PRINCIPALI TECNOLOGIE FRONT END:

HTML (HYPERTEXT MARKUP LANGUAGE)--→>>>> DEFINIRE LA STRUTTURA LAYOUT) DI UNA PAGINA WEB

OGNI PAGINA HTML PUO' ESSERE VISTA COME UNA STRUTTURA DOM (DOCUMENT OBJECT MODEL)

LIMITE: LINGUAGGIO STATICO-→>>> NON RIESCE A MODIFICARE LA STRUTTURA DEL DOM A RUNTIME RISPETTO ALL'INTERAZIONE DELL'UTENTE CHE NAVIGA LA PAGINA WEB

JAVASCRIPT = LINGUAGGIO INTERPRETATO DA UN RUNTIME ENVIRONMENT DI NOME JAVASCRIPT ENGINE (INTRINSECAMENTE ESISTENTE ALL'INTERNO DI OGNI BROWSER), EVENT DRIVEN (IN GRADO DI MODIFICARE LA STRUTTURA ORIGINARIA DEL DOM IN BASE AL VERIFICARSI DI EVENTI ALL'INTERNO DI UNA PAGINA WEB OVVERO INTERAZIONI DELL'UTENTE CHE LA NAVIGA, DEBOLMENTE TIPIZZATO (NON TIPIZZATO), ADERENTE A DUE PARADIGMI (OOP E FUNZIONALE)

REACT JS = JAVASCRIPT API = TECNOLOGIA CHE USA LINGUAGGIO NATIVO JAVASCRIPT E CONSENTE DI IMPLEMENTARE A RUNTIME IL VIRTUAL DOM (OTTIMIZZAZIONE DELLE PERFORMANCE DI CARICAMENTO DELLE PAGINE WEB)

ANGULAR = FRAMEWORK MVC FRONT END

TECNOLOGIE DI STYLING:CSS, SASS, LESS, BOOTSTRAP, ANGULAR MATERIAL

ESISTONO TRE DIVERSE MODALITA' SINTATTICHE PER CREARE LIVE OBJECTS:

- VAR/CONST OBJECT
- PROTOTYPE OBJECT
- FUNCTIONAL/CONSTRUCTOR OBJECT

INTRO

```
<html>

<body>

    <script>

        document.write("hello");

        alert("hello")

    </script>


</body>

</html>
```

TRADITIONAL FUNCTION

```
<html>

<body>

    <script>

       function sayHello(){

        alert("hello");

       }

    </script>

<input type="button" value="sayHello" onclick="sayHello()"></input>

</body>

</html>
```

ARROW ANONYMOUS FUNCTION

```
<html>

<body>

    <script>

       function sayHello(){

        alert("hello");

       }

       var sayHelloNew = ()=>{

        alert("hello");

       }

    </script>

<input type="button" value="sayHello" onclick="sayHelloNew()"></input>

</body>

</html>
```

LIVE OBJECTS

```
<html>

<body>

    <div id="p1"></div>
    <div id="p2"></div>
    <div id="p3"></div>

    <script>

        // VAR OBJECT
        var person1 = {

            firstName: "Mario",
            lastName: "Rossi",
            age: 21

        }

        person1.email = "mrossi@libero.it";

        // PROTOTYPE
        var person2 = new Object();
        person2.firstName = "Giulio";
        person2.lastName = "Verdi";
        person2.age = 24;

        // CONSTRUCTOR
        function Person(firstName, lastName, age) {

            this.firstName = firstName;
            this.lastName = lastName;
            this.age = age;

        }

        var person3 = new Person("Maria","Lesti",23);



        document.getElementById("p1").innerHTML = person1.firstName + " " +
person1.lastName + " " + person1.age + " " + person1.email;
        document.getElementById("p2").innerHTML = person2.firstName + " " +
person2.lastName + " " + person2.age;
        document.getElementById("p3").innerHTML = person3.firstName + " " +
person3.lastName + " " + person3.age;

    </script>
```

```
</body>

</html>
```

