University of Essex

Assignment Coversheet

- **Please complete and attach this form to your assignment. All assignments must be submitted on the stipulated submission date.**

Program / Intake : BSc (Hons) in Computer Science 1st Intake

Pathway: na

Assignment Type *(optional)***: CE212 Web Application Programming Part 2 (Simple Interest Calculator)**

Student Name:	Wong Pei Rong, Sam	**Student Number (PRID):**	CT0363889
Module Code:	CE212	**Module Title:**	**Web Application Programming**
Lecturer/Tutor:	Mdm Malar Kodi	**Grade:**	

DECLARATION: I hereby declare that the attached assignment is my own work. I understand that if I am suspected of plagiarism or another form of cheating, I will be subject to disciplinary actions.

Signed: Sam Wong Pei Rong Date Submitted: 26 May 2022

CE 212 Web Application Programming

Assignment 1 - Part 2

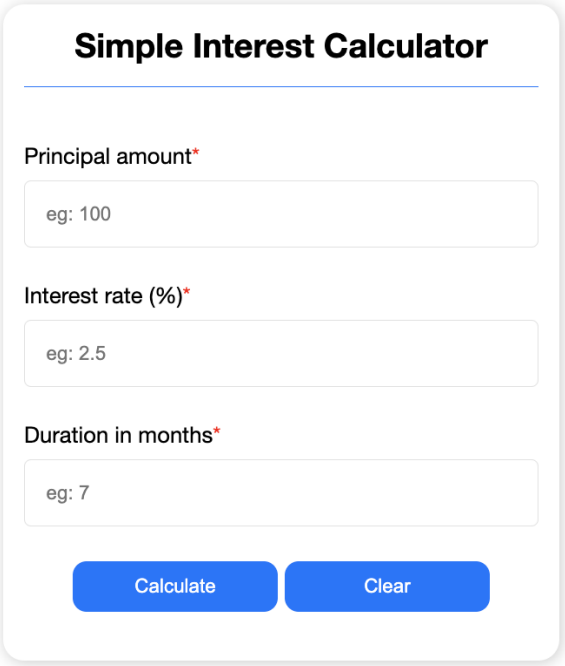
Simple Interest Calculator

Wong Pei Rong (Sam)
CT0363889

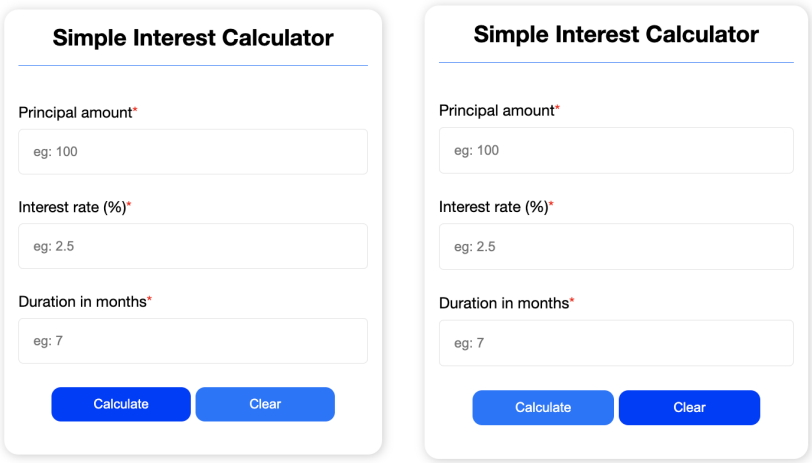
Simple Interest Calculator Demo

Upon running the code the user will see the following. Similar to part 1 I want my program to be functional while having equal emphasis on the design (UI/UX).

Placeholders have been included in each of the fills to guide the user on what input to enter.



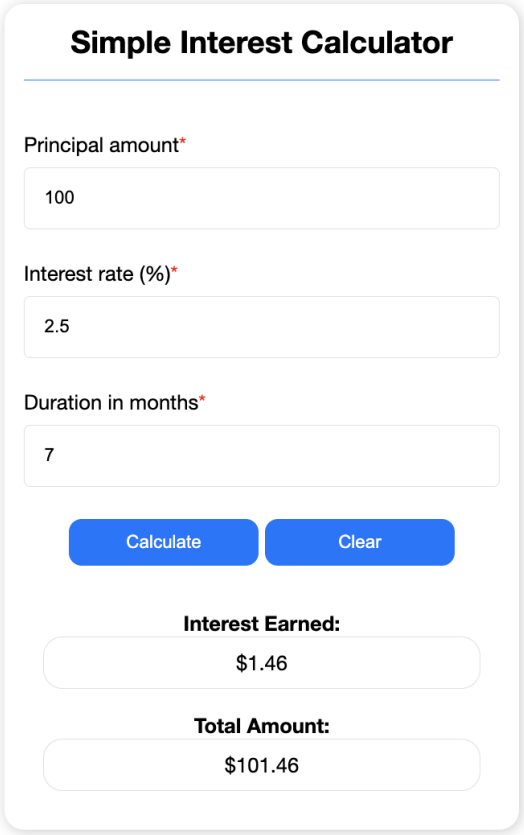
When the user hovers over the calculate button the color of the button will change.



