



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 01. **FUNDAMENTOS DEL LENGUAJE**

Desarrollo Web en entorno cliente
CFGs DAW

Ejercicios de repaso de programación

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2019/2020

Versión:210923.1338

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 1.....	3
2. Ejercicio 2.....	3
3. Ejercicio 3.....	4

UD01. FUNDAMENTOS DEL LENGUAJE

EJERCICIOS DE REPASO DE PROGRAMACIÓN

Estos ejercicios son para que repases tus conocimientos de programación. Si te cuesta hacerlos, necesitarás dedicarle algo de tiempo extra a consolidar los conocimientos para tener más garantías de aprobar este módulo: ponte en contacto conmigo para que te recomiende material adicional.

Los ejercicios los puedes realizar en cualquier lenguaje de programación (el que más te guste o alguno que quieras probar): Java, Ruby, Python, Rust, Javascript, Go, Piet...

1. EJERCICIO 1

Dada una serie de números, dividirla en dos series diferentes: una con los números más grandes y otra con los números más pequeños.

Por ejemplo:

Serie: 1, 5 ,8,40, 100, -3, 2.5, 3000
Salida

Pequeños: -3, 1, 2.5, 5
Grandes: 8, 40, 100, 3000

Puedes representar la serie como una constante, no hace falta que el programa lea los datos. No importa el orden en el que estén los números en las dos series que construyas.

2. EJERCICIO 2

Prepara un programa para el que, dado un número, produzca como salida una versión de la letra de la canción de "un limón, y medio limón" que acabe con el número de limones indicado (y medio limón)

Por ejemplo:

Entrada: 4
Salida:
1 limón, y medio limón
2 limones, y medio limón
3 limones, y medio limón
iiiY 4 LIMONES Y MEDIO LIMÓN!!!

Ojo a las mayúsculas de la última estrofa.

Si quieres puedes intentar la versión avanzada, en la que el programa debería imprimir:

Un limón, y medio limón
Dos limones, y medio limón
...
Doscientos veintitrés mil cuatrocientos diecisiete limones, y medio limón
iiiY DOSCIENTOS VEINTITRÉS MIL CUATROCIENTOS DIECIOCHO LIMONES Y MEDIO LIMÓN!!!

3. EJERCICIO 3

Crea un programa que calcule el cambio a devolver. El programa tendrá como entrada una lista de monedas/billetes disponibles y la cantidad a devolver. Como salida, devolverá el cambio a utilizar (puedes hacer que lo devuelva como texto o como cualquier estructura de datos que consideres)

Por ejemplo:

Entrada

```
-----  
Monedas disponibles: [ 0.10, 0.50, 1, 2, 5]  
Cambio a devolver: 9.50  
Resultado: 0.50x1, 2x2, 1x5
```

Debe devolver el cambio con la menor cantidad posible de monedas y billetes. En el ejemplo anterior, la solución $0.10 \times 5, 4 \times 1, 1 \times 5$ sería incorrecta ya que se usan 11 monedas, al contrario que en la solución del ejemplo en la que se utilizan 4.



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 01. **FUNDAMENTOS DEL LENGUAJE**

Desarrollo Web en entorno cliente
CFGs DAW

Ejercicios de Javascript

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2019/2020

Versión:210925.0809

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 1: ¡Hola Mundo!.....	3
2. Ejercicio 2: Buenos días.....	3
3. Ejercicio 3: Día de la semana.....	3
4. Ejercicio 4: FizzBuzz.....	4
5. Ejercicio 5: La Moda.....	4
6. Ejercicio 6: Alumnos de primero y segundo.....	4

UD01. FUNDAMENTOS DEL LENGUAJE

Estos ejercicios no hacen uso del navegador (excepto en el primero) por lo que lo más sencillo es que prepares un fichero con el código y lo ejecutes con node: de este modo no necesitarás una página web ni un navegador para probarlos.

Es además muy sencillo utilizar el depurador de Visual Studio code para introducir puntos de ruptura y examinar el estado de las variables: normalmente pulsando F5 desde la ventana de código puedes iniciar la depuración con node.

1. EJERCICIO 1: ¡HOLA MUNDO!

El objetivo de este ejercicio es asegurarte de que puedes ejecutar JS correctamente. Deberás probar tres cosas:

1. Ejecuta en la consola del navegador una orden que imprima "¡Hola Mundo!" en la misma consola
2. Abre en el navegador un fichero HTML que, al cargarse, imprima en la consola "¡Hola, Mundo!"
3. Prepara un fichero JavaScript que, al ser ejecutado con node, imprima "¡Hola, Mundo!" por la salida estándar

No basta con que准备es el código: asegúrate de poder ejecutarlo en tu sistema (abre la página web desde un navegador, ejecuta el fichero con node...)

2. EJERCICIO 2: BUENOS DÍAS

Construye un programa de Inteligencia Artificial que salude correctamente según la hora del día.

De 7 a 12 dirá “Buenos días”, de 12 a 20 “Buenas tardes”, de 20 a 2 “Buenas noches” y de 2 a 7 de la mañana “¿Qué haces despierto a estas horas?”

Por ejemplo, en el siguiente caso:

```
const hora = 15
```

El programa diría “Buenas tardes”

3. EJERCICIO 3: DÍA DE LA SEMANA

El 1 de enero de 2021 fue viernes.