JSON_OBJECTS.HTML

```
<html>

<body>

    <div id="p1"></div>
    <div id="p2"></div>


    <script>

        // SIMPLE JSON OBJECT
        var person1 = {

            "firstName": "Mario",
            "lastName": "Rossi",
            "age": 21

        }

        var person2 = {

            "firstName": "Giulia",
            "lastName": "Pisu",
            "age": 24,
            "auto" : {

                "auto1":"Dacia",
                "auto2":"Fiat"

            }

        }



        document.getElementById("p1").innerHTML = person1.firstName + " " +
person1.lastName + " " + person1.age;
        document.getElementById("p2").innerHTML = person2.firstName + " " +
person2.lastName + " " + person2.auto.auto1 + " " + person2.auto.auto2

    </script>
```

```
</body>

</html>
```

STRUTTURE DATI JAVASCRIPT

function.js

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark","Sql");
    courses.forEach(course=>console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course=>console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course=>console.log(course));



}
```

structures_journey.html

```
<html>


<body>


    <script src="functions.js">



    </script>

    <input type="button" value="array1" onclick="arrayBuild1()"></input><br>
    <input type="button" value="array2" onclick="arrayBuild2()"></input><br>
    <input type="button" value="array3" onclick="arrayBuild3()"></input><br>



</body>



</html>
```

PUSH E POP ALTERANO L'ARRAY ESISTENTE RESTITUENDO LA STRUTTURA ALTERATA DELL'ARRAY ORIGINARIO

AGGIUNGERE LA FUNZIONE arrayBuild4() a functions.js

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark","Sql");
    courses.forEach(course=>console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course=>console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course=>console.log(course));



}

function arrayBuild4(){

    var numbers = [76,89,54,21];
    numbers.map(number=>console.log(number));
    var numbersNew = numbers.map(number=>(number*2));
    numbersNew.forEach(numberNew=>console.log(numberNew));


}
```

MODIFICARE structures_journey.html

```
<html>


<body>


    <script src="functions.js">



    </script>

    <input type="button" value="array1" onclick="arrayBuild1()"></input><br>
    <input type="button" value="array2" onclick="arrayBuild2()"></input><br>
    <input type="button" value="array3" onclick="arrayBuild3()"></input><br>
    <input type="button" value="array4" onclick="arrayBuild4()"></input><br>



</body>



</html>
```

map può essere usato come forEach (in questo caso restituisce le informazioni contenute nell'array originario)

In alternativa, map può essere usato pure come operatore di transformation (in questo caso restituisce un nuovo Array)

AGGIUNGERE LA FUNZIONE arrayMerge a functions.js per far vedere come funziona lo spread operator (introdotto nella versione ECMA SCRIPT 6)

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark","Sql");
    courses.forEach(course=>console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course=>console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course=>console.log(course));



}

function arrayBuild4(){

    var numbers = [76,89,54,21];
    numbers.map(number=>console.log(number));
    var numbersNew = numbers.map(number=>(number*2));
    numbersNew.forEach(numberNew=>console.log(numberNew));


}

// SPREAD OPERATOR
function arrayMerge(){
```

```
    var letters1 = ['a','b','c'];
    var letters2 = ['d','e','f'];
    var letters3 = [...letters1,...letters2];
    console.log(letters3);


}
```

MODIFICARE structures_journey.html

```
<html>


<body>


    <script src="functions.js">



    </script>

    <input type="button" value="array1" onclick="arrayBuild1()"></input><br>
    <input type="button" value="array2" onclick="arrayBuild2()"></input><br>
    <input type="button" value="array3" onclick="arrayBuild3()"></input><br>
    <input type="button" value="array4" onclick="arrayBuild4()"></input><br>
    <input type="button" value="array5" onclick="arrayMerge()"></input><br>



