Programación con JavaScript II

Sesión sincrónica 8

Pruebas

Bienvenida y actividad de bienestar

Duración: 10 minutos.

Nombre de la práctica: Encuentra tu pasión con 3 preguntas
Descripción de la práctica: Hoy en día se nos pide seguir nuestra pasión, pero ¿y si no sabemos cuál es? Con las siguientes preguntas podrás identificarlas.
Palabras clave: Preguntas, pasión, felicidad
Instrucciones para el aprendedor:

Ejercicio 1
Responde en una hoja estas preguntas:

1. **¿Qué creencia tienes que casi nadie comparte contigo?**
   Esta pregunta de Peter Thiel, co-fundador de PayPal y jefe del Thiel Foundation, está diseñada para dos cosas, para ayudarte a identificar qué es importante para ti y para determinar si vale la pena perseguirlo por su originalidad.

   Si encuentras un problema o reto que nadie más está abordando puedes hacer tu propio nicho y crear valor. Aunque nos enseñan a hacer lo que otros están haciendo y luego competir para ganarles, esto equivale a golpear tu cabeza contra la pared, en lugar de atravesar la puerta abierta en la que nadie se fija.

2. **¿Cuáles son tus súper-poderes?**
   Esta pregunta de Yamashita se refiere a “desempacar la combinación de rasgos de personalidad y aptitudes que traes sin esfuerzo a cualquier situación. La cineasta Tiffany Shlain del Moxie Institute también explora las fortalezas y súper-poderes naturales en su nuevo film que sugiere que si podemos identificar nuestras fortalezas de carácter y construir a partir de ellas podemos llevar vidas más felices y exitosas. Si desconoces las tuyas, puedes hacer el Test Via de Fortalezas de carácter en forma gratuita, en este sitio: https://www.viacharacter.org/survey/account/register

3. **¿Qué disfrutabas hacer a los diez años?**
   Robert Rubin, exsecretario del Tesoro de Estados Unidos dice que, en ocasiones, voltear al pasado te permite una mirada de quién eres realmente y lo que amabas hacer, antes de que otros te dijeran lo que tienes que hacer. Eric Maisel, psicoterapeuta y autor concuerda agregando que las cosas que amábamos de niños son probablemente las cosas que seguimos amando. Sugiere hacer una lista de las actividades favoritas e intereses de la niñez y ver que resuena hoy en día.

Fuente: https://www.fastcompany.com/3028946/find-your-passion-with-these-8-thought-provoking-questions

Actividad guiada
Parte 1

Duración: 75 minutos.

Contesta las siguientes preguntas antes de iniciar con los ejercicios de práctica:

1. ¿Qué tipo de pruebas se pueden realizar a un código?
2. Explica qué es JEST y por qué nos interesaría utilizarla.
3. ¿Qué es un matcher?

Antes de iniciar los ejercicios del tema 8, asegúrate de que JEST se encuentre adecuadamente configurado en Visual Studio Code, revisando que en el archivo package.json se incluya la siguiente instrucción, la cual permite generar un reporte de cobertura:

"scripts": { "test"= "jest –coverage"},

Ejercicio 1

1. Revisa el siguiente código y observa la serie de pruebas que realiza a una operación de suma simple.
2. Completa los espacios del código para que cada una de las pruebas sea exitosa.

```
// Ejercicio 1
describe('Ejercicio 1', () => {
    test('Pruebas', () => {
      const value = 2 + 2;
      expect(value).toBeGreaterThan(_);
      expect(value).toBeGreaterThanOrEqual(_);
      expect(value).toBeLessThan(_);
      expect(value).toBeLessThanOrEqual(_);
      // toBe y toEqual son equivalentes para números
      expect(value).toBe(_);
      expect(value).toEqual(_);
    });
  });
```

Actividad guiada
Parte 1

3. Incluye un par de pruebas en el código anterior que permitan realizar evaluaciones del código *string* con los **matchers**:
   not.toMatch(string)
   toMatch(string)

4. Genera el reporte de cobertura desde la terminal y desde el archivo index.html.

Ejercicio 2

1. Crea cuatro funciones en un archivo llamado **index.js** que realicen las funciones básicas de sumar, restar, multiplicar y dividir.

```
const sumar = (a, b) => a + b;
const restar = (a, b) => _____;
const multiplicar = (a, b) => _____;
const dividir = (a, b) => _____;
module.exports ={
   sumar,
   restar,
   multiplicar,
   dividir
};
```

Actividad guiada
Parte 1

2. Crea un archivo llamado index.test.js con el que realizarás las pruebas del código, asegurándote de que se encuentre en la misma carpeta. Copia el siguiente código y completa las líneas indicadas para realizar las pruebas necesarias.

```
/* importamos las funciones de index.js */
var {sumar, restar, multiplicar, dividir} = require('./index');

 // Con describe describimos un conjunto de pruebas a realizar
 describe('Operaciones matemáticas', () => {

    // Probamos cada uno de los test
   test('Probamos la suma', () => {
     expect(sumar(1, 1)).toBe(2);
   });
   test('Probamos la resta', () => {
     ____________________________________
   });
   test('Probamos la multiplicación', () => {
     ____________________________________   });
   test('Probamos la división', () => {
     ____________________________________
   });
 });
```

3. Ejecuta el reporte npm test desde la terminal de Visual Studio Code. Haz los ajustes necesarios tanto al archivo **index.js** como al archivo **index.test.js** para asegurarte de que la prueba sea exitosa.