The results are hidden and will only appear when the user has entered their input. Ensuring that only relevant information is presented, preventing clutter and the hence improves the user experience.

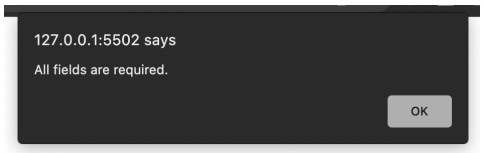
Using the calculator

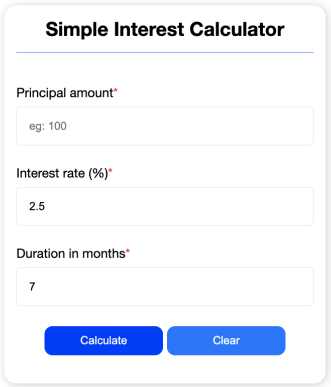
After the user's input the results will be shown at the bottom.



Error Handling - Invalid Input or leaving the fill blank

When leaving any required fills blank an alert will pop up.



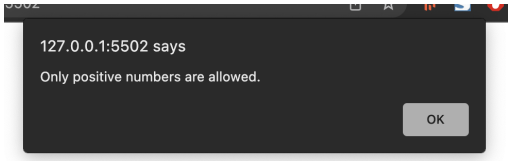


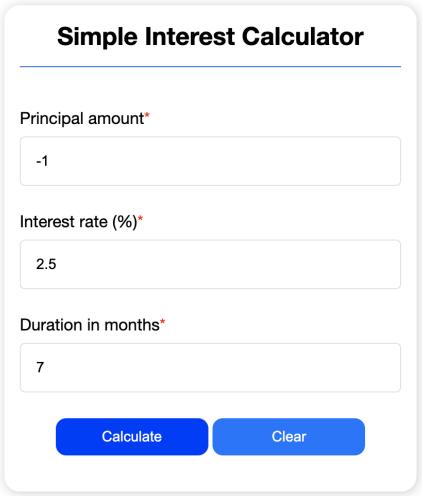

```
22      if(!P || !R || !T){
23          alert("All fields are required.");
24          reset();
25          return;
26      }
```

If any fill is empty, show alert and reset fills. This time round all fields will reset unlike the BMI calculator as there are only 3 fills in the interest calculate compared to 5 which is a lot more to rerenter.

Also the decision to do so is so that i can keep the validation code shorter and more concise

If user attempts to enter negative inputs an alert will pop up.

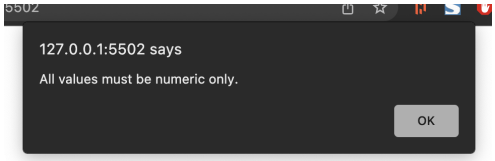


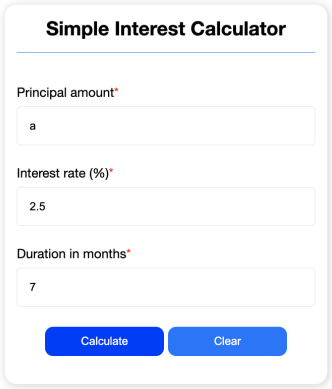



```
28      if(P<= 0 || R<= 0 || T<= 0){
29          alert("Only positive numbers are allowed.");
30          reset();
31          return;
32      }
```

If inputs are less then or equal to 0 show alert and reset fields

If user attempts to enter non-numeric inputs an alert will pop up.






```
34      if(isNaN(P) || isNaN(R) || isNaN(T)){
35          alert("All values must be numeric only.");
36          reset();
37          return;
38      }
39
```

If is Not-a-number, show alert and reset all fields

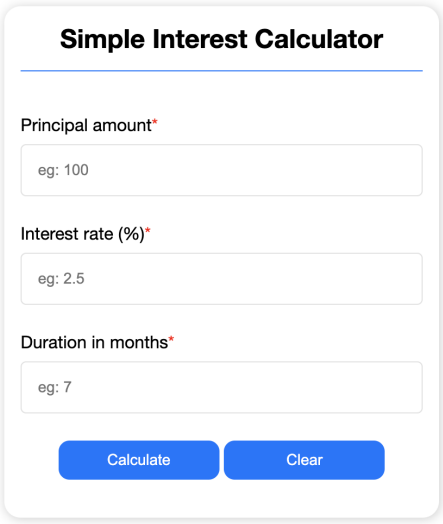
Breaking down my code (HTML)

HTML code for all the labels, fills, placeholders, and button.