</body>



</html>
```

AGGIUNGERE LE FUNZIONE arrayCopy a functions.js

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark","Sql");
    courses.forEach(course=>console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course=>console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course=>console.log(course));



}

function arrayBuild4(){

    var numbers = [76,89,54,21];
    numbers.map(number=>console.log(number));
    var numbersNew = numbers.map(number=>(number*2));
    numbersNew.forEach(numberNew=>console.log(numberNew));


}

// SPREAD OPERATOR
function arrayMerge(){
```

```
    var letters1 = ['a','b','c'];
    var letters2 = ['d','e','f'];
    var letters3 = [...letters1,...letters2];
    console.log(letters3);


}

// SPREAD OPERATOR
function arrayCopy(){

    var letters1 = ['a','b','c'];
    var letters2 = [...letters1];
    console.log(letters2);


}
```

MODIFICARE structures_journey.html

```
<html>


<body>


    <script src="functions.js">



    </script>

    <input type="button" value="array1" onclick="arrayBuild1()"></input><br>
    <input type="button" value="array2" onclick="arrayBuild2()"></input><br>
    <input type="button" value="array3" onclick="arrayBuild3()"></input><br>
    <input type="button" value="array4" onclick="arrayBuild4()"></input><br>
    <input type="button" value="array5" onclick="arrayMerge()"></input><br>
    <input type="button" value="array6" onclick="arrayCopy()"></input><br>



</body>



</html>
```

AGGIUNGERE LE FUNZIONE arraySplit a functions.js PER FAR VEDERE COME FUNZIONA L'OPERATORE REST PER IL DESTRUCTURING DI UN ARRAY (REST OPERATOR INTRODOTTO NELLA VERSIONE 6 DI ECMA SCRIPT)

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark","Sql");
    courses.forEach(course=>console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course=>console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course=>console.log(course));



}

function arrayBuild4(){

    var numbers = [76,89,54,21];
    numbers.map(number=>console.log(number));
    var numbersNew = numbers.map(number=>(number*2));
    numbersNew.forEach(numberNew=>console.log(numberNew));


}

// SPREAD OPERATOR
function arrayMerge(){
```

```
    var letters1 = ['a','b','c'];
    var letters2 = ['d','e','f'];
    var letters3 = [...letters1,...letters2];
    console.log(letters3);


}

// SPREAD OPERATOR
function arrayCopy(){

    var letters1 = ['a','b','c'];
    var letters2 = [...letters1];
    console.log(letters2);


}

// REST OPERATOR (DESTRUCTURING)
function arraySplit(){


    var [letter1,letter2,...rest] =  ['a','b','c'];
    console.log(letter1);
    console.log(letter2);
    console.log(rest);


}
```

MODIFICARE IL CODICE DI structures_journey.html

```
<html>


<body>


    <script src="functions.js">



    </script>

    <input type="button" value="array1" onclick="arrayBuild1()"></input><br>
    <input type="button" value="array2" onclick="arrayBuild2()"></input><br>
```

```
    <input type="button" value="array3" onclick="arrayBuild3()"></input><br>
    <input type="button" value="array4" onclick="arrayBuild4()"></input><br>
    <input type="button" value="array5" onclick="arrayMerge()"></input><br>
    <input type="button" value="array6" onclick="arrayCopy()"></input><br>
    <input type="button" value="array7" onclick="arraySplit()"></input><br>



</body>



</html>
```

MODIFICARE IL CODICE DI json_objects per usare l'array nei json complessi

```
<html>

<body>

    <div id="p1"></div>
    <div id="p2"></div>
    <div id="p3"></div>


    <script>

        // SIMPLE JSON OBJECT
        var person1 = {

            "firstName": "Mario",
            "lastName": "Rossi",
            "age": 21

        }

        var person2 = {

            "firstName": "Giulia",
            "lastName": "Pisu",
            "age": 24,
            "auto": {