Actividad guiada
Parte 1

Ejercicio 3

1. Analiza el siguiente código. La primera parte describe el archivo **arrays.js**, mientras que la segunda se encuentra en **arrays.test.js**.
   a) Explica la prueba realizada al arreglo “estados”.
   b) ¿Cuál es la prueba que pretende hacer al incluir new Set(estados)?

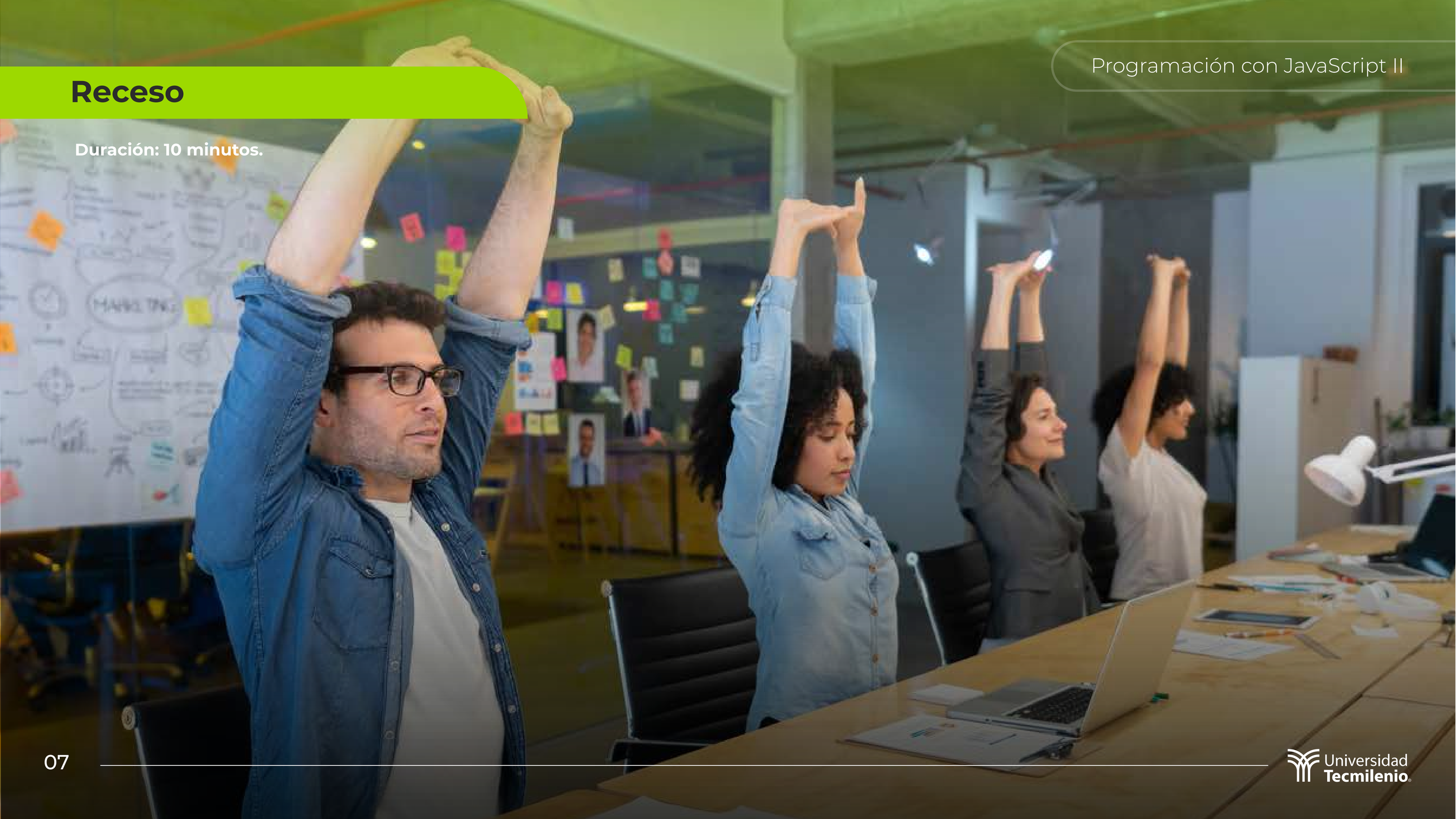
```
// arrays.js
const estados = ['Nuevo León', 'Aguascalientes', 'Colima', 'Sinaloa', 'Durango', 'Sonora', 'Chihuahua', 'Veracruz', 'Nayarit'];
const dias = ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sábado', 'Domingo'];
 module.exports ={
   estados,
   dias,
};

//arrays.test.js
var {estados, dias} = require('./arreglos');
test('Revisamos el contenido de los arreglos', () => {
   expect(estados).toContain('Nuevo León');
   expect(new Set(estados)).toContain('Nuevo León');
 });
```

