Day-03 Assignment

ALLU DIWAKAR_21BCE9213

2023-02-05

Task - 1

- 10 valid identifiers in JavaScript The first character must be a letter, underscore (_), or dollar sign ($).

```
myVariable
_private
$money
firstName
age_1
UPPERCASE
letterCount
$

_
PI
```

Task - 2

- Using template literals write a 3-line introduction (Must Include variables : name , regNo , branch , hooby) and log it on console using `console.log()`
  ##Expected output format

```
$ node Task2.js
Hey there! I'm Ashutosh Singh
My Registration no. is 20BCI7070
I'm pursuing BTECH with CSE AI at VIT-AP
I like Photography
```

```
let name = "Allu Diwakar";
let regNo = "21BCE9213";
let branch = "CSE";
let college = "VIT-AP";
let hobby = "Knowing the Unknown";

console.log(`Hey there! I'm ${name}
My Registration no. is ${regNo}
I'm pursuing BTECH with ${branch} at ${college}
I like ${hobby}`);
```

```
## Hey there! I'm Allu Diwakar
## My Registration no. is 21BCE9213
```

```
## I'm pursuing BTECH with CSE at VIT-AP
## I like Knowing the Unknown
```

Task - 3

Write a program that calculates and displays the grades of students based on their provided marks.

Marks	Grades
Mark < 44	F
44 < Mark <= 50	E
50 < Mark <= 60	D
60 < Mark <= 70	C
70 < Mark <= 80	B
80 < Mark <= 90	A
90 < Mark <= 100	S

Expected output format

```
$ node Task3.js
Marks : 93
Grade : S
```

```
let marks = 93;
let grade = "";

if (marks < 44) {
  grade = "F";
} else if (marks <= 50) {
  grade = "E";
} else if (marks <= 60) {
  grade = "D";
} else if (marks <= 70) {
  grade = "C";
} else if (marks <= 80) {
  grade = "B";
} else if (marks <= 90) {
  grade = "A";
} else {
  grade = "S";
}

console.log(`Marks : ${marks}
Grade : ${grade}`);
```

```
## Marks : 93
## Grade : S
```

Task - 4

Write a program to demonstrate the use of `switch case` and then re-write it using the ternary operator

```
let day = "Monday";
let message = "";

switch (day) {
  case "Monday":
    message = "Today is Monday";
    break;
  case "Tuesday":
    message = "Today is Tuesday";
    break;
  case "Wednesday":
    message = "Today is Wednesday";
    break;
  case "Thursday":
    message = "Today is Thursday";
    break;
  case "Friday":
    message = "Today is Friday";
    break;
  case "Saturday":
    message = "Today is Saturday";
    break;
  case "Sunday":
    message = "Today is Sunday";
    break;
  default:
    message = "Invalid day";
    break;
}

console.log(message);
```

```
## Today is Monday
```

```
let day = "Monday";
let message = "";

message = (day === "Monday") ? "Today is Monday" :
          (day === "Tuesday") ? "Today is Tuesday" :
          (day === "Wednesday") ? "Today is Wednesday" :
```

```
            (day === "Thursday") ? "Today is Thursday" :
            (day === "Friday") ? "Today is Friday" :
            (day === "Saturday") ? "Today is Saturday" :
            (day === "Sunday") ? "Today is Sunday" : "Invalid day";

console.log(message);
```

```
## Today is Monday
```

Task - 5

Write a program to print the following output using : - `for` - `while` - `do while`

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Expected output format

```
$ node Task5-1.js
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*

$ node Task5-2.js
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*

$ node Task5-3.js
*
```

```
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

```
for (let i = 1; i <= 5; i++) {
  let row = "";
  for (let j = 1; j <= i; j++) {
    row += "* ";
  }
  console.log(row);
}

for (let i = 4; i >= 1; i--) {
  let row = "";
  for (let j = 1; j <= i; j++) {
    row += "* ";
  }
  console.log(row);
}
```

```
## *
## * *
## * * *
## * * * *
## * * * * *
## * * * *
## * * *
## * *
## *
```

```
let i = 1;
while (i <= 5) {
  let row = "";
  let j = 1;
  while (j <= i) {
    row += "* ";
    j++;
  }
  console.log(row);
  i++;
}
```

```
i = 4;
while (i >= 1) {
  let row = "";
  let j = 1;
  while (j <= i) {
    row += "* ";
    j++;
  }
  console.log(row);
  i--;
}
```

```
## *
## * *
## * * *
## * * * *
## * * * * *
## * * * *
## * * *
## * *
## *
```

```
let i = 1;
do {
  let row = "";
  let j = 1;
  do {
    row += "* ";
    j++;
  } while (j <= i);
  console.log(row);
  i++;
} while (i <= 5);

i = 4;
do {
  let row = "";
  let j = 1;
  do {
    row += "* ";
    j++;
  } while (j <= i);
  console.log(row);
  i--;
} while (i >= 1);
```

```
## *
## * *
```

```
## * * *
## * * * *
## * * * * *
## * * * *
## * * *
## * *
## *
```

Task - 6

Re-write Task - 3 using different types of arrow functions. Which takes `marks` as an argument and returns the grade. Do multiple function calls with different values of marks.

Expected output format

```
$ node Task6.js
Marks : 63
Grade : C
Marks : 90
Grade : A
Marks : 42
Grade : F
Marks : 72
Grade : B
```