                "auto1": "Dacia",
                "auto2": "Fiat"

            }

        }
```

```
        var person3 = {

            "firstName": "Loris",
            "lastName": "Antonelli",
            "age": 29,
            "auto": ["Toyota","Renault"]

        }



        document.getElementById("p1").innerHTML = person1.firstName + " " +
person1.lastName + " " + person1.age;
        document.getElementById("p2").innerHTML = person2.firstName + " " +
person2.lastName + " " + person2.auto.auto1 + " " + person2.auto.auto2;
        document.getElementById("p3").innerHTML = person3.firstName + " " +
person3.lastName + " " + person3.auto[0] + " " + person3.auto[1];

    </script>


</body>

</html>
```

AGGIUNGERE LE FUNZIONE objectCopy1 a functions.js PER FAR VEDERE COME FUNZIONA LO SPREAD OPERATOR CON GLI OGGETTI (LO SPREAD OPERATOR NON CAMBIA GLI OGGETTI ORIGINARI!!!!)

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark", "Sql");
    courses.forEach(course => console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course => console.log(course));


}
```

```
function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course => console.log(course));



}

function arrayBuild4() {

    var numbers = [76, 89, 54, 21];
    numbers.map(number => console.log(number));
    var numbersNew = numbers.map(number => (number * 2));
    numbersNew.forEach(numberNew => console.log(numberNew));


}

// SPREAD OPERATOR
function arrayMerge() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = ['d', 'e', 'f'];
    var letters3 = [...letters1, ...letters2];
    console.log(letters3);


}

// SPREAD OPERATOR
function arrayCopy() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = [...letters1];
    console.log(letters2);


}

// REST OPERATOR (DESTRUCTURING)
function arraySplit() {


    var [letter1, letter2, ...rest] = ['a', 'b', 'c'];
    console.log(letter1);
```

```
    console.log(letter2);
    console.log(rest);


}

// SPREAD OPERATOR WITH OBJECTS
function objectCopy1() {

    var object1 = {

        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }

    var object3 = {...object1, ...object2};

   console.log(object3);
   console.log(object1);
   console.log(object2);


}
```

MODICARE IL CODICE DI structures_journey.html

```
<html>


<body>


    <script src="functions.js">


    </script>

    <input type="button" value="array1" onclick="arrayBuild1()"></input><br>
    <input type="button" value="array2" onclick="arrayBuild2()"></input><br>
    <input type="button" value="array3" onclick="arrayBuild3()"></input><br>
    <input type="button" value="array4" onclick="arrayBuild4()"></input><br>
    <input type="button" value="array5" onclick="arrayMerge()"></input><br>
```

```
    <input type="button" value="array6" onclick="arrayCopy()"></input><br>
    <input type="button" value="array7" onclick="arraySplit()"></input><br>
    <input type="button" value="array8" onclick="objectCopy1()"></input><br>




</body>



</html>
```

AGGIUNGERE LA FUNZIONE objectCopy2 a functions.js per far vedere come funziona Object assign (assign modifica l'oggetto originario)

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark", "Sql");
    courses.forEach(course => console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course => console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course => console.log(course));
```

```
}

function arrayBuild4() {

    var numbers = [76, 89, 54, 21];
    numbers.map(number => console.log(number));
    var numbersNew = numbers.map(number => (number * 2));
    numbersNew.forEach(numberNew => console.log(numberNew));


}

// SPREAD OPERATOR
function arrayMerge() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = ['d', 'e', 'f'];
    var letters3 = [...letters1, ...letters2];
    console.log(letters3);


}

// SPREAD OPERATOR
function arrayCopy() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = [...letters1];
    console.log(letters2);


}

// REST OPERATOR (DESTRUCTURING)
function arraySplit() {


    var [letter1, letter2, ...rest] = ['a', 'b', 'c'];
    console.log(letter1);
    console.log(letter2);
    console.log(rest);


