

AUTOMATION COURSE 

JEST TESTING

ABEDALRAHEEM ALSAQQQA



CMD Commands

1- MKDIR : create folder

2- CD to project : to go the folder that we created

3- check if we have node js installed if not will go to the official website and download it

<https://nodejs.org/en/download/>

4- check node version by the command node -v





the inside the folder we create we will do

A- npm init

b - give a name for the project eg :- Jest

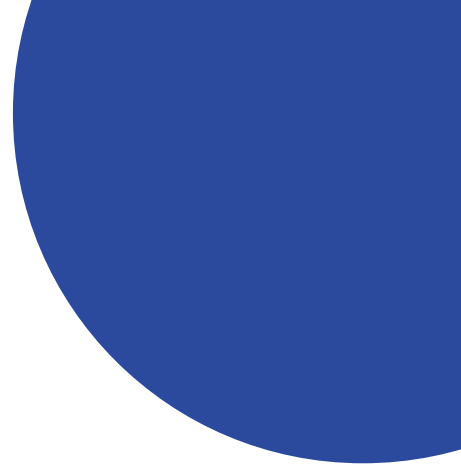
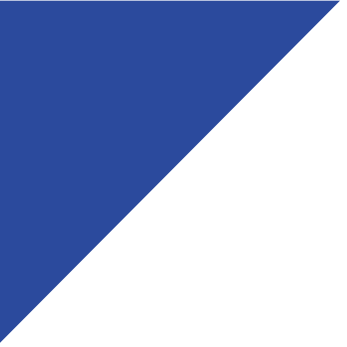
c- keep the version as it is

d- other fields just press enter


E - press yes



f- npm l jest --save-dev





inside package.json
we will update the test to "jest --watchAll"
instead of echo\"error etc..
install jest extension inside vs - code





```
function sayhello (){  
    return 'hello'  
}
```

```
module.exports=sayhello;
```

inside say.js





```
const say = require ('./say')
```

```
test("testsayfunctionmsg",()=>{  
  expect(say()).toBe("hello")  
  },)
```



inside
say.test.js



another example

```
const thesum = (num)=>{  
  return num ;  
}
```

```
module.exports=thesum;
```



inside sum.js



```
const myTesting = require("./sum");
```

```
  test("return the num",()=>{  
    expect(myTesting(10)).toBe(10);  
  })
```



```
inside sum.test.js
```


another example

```
const thesum = (num,num2)=>{  
    return num+num2;  
}
```

```
module.exports=thesum;
```

inside sum.test.js

```
const myTesting = require("./sum");  
  
test("return the num1+num2",()=>{  
    expect(myTesting(25,15)).toBe(40);  
})
```

inside sum.test.js

Truthy and Falsy

```
const thesum = (num,num2)=>{  
  return num+(num2 || null);  
}
```

```
module.exports=thesum;
```

```
const thesum = (num,num2,num3)=>{  
  return num+(num2 || null)+(num3 || null) ;  
}
```

inside sum.js

```
test("return the num1+num2+number3",()=>{  
    expect(myTesting(25,15,20)).toBe(60);  
})
```

inside sum.test.js

```
// fifth test case  
test("return 0 if no number",()=>{  
  
    expect(myTesting()).toBe(0)  
})
```

if user didnt enter any value what will happen it will return 0 and if we use the what inside next slide this will fail

if we dont know the total numbers

```
function thesum(...numbers) {  
  return numbers.reduce((pv,cv)=>pv+cv) ;  
}
```

reduce take two parameter

cv: current value

pv : previous value

```
test("return 0 if no number",()=>{  
  expect(myTesting()).toBe(0)  
})
```

inside sum.js this will fail

see next slide to see how to make it pass


```
function thesum(...numbers) {  
  return numbers.reduce((pv,cv)=>pv+cv,0) ;  
}
```

the 0 here is the initial value
this means that if no previous and no
current value this will bring 0 for me

describe ("the name of the suite",()=>{

all of our tests

})

```
describe("",()=>{
  test("return the num", () => {
    expect(myTesting(10)).toBe(10);
  });
  test("return 0 if no number", () => {
    expect(myTesting()).toBe(0);
  });

  test("return the num+numb2", () => {
    expect(myTesting(10, 20)).toBe(30);
  });

  test("return the num+numb2+num3", () => {
    expect(myTesting(10, 20, 50)).toBe(80);
  });

  test("return the num+numb2+num3+numb4", () => {
    expect(myTesting(10, 20, 50, 100)).toBe(180);
  });

})
```

```
describe("",()=>{  
  test("return the num", () => {  
    expect(myTesting(10)).toBe(10);  
  });  
  test("return 0 if no number", () => {  
    expect(myTesting()).toBe(0);  
  });  
})
```

```
function namefilter (name){  
  
    if (name === undefined) {  
        name = "unknown"  
    }  
    return name  
}  
  
module.exports=namefilter;
```

