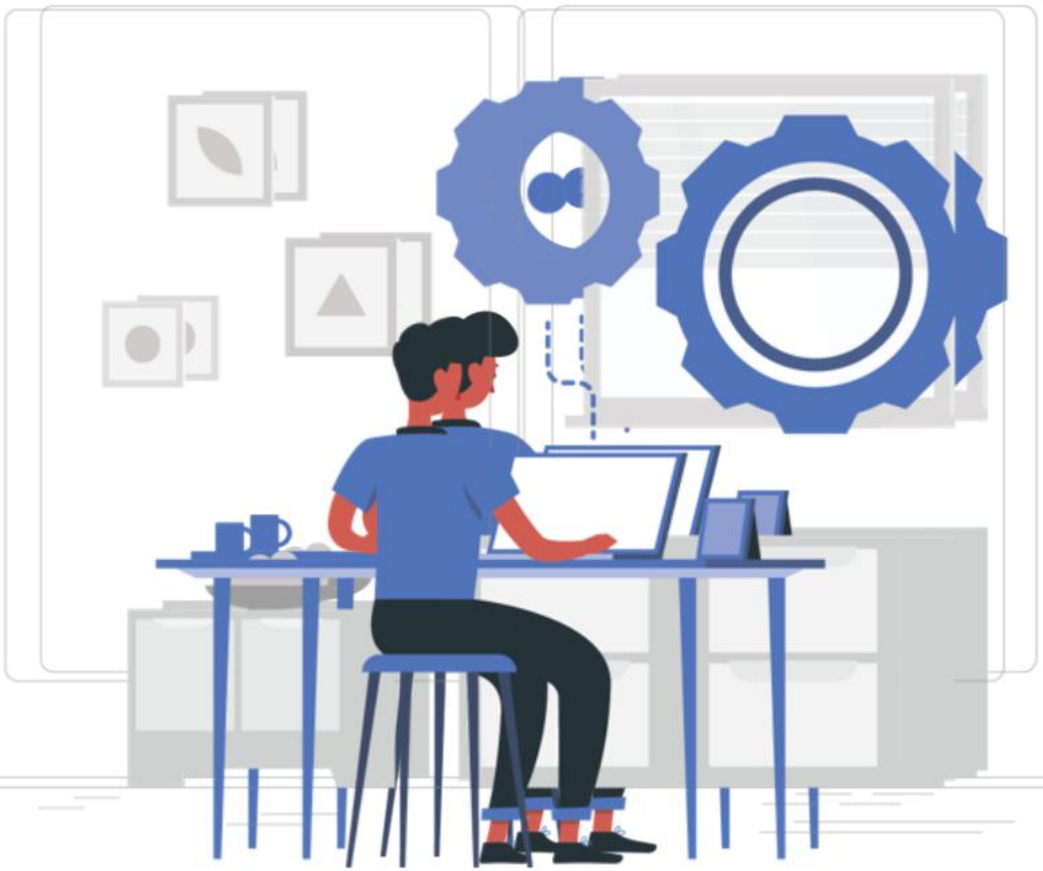

Practice

Getting Started With JavaScript

Practices

- Practice 1: Find the largest number using
  - nested if
  - ternary operator
- Practice 2: Count days for a given month and year
- Practice 3: Print first 10 numbers of the Fibonacci series
- Practice 4: Calculate the Sum of Digits of a Number

Points to Remember

- Ensure that the operator precedence and associativity rules are followed with relational and logical operators.
- Use DevTools for debugging the code.
- JavaScript code should be well-indented.
- Follow the JavaScript naming conventions for naming variables and constants.