}

// SPREAD OPERATOR WITH OBJECTS
function objectCopy1() {

    var object1 = {
```

```
        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }

    var object3 = {...object1, ...object2};

   console.log(object3);
   console.log(object1);
   console.log(object2);


}

// COPY OBJECT WITH ASSIGN
function objectCopy2() {

    var object1 = {

        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }

    var object3 = Object.assign(object1,object2);

   console.log(object3);
   console.log(object1);
   console.log(object2);


}
```

Cambiare il codice di structures_journey.html

```
<html>


<body>


    <script src="functions.js">


    </script>

    <input type="button" value="array1" onclick="arrayBuild1()"></input><br>
    <input type="button" value="array2" onclick="arrayBuild2()"></input><br>
    <input type="button" value="array3" onclick="arrayBuild3()"></input><br>
    <input type="button" value="array4" onclick="arrayBuild4()"></input><br>
    <input type="button" value="array5" onclick="arrayMerge()"></input><br>
    <input type="button" value="array6" onclick="arrayCopy()"></input><br>
    <input type="button" value="array7" onclick="arraySplit()"></input><br>
    <input type="button" value="array8" onclick="objectCopy1()"></input><br>
    <input type="button" value="array9" onclick="objectCopy2()"></input><br>







</body>


</html>
```

AGGIUNGERE LA FUNZIONE setBuild a functions.js

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark", "Sql");
    courses.forEach(course => console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course => console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course => console.log(course));



}

function arrayBuild4() {

    var numbers = [76, 89, 54, 21];
    numbers.map(number => console.log(number));
    var numbersNew = numbers.map(number => (number * 2));
    numbersNew.forEach(numberNew => console.log(numberNew));


}

// SPREAD OPERATOR
function arrayMerge() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = ['d', 'e', 'f'];
```

```
    var letters3 = [...letters1, ...letters2];
    console.log(letters3);


}

// SPREAD OPERATOR
function arrayCopy() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = [...letters1];
    console.log(letters2);


}

// REST OPERATOR (DESTRUCTURING)
function arraySplit() {


    var [letter1, letter2, ...rest] = ['a', 'b', 'c'];
    console.log(letter1);
    console.log(letter2);
    console.log(rest);


}

// SPREAD OPERATOR WITH OBJECTS
function objectCopy1() {

    var object1 = {

        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }

    var object3 = {...object1, ...object2};

   console.log(object3);
   console.log(object1);
   console.log(object2);
```

```
}

// COPY OBJECT WITH ASSIGN
function objectCopy2() {

    var object1 = {

        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }

    var object3 = Object.assign(object1,object2);

   console.log(object3);
   console.log(object1);
   console.log(object2);


}


function setBuild(){

    const mix = new Set();
    mix.add('a');
    mix.add('a');
    mix.add(3);
    mix.add('b');

    console.log(mix.has('a'));
    console.log(mix.size);
    console.log(mix.keys());

}
```

CREARE UN NUOVO FILE sets_maps_journey.html

```
<html>


<body>


    <script src="functions.js">


    </script>

    <input type="button" value="array1" onclick="setBuild()"></input><br>


</body>



</html>
```

AGGIUNGERE LA FUNZION mapBuild() al file functions.js (MAP E' LIKE SET PERCHE' HA LE STESSE FUNZIONI has(),size, keys....). L'ARRAY NON E' INVECE UNA STRUTTURA LIKE SET

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark", "Sql");
    courses.forEach(course => console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course => console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
```

```
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course => console.log(course));



}

function arrayBuild4() {

    var numbers = [76, 89, 54, 21];
    numbers.map(number => console.log(number));
    var numbersNew = numbers.map(number => (number * 2));
    numbersNew.forEach(numberNew => console.log(numberNew));


}

// SPREAD OPERATOR
function arrayMerge() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = ['d', 'e', 'f'];
    var letters3 = [...letters1, ...letters2];
    console.log(letters3);