input.js

```
const filtername = require("./input")

describe("filter the name only",()=>{
  test("check if the name is null",()=>{
    expect (filtername()).toBe("unknown")
  })
})
```

```
function namefilter (name){
```

```
    if (name === undefined) {  
        name = "unknown"  
    }
```

```
    if (name.startsWith(" ") || name.endsWith(" ")) {  
        name=name.trim();  
    }
```

```
    return name  
}
```

```
const filtername = require("./input")
```

```
describe("filter the name only",()=>{  
  test("check if the name is null",()=>{  
    expect (filtername()).toBe("unknown")  
  });
```

```
    test("check for spaces",()=>{  
      expect(filtername("abed ")).toBe("abed")  
    })  
  })
```



```
const filtername = require("./input")
```

```
describe("filter the name only",()=>{  
  test("check if the name is null",()=>{  
    expect (filtername()).toBe("unknown")  
  });
```

```
    test("check for spaces",()=>{  
      expect(filtername("abed ")).toBe("abed")  
    })
```

```
    test("check if the name more than 10 letters",()=>{  
      expect(filtername("abedalraheem")).toBe("abedalrahe")  
    })  
  })
```

```
function namefilter (name){  
  
    if (name === undefined) {  
        name = "unknown"  
    }  
  
    if (name.startsWith(" ") || name.endsWith(" ")) {  
        name=name.trim();  
    }  
  
    if(name.length > 10){  
        name = name.substring(0,10)  
    }  
    return name  
}  
  
module.exports=namefilter;
```

```
test.skip("test the returning of one number", ()=>{
```

```
    expect(functionformath(10)).toBe(10)  
    })
```

```
test.only("if a negative number is entered return positive  
one", ()=>{
```

```
    expect(functionformath(-800)).toBe(800)  
    })
```

matchers

```
const myarray = ["heba", "ahmad", "ali", "rama",  
                 "nada", "omar", "raghad"];
```

```
module.exports = myarray;
```

inside data.js

```
const myDataCheck = require("../Data");
```

```
test("to check if the length of my array is 7 ", () => {  
    expect(myDataCheck.length).toBe(7);  
});
```

inside data.test.js

```
test("to check if the length of my array is 7 ", () => {  
    expect(myDataCheck).toHaveLength(7);  
});
```

inside data.test.js

```
test("to check if my array contains the name of  
    ahmad", () => {  
    expect(myDataCheck).toContain("ahmad");  
});
```

inside data.test.js


```
test("to check if my array dose not contains the  
      name abed", () => {  
    expect(myDataCheck).not.toContain("abed");  
});
```

inside data.test.js

```
test(" same as pervious one to check if my array  
dose not contains the name abed", () => {  
  for (let i = 0; i < myDataCheck.length; i++) {  
    expect(myDataCheck).not.toContain("abed");  
  }  
});
```

inside data.test.js

```
test(" tocheck if my array dosent have any number inside it  
    ", () => {  
    for (let i = 0; i < myDataCheck.length; i++) {  
        expect(isNaN(myDataCheck[i])).toBe(true)  
    }  
});
```

inside data.test.js

```
test(" same as pervious one to check if my array dose not  
contains the name abed", () => {  
  for (let i = 0; i < myDataCheck.length; i++) {  
    expect(isNaN(myDataCheck[i])).toBeTruthy()  
  }  
});
```

inside data.test.js

```
test(" same as pervious one to check if my array dose not  
contains the name abed", () => {  
  for (let i = 0; i < myDataCheck.length; i++) {  
    expect(isNaN(myDataCheck[i])).toBeTruthy()  
  }  
});
```

inside data.test.js

```
test(" same as pervious one to check if my array dose not  
contains the name abed", () => {  
  for (let i = 0; i < myDataCheck.length; i++) {  
    expect(isNaN(myDataCheck[i])).toBeFalsy()  
  }  
});
```

inside data.test.js

```
const myarray = [3,"heba","ahmad", "ali", "rama", "nada",  
                 "omar", "raghad"];
```

```
module.exports = myarray;
```

inside data.js

```
test(" check if my array first element is larger than 1", () => {  
    for (let i = 0; i < myDataCheck.length; i++) {  
        expect(myDataCheck[0]).toBeGreaterThan(1)  
    }  
});
```

inside data.test.js


```
test(" check if my array first element is larger than 1", () => {  
    for (let i = 0; i < myDataCheck.length; i++) {  
        expect(myDataCheck[0]).toBeLessThanOrEqual(1)  
    }  
});
```

```
test(" check if my String have substring inside it", () => {  
    let fullName = "ahmad Mohammad ali"  
  
    expect(fullName).toMatch(/ahmad/)  
});
```

```
test("check if my info have the property age", () => {  
    let myInfo = {  
        name: "abedairaheem",  
        age: 29,  
        nationality: "jordanian",  
    };  
  
    expect(myInfo).toHaveProperty("name");  
});
```

```
test("check if my info have the property age", () => {  
    let myInfo2 = {  
        name: "abedatraheem",  
        age: 29,  
        nationality: "jordanian",  
    };  
  
    expect(myInfo2).toHaveProperty("age",29);  
});
```