Consideraciones:
Instructor:
Al realizar estos ejercicios es necesario guiar al aprendedor en la configuración adecuada de la aplicación Visual Studio Code.
Configurar los reportes de cobertura ayudará a comprobar los resultados de las pruebas.
Aprendedor:
Considera usar la extensión JEST con Visual Studio Code. Si llegas a tener algún problema con su configuración, puedes apoyarte de tu instructor.
Te recomendamos tener una lista de “matchers” a la mano con el fin de que puedas hacer otro tipo de pruebas con el código propuesto.

Receso

Duración: 10 minutos.



Actividad guiada
Parte 2

Duración: 75 minutos.

Ejercicio 4

1. Analiza el siguiente código y explica cuál es el objetivo de la prueba.
2. Genera el reporte de cobertura e identifica el porcentaje de aprobación.
3. ¿A qué se debe que la prueba falle?
4. ¿Qué líneas de código cambiarías para que la prueba funcione?
5. Si se agregara el idioma francés con el valor ***"fr-FR"*** como un tercer lenguaje en una página web, ¿qué líneas agregarías y cuál sería la prueba?
6. Genera el reporte de cobertura e identifica el porcentaje de las funciones que pasaron la prueba.

El siguiente código se encuentra en el archivo **lenguaje.js**.

```
const idiomaIngles = "en-US";
const idiomaEspanol = "es-ES";
function lenguajepagina(language){
    switch (language.toLowerCase()){
      case idiomaIngles.toLowerCase():
        return '/about-us';
      case idiomaEspanol.toLowerCase():
        return '/acerca-de-nosotros';
    }
    return '';
}
module.exports = lenguajepagina;
```

Actividad guiada Parte 2

En el archivo **lenguaje.test.js** se encuentran las siguientes líneas:

```
const lenguajepagina = require("./lenguaje");
test("Revisamos el valor de retorno del lenguaje seleccionado”, () => {
    expect(lenguajepagina("es-ES")).toBe("/about-us");
});
```

Ejercicio 5

El siguiente código se encuentra en el archivo **objectMapping.js**.

```
const objectMapping = {
    ACTIVO: 'Activo',
    INACTIVO: 'Inactivo',
    OBSOLETO: 'Obsoleto',
};

function getObjectDescription(tipo) {
    if (!tipo) {
        return "El parámetro 'tipo' no existe";
    }
    return objectMapping[tipo];
}

module.exports = getObjectDescription;
```

Actividad guiada
Parte 2

1. Crea un archivo **objectMapping.test.js** con las siguientes pruebas:
   a. Garantiza que “getObjectDescription” exista.
   b. Usa el método “getObjectDescription” con un parámetro que existe en el objeto “objectMapping”.
   c. Usa el método “getObjectDescription” con un parámetro que no exista en el “objectMapping”.
   d. Usa el método “getObjectDescription” con un parámetro vacío.
   e. Usa el método “getObjectDescription” sin parámetro.

Ejercicio 6

El siguiente código pertenece al archivo **valoresnumericos.js**.

```
const suma = (vals) => {
    let suma = 0;
    vals.forEach((val) => {
        suma += val;
    });
    return suma;
}
const positivos = (valores) => {
    return valores.filter((x) => { return x > 0; });
}
const negativos = (valores) => {
    return valores.filter((x) => { return x < 0; });
}
module.exports = { suma, positivos, negativos };
```

Actividad guiada
Parte 2

1. Escribe las pruebas que realizarías para saber si el código devuelve los valores correctos.
2. Escribe el código JEST en un archivo llamado **valoresnumericos.test.js** y evalúa las pruebas que diseñaste en el paso anterior.
3. ¿Qué cambiarías en el código original para saber si son números primos?
4. Realiza las pruebas necesarias para garantizar si son correctos.
5. Demuestra que tus pruebas obtienen un 100% en el reporte de cobertura de JEST.

Consideraciones:

Instructor:
Considera realizar los ejercicios con anticipación y ofrecer a los aprendedores una alternativa de solución para cada uno de los ejercicios.

Aprendedor:
Recuerda que una buena práctica es separar el código fuente del código de prueba, por lo que se sugiere que las líneas de código JEST estén en un archivo con el sufijo test.js.

Cierre

Duración: 10 minutos.

En esta práctica comenzaste a familiarizarte con las funciones básicas de JEST para probar código de JavaScript, lo que constituye una gran herramienta de apoyo para que las pruebas unitarias garanticen la calidad del software. Además, los reportes de cobertura obtenidos durante este ejercicio constituyen una evidencia adecuada de la fase de *testing* para el personal que conforma el equipo de desarrollo.