**Usando Jest**

Documentacion:

<https://jestjs.io/docs/api>

|  |
| --- |
| **Iniciar Depuracion**  Podemos iniciar una depuracion en VSC yendo a Ejecutar -> Iniciar depuracion    Abrira una ventana de chrome, tambien creara una carpeta .vscode con un **launch.json.**    En este launch.json tenemos informacion sobre el proyecto, por ejemplo tenemos el “url” que indicaremos la ruta para que encuentre el proyecto que necesita abrir el depurador.    En archivo **package.json** y en la **“key”** que se llama **“test”** le ponemos como valor entre comillas **“jest”**, esto por convencion de la librería jest.    vamos al archivo **package.json** y en la **“key”** que se llama **“test”** le ponemos como valor entre comillas **“jest”**, esto por convencion de la librería jest.    vamos a la terminal y ejecutamos la siguiente linea de comandos:  checkear que estabamos en la carpeta correcta y se instalo correctamente, podemos checkear en que directorio estamos con pwd en bash o dir en powershell para saber nuestra posicion.  para saber s esta instalado, vemos si aparece como dependencia en package.json    Ahora necesitamos instalar todas las dependencias del proyecto con el comando    Una vez efectuamos los cambios/correcciones en el codigo, para recompilarlo ejecutamos el siguiente comando.    Nos va a informar donde se realizaron cambios. |

|  |
| --- |
| **Carpeta del test y archivo para la prueba de componente.**  Una vez iniciada la depuracion del proyecto, tenemos que crear una carpeta llamada **\_\_test\_\_** y dentro de ella ira el archivo donde escribiremos los test de **Jest**. Tenemos que llamar al ultimo archivo de la rama llamado, en el siguiente ejemplo es **until.js**. El nombre del archivo dentro de la carpeta \_\_test\_\_ sera **until.test.js**  esta sera nuestra prueba de componente  Tiene que estar en el mismo nivel del proyecto, no dentro de otra carpeta. |
| **Npm run test** o **npm test**  Con este comando iniciaremos a correr la prueba de componente. En el siguiente caso fallara por que no tenemos aun ninguna prueba escrita. |
| **Require**  Sirve para importar desde otros archivos, en el siguiente ejemplo importamos **generateText** que esuna funcion que retorna el nombre y la edad que ingresan como parametros. |
| **describe(name, fn)**  **describe** crea un bloque que agrupa diferentes test, no es obligatorio hacerlo de esta forma, podemos escribir bloques de test directamente en el nivel superior, pero se prefiere de esta manera para poder organizar los test en grupos.  Recibe por parametro el nombre de la agrupacion y una funcion callback.  const myBeverage = {      delicious: true,      sour: false,    };      describe('my beverage', () => {      test('is delicious', () => {        expect(myBeverage.delicious).toBeTruthy();      });        test('is not sour', () => {        expect(myBeverage.sour).toBeFalsy();      });    });  <https://jestjs.io/docs/api#describename-fn> |
| **test(name, fn, timeout)**  Es el metodo con el cual corremos los test, por ejemplo digamos que tenemos una funcion **inchesOfRain()** que deberia ser **cero**.  El test completo seria:  test('did not rain', () => {      expect(inchesOfRain()).toBe(0);    });  El primer argumento del metodo es el nombre, el segundo es una funcion callback que contiene la expectacion del test. El tercero es opcional, un timeout en milisegundos para especificar cuanto tiempo deberia esperar antes de abortar.  *NOTA: el tiempo default para el timeout es de 5 segundos.*  *NOTA2: si el retorno del test es una promesa, Jest esperara que la promesa se resuelva antes de que se complete el test. Documentacion para testear codigo asyncronico:* [*https://jestjs.io/docs/asynchronous#callbacks*](https://jestjs.io/docs/asynchronous#callbacks)  [*https://jestjs.io/docs/api#testname-fn-timeout*](https://jestjs.io/docs/api#testname-fn-timeout) |