PRACTICE

Practice 1: Find the Largest Number

Find the largest number from three unique numbers using:

a. nested `if`

b. ternary operator

Tasks

- The solution for this practice should be written in the file **p1-submission.js** located inside folder **p1-find-the-largest-number** of the boilerplate.
- Steps to perform this task
  - Step 1: Declare and initialize variables with the following given values:

Value 1	Value 2	Value 3
30	50	45

  - Step 2: Declare a variable to store the resultant largest number

Tasks (cont'd)

- Steps for method 1:
  - Step 1: Using nested if statements, compare the 3 values and store the largest value in the max variable.
  - Step 2: Display the largest value.
- Steps for method 2:
  - Step 1: Using ternary operator, compare the 3 values and store the largest value in the max variable.
  - Step 2: Display the largest value.
- Note: For the given values, both the methods should generate the same output as shown below:

```
Largest Number is 50
```

Expected Output

- Refer to the table given below for different numbers and the expected largest number:

Number 1	Number 2	Number 3	Largest Number
30	50	45	50
-20	-15	-10	-10
23	18	20	23
75	65	55	75
82	62	92	92

PRACTICE

Practice 2: Count Days for a Given Month and Year

Write a program that checks the month and year value and prints the number of days in that month.

Tasks

- The solution for this practice should be written in the file **p2-submission.js** located inside the folder **p2-day-count** of the boilerplate.
- Steps to perform this task
  - Step 1: Declare and initialize variables to store month and year values.
  - Step 2: Declare a variable to store day count
  - Step 3: Write the logic used to check the number of days in the month using `switch case` Statement.
  - Step 4: Test the logic for the following values of month and year:

Month	Year
2	2008
2	2009
1	2009
11	2009

Task (cont'd)

- Note:
  - 1st, 3rd, 5th, 7th, 8th, 10th and 12th month have 31 days.
  - 4th, 6th, 9th, and 11th month have 30 days.
  - 2nd month has 28 days, except the leap year, that has 29 days.
    - A leap year occurs every 4 years, e.g., 2004, 2008, 2012 and so on.
    - However, a century year is a leap year only if it comes after every 400 years, eg. 1200, 1600, 2000 and so on.
    - So, a year is a leap year if it is:
      - Divisible by 4 and not divisible by 100
      - Or divisible by 400

Expected Output

- Refer to the table given below for different values of month, year and the expected number of days:

Month	Year	No. of Days
2	2008	29
2	2009	28
1	2009	31
11	2009	30

PRACTICE

Practice 3: Print first 10 numbers of the Fibonacci series

Fibonacci numbers form a sequence of numbers where every number is the sum of the preceding two numbers. The first 2 values are 0 and 1 and their sum is 1 which is the next number in the sequence.

0,1,1,2,3,5...

Print the first 10 numbers of the Fibonacci Series.

Tasks

- The solution for this practice should be written in the file **p3-submission.js** located inside the folder **p3-fibonacci-first-10** of the boilerplate.
- Steps to perform this task:
  - Declare and initialize variable count with value 10.
  - Declare and initialize variable n1 with value 0.
  - Declare and initialize variable n2 with value 1
  - Display values of variables n1 and n2.
  - Write for loop that iterates till the value of variable controlling the loop is less than or equal to the value of count - 2.
  - At each iteration:
    - Store sum of n1 and n2 in variable n3.
    - Display value of n3.
    - Assign value of n2 to n1 and n3 to n2.

Expected Output

- The output expected for this practice is shown below:

```
0 1 1 2 3 5 8 13 21 34
```

PRACTICE

Practice 4: Calculate the Sum of Digits of a Number

Calculate the sum of the digits of the number 4386.

4386 → 4 + 3 + 8 + 6 = 21

Tasks

- The solution for this practice should be written in the file **p4-submission.js** located inside folder **p4-sum-of-digits** of the boilerplate.
- Steps to perform this task:
  - Declare and initialize variable `num` with value `4386`.
  - Declare and initialize variable `sum` with value `0`.
  - Write while loop that iterates till the value of `num` is greater than `0`.
    - At each iteration,
      - Fetch the digit at unit position by performing modulo division of value of `num` by `10`.
      - Update value of `num` to get the number without the last digit by dividing value of `num` by `10`.
      - Add the digit fetched to the current value of `sum` variable and store the resultant in `sum` variable.
    - Display the value of `sum` variable after the loop terminates.

Expected Output

- The output expected for this practice is shown below:

```
Sum of Digits of 4386 = 21
```