Seperating them using instead or

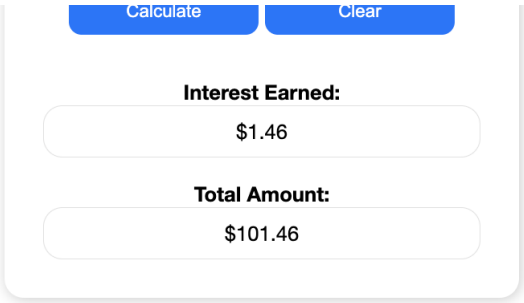
Grouping labels by class=”input” so they can be styled together in CSS.

```
<body>
    <div class="container">
        <div class="header">
            <h2 class="sim-interest">Simple Interest Calculator</h2>
        </div>
        <div class="form">
            <div class="input">
                <label>Principal amount<span class="required">*</span></label>
                <input type="text" id="p" placeholder="eg: 100" autocomplete="off">
            </div>
            <div class="input">
                <label>Interest rate (%)<span class="required">*</span></label>
                <input type="text" id="r" placeholder="eg: 2.5" autocomplete="off">
            </div>
            <div class="input">
                <label>Duration in months<span class="required">*</span></label>
                <input type="text" id="t" placeholder="eg: 7" autocomplete="off">
            </div>

        </div>
        <div class="btn-wrapper">
            <button class="btn">Calculate</button>
            <button class="btn-clear">Clear</button>
```



HTML Code for results

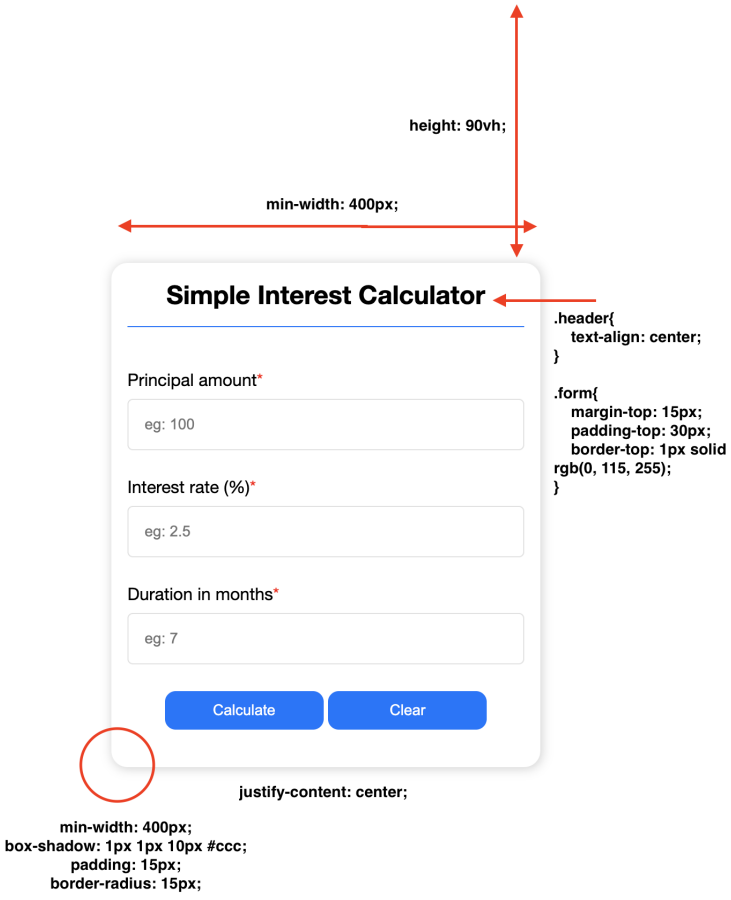
```
</div>
<div class="result">
    <strong>Interest Earned: </strong>
    <span class="Interest-earned">Test</span><br>

    <strong>Total Amount: </strong>
    <span class="total-amount">Test</span>
</div>
```



Breaking down my code (CSS)