| **Expect**  Cuando escribimos Tests, frecuentemente necesitamos checkear los valores, **expect** nos da acceso a un numero de “**matchers”** que nos permitira validar diferentes cosas.  Ctrl+click para abrir hipervinculo a cada matcher.   * [expect(value)](https://jestjs.io/docs/expect#expectvalue) * [expect.extend(matchers)](https://jestjs.io/docs/expect#expectextendmatchers) * [expect.anything()](https://jestjs.io/docs/expect#expectanything) * [expect.any(constructor)](https://jestjs.io/docs/expect#expectanyconstructor) * [expect.arrayContaining(array)](https://jestjs.io/docs/expect#expectarraycontainingarray) * [expect.assertions(number)](https://jestjs.io/docs/expect#expectassertionsnumber) * [expect.hasAssertions()](https://jestjs.io/docs/expect#expecthasassertions) * [expect.not.arrayContaining(array)](https://jestjs.io/docs/expect#expectnotarraycontainingarray) * [expect.not.objectContaining(object)](https://jestjs.io/docs/expect#expectnotobjectcontainingobject) * [expect.not.stringContaining(string)](https://jestjs.io/docs/expect#expectnotstringcontainingstring) * [expect.not.stringMatching(string | regexp)](https://jestjs.io/docs/expect#expectnotstringmatchingstring--regexp) * [expect.objectContaining(object)](https://jestjs.io/docs/expect#expectobjectcontainingobject) * [expect.stringContaining(string)](https://jestjs.io/docs/expect#expectstringcontainingstring) * [expect.stringMatching(string | regexp)](https://jestjs.io/docs/expect#expectstringmatchingstring--regexp) * [expect.addSnapshotSerializer(serializer)](https://jestjs.io/docs/expect#expectaddsnapshotserializerserializer) * [.not](https://jestjs.io/docs/expect#not) * [.resolves](https://jestjs.io/docs/expect#resolves) * [.rejects](https://jestjs.io/docs/expect#rejects) * [.toBe(value)](https://jestjs.io/docs/expect#tobevalue) * [.toHaveBeenCalled()](https://jestjs.io/docs/expect#tohavebeencalled) * [.toHaveBeenCalledTimes(number)](https://jestjs.io/docs/expect#tohavebeencalledtimesnumber) * [.toHaveBeenCalledWith(arg1, arg2, ...)](https://jestjs.io/docs/expect#tohavebeencalledwitharg1-arg2-) * [.toHaveBeenLastCalledWith(arg1, arg2, ...)](https://jestjs.io/docs/expect#tohavebeenlastcalledwitharg1-arg2-) * [.toHaveBeenNthCalledWith(nthCall, arg1, arg2, ....)](https://jestjs.io/docs/expect#tohavebeennthcalledwithnthcall-arg1-arg2-) * [.toHaveReturned()](https://jestjs.io/docs/expect#tohavereturned) * [.toHaveReturnedTimes(number)](https://jestjs.io/docs/expect#tohavereturnedtimesnumber) * [.toHaveReturnedWith(value)](https://jestjs.io/docs/expect#tohavereturnedwithvalue) * [.toHaveLastReturnedWith(value)](https://jestjs.io/docs/expect#tohavelastreturnedwithvalue) * [.toHaveNthReturnedWith(nthCall, value)](https://jestjs.io/docs/expect#tohaventhreturnedwithnthcall-value) * [.toHaveLength(number)](https://jestjs.io/docs/expect#tohavelengthnumber) * [.toHaveProperty(keyPath, value?)](https://jestjs.io/docs/expect#tohavepropertykeypath-value) * [.toBeCloseTo(number, numDigits?)](https://jestjs.io/docs/expect#tobeclosetonumber-numdigits) * [.toBeDefined()](https://jestjs.io/docs/expect#tobedefined) * [.toBeFalsy()](https://jestjs.io/docs/expect#tobefalsy) * [.toBeGreaterThan(number | bigint)](https://jestjs.io/docs/expect#tobegreaterthannumber--bigint) * [.toBeGreaterThanOrEqual(number | bigint)](https://jestjs.io/docs/expect#tobegreaterthanorequalnumber--bigint) * [.toBeLessThan(number | bigint)](https://jestjs.io/docs/expect#tobelessthannumber--bigint) * [.toBeLessThanOrEqual(number | bigint)](https://jestjs.io/docs/expect#tobelessthanorequalnumber--bigint) * [.toBeInstanceOf(Class)](https://jestjs.io/docs/expect#tobeinstanceofclass) * [.toBeNull()](https://jestjs.io/docs/expect#tobenull) * [.toBeTruthy()](https://jestjs.io/docs/expect#tobetruthy) * [.toBeUndefined()](https://jestjs.io/docs/expect#tobeundefined) * [.toBeNaN()](https://jestjs.io/docs/expect#tobenan) * [.toContain(item)](https://jestjs.io/docs/expect#tocontainitem) * [.toContainEqual(item)](https://jestjs.io/docs/expect#tocontainequalitem) * [.toEqual(value)](https://jestjs.io/docs/expect#toequalvalue) * [.toMatch(regexp | string)](https://jestjs.io/docs/expect#tomatchregexp--string) * [.toMatchObject(object)](https://jestjs.io/docs/expect#tomatchobjectobject) * [.toMatchSnapshot(propertyMatchers?, hint?)](https://jestjs.io/docs/expect#tomatchsnapshotpropertymatchers-hint) * [.toMatchInlineSnapshot(propertyMatchers?, inlineSnapshot)](https://jestjs.io/docs/expect#tomatchinlinesnapshotpropertymatchers-inlinesnapshot) * [.toStrictEqual(value)](https://jestjs.io/docs/expect#tostrictequalvalue) * [.toThrow(error?)](https://jestjs.io/docs/expect#tothrowerror) * [.toThrowErrorMatchingSnapshot(hint?)](https://jestjs.io/docs/expect#tothrowerrormatchingsnapshothint) * [.toThrowErrorMatchingInlineSnapshot(inlineSnapshot)](https://jestjs.io/docs/expect#tothrowerrormatchinginlinesnapshotinlinesnapshot)   En el siguiente ejemplo se usa el matcher .toBe() para esperar que el resultado de la funcion sea 0.  test('did not rain', () => {      expect(inchesOfRain()).toBe(0);    }); |
| **Escribiendo una prueba**  1. Primero declaramos las constantes que vamos a utlizar y le indicamos con un require donde estan unicadas estas constantes    2. Hacemos un describe en el cual agrupamos las test que querramos, en este caso, pruebas referentes a la salida de datos.    3. escribimos el test que llevaremos a cabo, generamos un text para ser probado declarando una constante en la cual almacenamos el resultado de la funcion    4. usamos el metodo expect y le introducimos por parametros el valor de la funcion ejectuada anteriormente, luego usamos el matcher .toBe() y introducimos lo que esperamos que de como resultado, el toBe matchea “lo que deberia ser”.    5. Con el test escrito, podemos correr la prueba escribiendo por consola **npm start** y nos dara los resultados de la prueba.    Como podemos observar, la prueba es excitosa. Tambien podemos observar aquí como se indica el grupo de pruebas y la prueba ejecutada (argumentos name de los metodos)  **Npm run test:coverage**  Con este comando en consola, traemos la covertura de nuestro unit test.    En este caso nos tira un error por que Jest tiene otro requisito por convencion. Necesitamos agregar en el package.json una linea despues de “test”:”jest”,  Que sea:  **“test:coverage”: “jest - -coverage”**    Podemos observar que tenemos 26.66% de statements y lineas (sentencias), 0% en branch (no tenemos ninguna decisión) y 33.33%, es decir que una funcion la leyo y las otras 2 no, fijemonos que tenemos importadas 3 funciones originalmente.    Si se ejecuto la cobertura, nos creara una carpeta **coverage** en el proyecto  Dentro podremos encontrar el reporte y abrirlo en un index.html que se ve de forma mas estilizada    **Tambien podemos ingresar al link de la columna “file” y entrar a los detalles de la cobertura.**  **Todo lo que esta en rojo es lo que se puede aumentar a nivel de cobertura. Cuando hablamos de esto es calidad en el codigo.**  **Calidad/eficiencia** |
|  |