```
// Using traditional function
const calculateGrade = function(marks) {
  if (marks < 44) {
    return 'F';
  } else if (marks <= 50) {
    return 'E';
  } else if (marks <= 60) {
    return 'D';
  } else if (marks <= 70) {
    return 'C';
  } else if (marks <= 80) {
    return 'B';
  } else if (marks <= 90) {
    return 'A';
  } else {
    return 'S';
  }
};

// Using anonymous arrow function
const calculateGrade2 = (marks) => {
  if (marks < 44) {
    return 'F';
  } else if (marks <= 50) {
    return 'E';
```

```
  } else if (marks <= 60) {
    return 'D';
  } else if (marks <= 70) {
    return 'C';
  } else if (marks <= 80) {
    return 'B';
  } else if (marks <= 90) {
    return 'A';
  } else {
    return 'S';
  }
};

// Using arrow function with implicit return
const calculateGrade3 = (marks) =>
  marks < 44
    ? 'F'
    : marks <= 50
    ? 'E'
    : marks <= 60
    ? 'D'
    : marks <= 70
    ? 'C'
    : marks <= 80
    ? 'B'
    : marks <= 90
    ? 'A'
    : 'S';

console.log(`Marks : 63\nGrade : ${calculateGrade(63)}`);
console.log(`Marks : 90\nGrade : ${calculateGrade2(90)}`);
console.log(`Marks : 42\nGrade : ${calculateGrade3(42)}`);
console.log(`Marks : 72\nGrade : ${calculateGrade(72)}`);
```

```
## Marks : 63
## Grade : C
## Marks : 90
## Grade : A
## Marks : 42
## Grade : F
## Marks : 72
## Grade : B
```

Task - 7

Create an object that contains your details as listed in Task - 2.

```
const personalDetails = {
  name: "Allu Diwakar",
```

```
  regNo: '21BCE9213',
  branch: 'BTECH in CSE ',
  institute: 'VIT-AP',
  hobby: 'Knowing the Unknown'
};

console.log(personalDetails);
```

```
## {
##   name: 'Allu Diwakar',
##   regNo: '21BCE9213',
##   branch: 'BTECH in CSE ',
##   institute: 'VIT-AP',
##   hobby: 'Knowing the Unknown'
## }
```

Task - 8

Modify the object in Task - 7 and add a function that introduces the user upon being called as demonstrated in Task - 2. (`objectName.introduce()`)

```
const personalDetails = {
  name: 'Allu Diwakar',
  regNo: '21BCE9213',
  branch: 'BTECH in CSE ',
  institute: 'VIT-AP',
  hobby: 'Enfolding the Unfolding',
  introduce: function() {
    console.log(`Hey there! I'm ${this.name}`);
    console.log(`My Registration no. is ${this.regNo}`);
    console.log(`I'm pursuing ${this.branch} at ${this.institute}`);
    console.log(`I like ${this.hobby}`);
  }
};

personalDetails.introduce();
```

```
## Hey there! I'm Allu Diwakar
## My Registration no. is 21BCE9213
## I'm pursuing BTECH in CSE  at VIT-AP
## I like Enfolding the Unfolding
```

Task - 9

Create an array in JavaScript and demonstrate the use of `push()` and `pop()` on it

```
let arr = [1, 2, 3, 4];

// Using push() to add an element to the end of the array
arr.push(5);
console.log(arr);

// Using pop() to remove the last element of the array
arr.pop();
console.log(arr);
```

```
## [ 1, 2, 3, 4, 5 ]
## [ 1, 2, 3, 4 ]
```

Task - 10

Demonstrate the use of the following function in JavaScript - map() - filter() - reduce() - some() / every() - find() / findIndex() - forEach() - slice() - concat() - includes()

```
let arr = [1, 2, 3, 4, 5];

// Using map() to create a new array with the result of a function on each ele

let mappedArray = arr.map(function(value) {
  return value * 2;
});
console.log(mappedArray);

// Using filter() to create a new array with elements that pass a certain cond
let filteredArray = arr.filter(function(value) {
  return value % 2 === 0;
});
console.log(filteredArray);

// Using reduce() to reduce an array to a single value
let reducedValue = arr.reduce(function(accumulator, currentValue) {
  return accumulator + currentValue;
});
console.log(reducedValue);

// Using some() to check if at least one element in an array passes a conditio
let someResult = arr.some(function(value) {
  return value > 3;
});
console.log(someResult);

// Using every() to check if all elements in an array pass a condition
let everyResult = arr.every(function(value) {
  return value > 0;
});
```

```
console.log(everyResult);

// Using find() to find the first element that passes a condition
let foundValue = arr.find(function(value) {
  return value > 3;
});
console.log(foundValue);

// Using findIndex() to find the index of the first element that passes a cond
let foundIndex = arr.findIndex(function(value) {
  return value > 3;
});
console.log(foundIndex);

// Using forEach() to perform an action for each element in an array
arr.forEach(function(value) {
  console.log(value);
});


// Using slice() to create a new array with a portion of an existing array
let slicedArray = arr.slice(1, 3);
console.log(slicedArray);

// Using concat() to concatenate two arrays
let arr2 = [6, 7, 8];
let concatenatedArray = arr.concat(arr2);
console.log(concatenatedArray);

// Using includes() to check if an array contains a certain element
let includesResult = arr.includes(3);
console.log(includesResult);
```

```
## [ 2, 4, 6, 8, 10 ]
## [ 2, 4 ]
## 15
## true
## true
## 4
## 3
## 1
## 2
## 3
## 4
## 5
## [ 2, 3 ]
## [
##   1, 2, 3, 4,
##   5, 6, 7, 8
```

```
## ]
## true
```