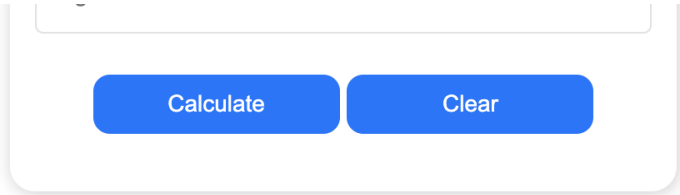
```css
body{
    display: flex;
    align-items: center;
    justify-content: center;
    margin-top: 30px;
    height: 90vh;
    width: 100%;
    font-family: "Helvetica Neue",Helvetica,Arial;
    font-size: 16;
}
```

```css
.container{
    min-width: 400px;
    box-shadow: 1px 1px 10px #ccc;
    padding: 15px;
    border-radius: 15px;
}

.header{
    text-align: center;
}

.form{
    margin-top: 15px;
    padding-top: 30px;
    border-top: 1px solid rgb(0, 115, 255);
}
```

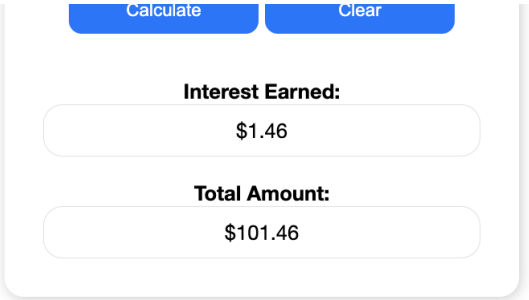


```
58
59  .btn,
60  .btn-clear{
61        padding: 10px 25px;
62        min-width: 40%;
63        margin-top: 10px;
64        margin-bottom: 20px;
65        border: 0;
66        font-size: 16;
67        color: white;
68        background-color: rgb(0, 115, 255);
69        border-radius: 10px;
70        text-align: center;
71        cursor: pointer;
72  }
73
```

```
73
74  .btn:hover,
75  .btn-clear:hover{
76        background-color: rgb(0, 51, 255);
77  }
78
79  .btn-wrapper{
80        text-align: center;
81  }
82
```



```
83
84  .result{
85      padding: 15px;
86      text-align: center;
87  }
88
89
90  .Interest-earned{
91      display: block;
92      border: 1px solid rgb(226, 226, 226);
93      padding: 10px;
94      border-radius: 15px;
95  }
96
97  .total-amount{
98      display: block;
99      border: 1px solid rgb(226, 226, 226);
100     padding: 10px;
101     border-radius: 15px;
102 }
```



Breaking down my code (Javascript)

For the interest calculator the aim what to make my code more concise so it is less prone to errors and easier to read.

Hiding Results

```
const resultSection = document.getElementsByClassName("result")[0];
resultSection.style.display = "none";
```

Getting result via class name and hidding it. When calculation function is invoked without errors the result section will be made visible.

```
53
54          resultSection.style.display = "block";
```

Connecting Buttons

```
6   const calculate = document.getElementsByClassName("btn")[0];
7   const clear = document.getElementsByClassName("btn-clear")[0];
8
```

Getting buttons via class name and assigning them with their relevant methods

```
59
60    calculate.addEventListener('click', calculateAmount);
61    clear.addEventListener('click', reset);
```

Function that resets all text fields and hides the results

```
const reset = () => {
    const P = document.getElementById("p").value = '';
    const R = document.getElementById("r").value = '';
    const T = document.getElementById("t").value = '';
    resultSection.style.display = "none";
}
```

Calculation Function

```
const calculateAmount = () => {
    const P = document.getElementById("p").value;
    const R = document.getElementById("r").value;
    const T = document.getElementById("t").value;

    if(!P || !R || !T){
        alert("All fields are required.");
        reset();
        return;
    }

    if(P<= 0 || R<= 0 || T<= 0){
        alert("Only positive numbers are allowed.");
        reset();
        return;
    }

    if(isNaN(P) || isNaN(R) || isNaN(T)){
        alert("All values must be numeric only.");
        reset();
        return;
    }
```

Gettign text inputs via ID and validates inputs no parsing is done yet so as to not change the user's input

```
const P2 = parseFloat(P);
const R2 = parseFloat(R);
const T2 = parseFloat(T);

let interestEarn = 0;
let result2 = 0;

interestEarn = ((P2 * R2 * (T2/12)) / 100);
result2 =  P2 + interestEarn;

resultSection.style.display = "block";
InterestAmount.innerHTML = "$" + interestEarn.toFixed(2);
finalAmount.innerHTML = "$" + result2.toFixed(2);
```

Parsing is then done so calculation can be done without error.