}

// SPREAD OPERATOR
function arrayCopy() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = [...letters1];
    console.log(letters2);


}

// REST OPERATOR (DESTRUCTURING)
function arraySplit() {


    var [letter1, letter2, ...rest] = ['a', 'b', 'c'];
    console.log(letter1);
    console.log(letter2);
    console.log(rest);


}
```

```
// SPREAD OPERATOR WITH OBJECTS
function objectCopy1() {

    var object1 = {

        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }

    var object3 = {...object1, ...object2};

   console.log(object3);
   console.log(object1);
   console.log(object2);


}

// COPY OBJECT WITH ASSIGN
function objectCopy2() {

    var object1 = {

        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }

    var object3 = Object.assign(object1,object2);

   console.log(object3);
   console.log(object1);
   console.log(object2);
```

```
}


function setBuild(){

    const mix = new Set();
    mix.add('a');
    mix.add('a');
    mix.add(3);
    mix.add('b');

    console.log(mix.has('a'));
    console.log(mix.size);
    console.log(mix.keys());

}

function mapBuild(){

    const mix = new Map();
    mix.set(1,'a');
    mix.set(2,'a');
    mix.set(3,3);
    mix.set(4,'b');

    console.log(mix.has(1));
    console.log(mix.size);
    console.log(mix.keys());
    console.log(mix.values());

}
```

MODIFICARE IL CODICE di sets_maps_journey.html

```
<html>


<body>


    <script src="functions.js">


    </script>

    <input type="button" value="array1" onclick="setBuild()"></input><br>
    <input type="button" value="array2" onclick="mapBuild()"></input><br>


</body>
```

```
</html>
```

AGGIUNGERE LA FUNZIONE setMapBuild() al file functions.js per far vedere come si può fare la union di una set e di una map

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark", "Sql");
    courses.forEach(course => console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course => console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course => console.log(course));



}

function arrayBuild4() {

    var numbers = [76, 89, 54, 21];
    numbers.map(number => console.log(number));
    var numbersNew = numbers.map(number => (number * 2));
    numbersNew.forEach(numberNew => console.log(numberNew));
```

```
}

// SPREAD OPERATOR
function arrayMerge() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = ['d', 'e', 'f'];
    var letters3 = [...letters1, ...letters2];
    console.log(letters3);


}

// SPREAD OPERATOR
function arrayCopy() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = [...letters1];
    console.log(letters2);


}

// REST OPERATOR (DESTRUCTURING)
function arraySplit() {


    var [letter1, letter2, ...rest] = ['a', 'b', 'c'];
    console.log(letter1);
    console.log(letter2);
    console.log(rest);


}

// SPREAD OPERATOR WITH OBJECTS
function objectCopy1() {

    var object1 = {

        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }
```

```
    var object3 = {...object1, ...object2};

   console.log(object3);
   console.log(object1);
   console.log(object2);


}

// COPY OBJECT WITH ASSIGN
function objectCopy2() {

    var object1 = {

        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }

    var object3 = Object.assign(object1,object2);

   console.log(object3);
   console.log(object1);
   console.log(object2);


}


function setBuild(){

    const mix = new Set();
    mix.add('a');
    mix.add('a');
    mix.add(3);
    mix.add('b');

    console.log(mix.has('a'));
    console.log(mix.size);
    console.log(mix.keys());

}
```

```
function mapBuild(){

    const mix = new Map();
    mix.set(1,'a');
    mix.set(2,'a');
    mix.set(3,3);
    mix.set(4,'b');

    console.log(mix.has(1));
    console.log(mix.size);
    console.log(mix.keys());
    console.log(mix.values());

}

function setMapBuild(){

    const set = new Set();
    set.add('a');
    set.add('a');
    set.add(3);
    set.add('b');

    const map = new Map();
    map.set(1,'a');
    map.set(2,'a');
    map.set(3,3);
    map.set(4,'b');

   const newStucture = [...set,map.values()];
   console.log(newStucture);


}
```

modificare il codice di sets_maps_journey.html

```
<html>


<body>


    <script src="functions.js">


    </script>
```

```
    <input type="button" value="array1" onclick="setBuild()"></input><br>
    <input type="button" value="array2" onclick="mapBuild()"></input><br>
    <input type="button" value="array3" onclick="setMapBuild()"></input><br>


</body>



</html>
```

AGGIUNGERE LA FUNZIONE setArrayBuild() al file functions.js

```
function arrayBuild1() {

    var courses = ["JavaSE", "Angular", "React"];
    courses.push("Spark", "Sql");
    courses.forEach(course => console.log(course));

}


function arrayBuild2() {

    var courses = new Array();
    courses[0] = "JavaSE";
    courses[1] = "Angular";
    courses[2] = "React";
    courses.pop();
    courses.forEach(course => console.log(course));


}


function arrayBuild3() {

    var student1 = "Mario";
    var student2 = "Giulio";
    var average_rating = 22;
    var mix = ["Mario", "Giulio", average_rating];
    mix.forEach(course => console.log(course));



}

function arrayBuild4() {

    var numbers = [76, 89, 54, 21];
```

```
    numbers.map(number => console.log(number));
    var numbersNew = numbers.map(number => (number * 2));
    numbersNew.forEach(numberNew => console.log(numberNew));


}

// SPREAD OPERATOR
function arrayMerge() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = ['d', 'e', 'f'];
    var letters3 = [...letters1, ...letters2];
    console.log(letters3);


}

// SPREAD OPERATOR
function arrayCopy() {

    var letters1 = ['a', 'b', 'c'];
    var letters2 = [...letters1];
    console.log(letters2);


}

// REST OPERATOR (DESTRUCTURING)
function arraySplit() {


    var [letter1, letter2, ...rest] = ['a', 'b', 'c'];
    console.log(letter1);
    console.log(letter2);
    console.log(rest);


}

// SPREAD OPERATOR WITH OBJECTS
function objectCopy1() {

    var object1 = {

        name: "myName"


    }

    var object2 = {
```

```
        email: "myEmail"


    }

    var object3 = {...object1, ...object2};

   console.log(object3);
   console.log(object1);
   console.log(object2);


}

// COPY OBJECT WITH ASSIGN
function objectCopy2() {

    var object1 = {

        name: "myName"


    }

    var object2 = {

        email: "myEmail"


    }

    var object3 = Object.assign(object1,object2);

   console.log(object3);
   console.log(object1);
   console.log(object2);


}


function setBuild(){

    const mix = new Set();
    mix.add('a');
    mix.add('a');
    mix.add(3);
    mix.add('b');

    console.log(mix.has('a'));
```

```
    console.log(mix.size);
    console.log(mix.keys());

}

function mapBuild(){

    const mix = new Map();
    mix.set(1,'a');
    mix.set(2,'a');
    mix.set(3,3);
    mix.set(4,'b');

    console.log(mix.has(1));
    console.log(mix.size);
    console.log(mix.keys());
    console.log(mix.values());

}

function setMapBuild(){

    const set = new Set();
    set.add('a');
    set.add('a');
    set.add(3);
    set.add('b');

    const map = new Map();
    map.set(1,'a');
    map.set(2,'a');
    map.set(3,3);
    map.set(4,'b');

   const newStucture = [...set,map.values()];
   console.log(newStucture);


}

function setArrayBuild(){

    const array = new Array();
    array.push('a');
    array.push('b');

    const set = new Set();
    set.add('a');
    set.add('a');
    set.add(3);
```

```
    set.add('b');

   const newStucture = [...array,...set.keys()];
   console.log(newStucture);


}
```

modificare il codice di sets_maps_journey.html

```
<html>


<body>


    <script src="functions.js">


    </script>

    <input type="button" value="array1" onclick="setBuild()"></input><br>
    <input type="button" value="array2" onclick="mapBuild()"></input><br>
    <input type="button" value="array3" onclick="setMapBuild()"></input><br>
    <input type="button" value="array4" onclick="setArrayBuild()"></input><br>


</body>



</html>
```

creare il file var_let_constant.html per far vedere la differenza tra var let e const (PARLARE DI HOISTING)

```
<html>

<body>


<script>


 var firstName="Mario";
 let lastName="Rossi";


 if (firstName=="Mario"){


    var address = "Via Dei Mille";
    let ge=21;

 }

 console.log(firstName);
 console.log(lastName);
 console.log(address);
 console.log(age);



</script>




</body>









</html>
```

PARLARE DELL'OPERATORE == e ====

CLASSES OBJECTS

Creare il file classes_objects.html

```
<html>

<body>


    <div id="p1"></div>


    <script>


        class Person {



            constructor(firstName, lastName, age) {


                this.firstName = firstName;
                this.lastName = lastName;
                this.age = age;



            }


            eat() {

                console.log("eat");
            }

            speek() {

                console.log("speek");

            }



        }


     let person = new Person("Maria","Russo",21);
     person.address="Via dei Mille 00175 Roma";
```

```
     document.getElementById("p1").innerHTML=person.firstName + " "
+person.lastName + " " + person.age + " " +
     person.address;
     person.eat();
     person.speek();




    </script>


</body>




</html>
```

AGGIUNGERE UN ALTRO METODO eat nella CLASSE

```
<html>

<body>


    <div id="p1"></div>


    <script>


        class Person {


            constructor(firstName, lastName, age) {


                this.firstName = firstName;
                this.lastName = lastName;
                this.age = age;



            }
```

```
            eat() {

                console.log("eat");
            }

            speek() {

                console.log("speek");

            }

            eat() {

                console.log("eat very well");
            }



        }


        let person = new Person("Maria", "Russo", 21);
        person.address = "Via dei Mille 00175 Roma";
        document.getElementById("p1").innerHTML = person.firstName + " " +
person.lastName + " " + person.age + " " +
            person.address;
        person.eat();
        person.speek();




    </script>


</body>




</html>
```

AGGIUNGERE LA CLASSE Students Figlia di Person nel file classes_objects.html

```
<html>

<body>


    <div id="p1"></div>
    <div id="s1"></div>


    <script>


        class Person {



            constructor(firstName, lastName, age) {


                this.firstName = firstName;
                this.lastName = lastName;
                this.age = age;



            }


            eat() {

                console.log("eat");
            }

            speek() {

                console.log("speek");

            }

            eat() {

                console.log("eat very well");
            }



        }

        class Student extends Person {
```

```
            constructor(firstName, lastName, age, passportNumber) {

                super(firstName, lastName, age);
                this.passportNumber = passportNumber;




            }

            study() {

                console.log("study");
            }


        }


        let person = new Person("Maria", "Russo", 21);
        person.address = "Via dei Mille 00175 Roma";
        document.getElementById("p1").innerHTML = person.firstName + " " +
person.lastName + " " + person.age + " " +
            person.address;
        person.eat();
        person.speek();
        let student = new Student("Giulio", "Verdi", 24, "1028A");
        document.getElementById("s1").innerHTML = student.firstName + " " +
student.lastName + " " + student.age + " " +
            student.passportNumber;
        student.speek();
        student.eat();
        student.study();



    </script>


</body>
```

```
</html>
```

CONCETTO DI FUNZIONE DI CALLBACK E FUNZIONE ASINCRONA

Creare file callback.html

```
<html>

<body>

<script>

  const sayHello = () => {

    alert("say Hello");

  }

 const  asynchronousSayHello = () =>{

    setTimeout(sayHello,4000);

  }

</script>

<input type="button" value="sayHello" onclick="asynchronousSayHello()"></input>

</body>




</html>
```