Prepara un programa que, dado un día y un mes (de 2021) calcule qué día de la semana es.

Por ejemplo, si el programa tiene de entrada:

```
const dia = 10  
const mes = 1
```

debería imprimir “Domingo”.

No puedes utilizar ninguna función predefinida de manejo de fechas de JavaScript (puedes hacerlo

calculando cuántas semanas y días han pasado desde el 1 de enero, por ejemplo)

4. EJERCICIO 4: FizzBuzz

Escribir un programa que muestre en pantalla los números del 1 al 300 sustituyendo los números que terminen en 3 por la palabra “fizz”, los números que acaben en 5 por “buzz” y los números que acaben en 15 por la palabra “fizzbuzz”.

Un ejemplo de salida sería:

`1, 2, fizz, 4, buzz, ..., 14, fizzbuzz, 16, ...`

Puedes separar los números por coma o escribir uno en cada línea.

5. EJERCICIO 5: LA MODA

Prepara un programa que construya una matriz de veinte números aleatorios entre 0 y 10. Una vez construida dicha matriz, el programa debe calcular cual es la moda.

Para obtener un número aleatorio entre 0 y 10 puedes utilizar el siguiente código:

`Math.floor(Math.random() * 10);`

Puedes consultar cómo se calcula la moda en este enlace:<https://www.disfrutalasmaticas.com/datos/moda.html>

Un ejemplo de ejecución del programa sería:

`[1, 2, 4, 5, 4 ,4 ,4 ,4 ,4 ,0 ,7, 8, 4 ,9, 7, 3, 3, 1, 0]`

Moda: 4

6. EJERCICIO 6: ALUMNOS DE PRIMERO Y SEGUNDO

Tienes una estructura que contiene los alumnos matriculados en cada uno de los módulos de un ciclo de formación profesional. La estructura tiene el siguiente formato:

```
const modulos = [
  {
    nombre: 'Sistemas informáticos',
    curso: 1,
    alumnos: [
      'Don Pepito', 'Perico', 'Don José'
    ]
  },
  ...
  {
    nombre: 'Desarrollo Web en entorno cliente',
    curso: 2,
    asignatura: '',
    alumnos: [
      'Juan', 'Perico', 'Andrés', 'Don Pepito'
    ]
  },
  ...
]
```

Construye un programa que imprima los alumnos que están matriculados a la vez en asignaturas de primer y segundo curso.

Con los datos visibles en el ejemplo anterior debería imprimir: 'Perico' y 'Don Pepito'

Ten en cuenta que la estructura puede contener más asignaturas de primero y segundo, y no tienen por qué estar en orden.



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 01. **FUNDAMENTOS DEL LENGUAJE**

Desarrollo Web en entorno cliente
CFGs DAW

Ejercicios de Javascript

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2019/2020

Versión:211001.1034

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicios Rest y Spread.....	3
1.1 Elementos únicos.....	3
1.2 Combinación de Arrays.....	3
1.3 Suma de números.....	3
2. Diamante.....	3
3. LCD.....	4

UD01. FUNDAMENTOS DEL LENGUAJE

1. EJERCICIOS REST Y SPREAD

1.1 ELEMENTOS ÚNICOS

Escribe una función llamada `uniques` que acepte un número variable de argumentos y devuelva un array que contenga todos los argumentos pero eliminando los repetidos.

Por ejemplo:

```
uniques(2, 4, 'patata', [1,2], [1, 2], 'patata', 4) => [2, 4, 'patata', [1,2], [1,2]]
```

Ten en cuenta que [1,2] y [1,2] son dos arrays diferentes, y por tanto no se consideran duplicados

1.2 Combinación de Arrays

Escribe una función `combineArrays` que, usando el operador spread, reciba dos arrays como parámetros y devuelva un array con el contenido de ambos arrays: primero el segundo y después el primero.

Por ejemplo:

```
combineArrays([1,2], [3,4]) => [3,4,1,2]
```

1.3 Suma de números

Crea una función llamada `sumNumbers` que acepte un número arbitrario de argumentos (de cualquier tipo) y devuelva la suma de los argumentos numéricos.

Por ejemplo:

```
sumNumbers('hola', 2, 3, [10, 20, 30], { value: 300 }) => 5
```

2. DIAMANTE

Crea una función `diamante` que cree un diamante con todas las letras hasta la letra pasada como parámetro. Es suficiente con que funcione con letras mayúsculas. No hace falta que hagas validaciones sobre parámetros.

Por ejemplo:

```
diamante('C')
--A--
-B-B-
C---C
-B-B-
--A--
```

3. LCD

Crea un programa que, dado un número, imprima en pantalla una representación de ese número en un display LCD utilizando los caracteres — y |.

```
lcd(3):
```

```
—  
|  
—  
|  
—
```

```
lcd(45)
```

```
| | | —  
— | —  
| | —
```



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 02. MANEJO DEL NAVEGADOR

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:231006.1006

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Dónde está el ratón.....	3
2. Índice de una página web.....	3
3. Elementos autodestructibles.....	3
4. Cambio de color.....	4

UD02. MANEJO DEL NAVEGADOR

1. EJERCICIO 1: DÓNDE ESTÁ EL RATÓN

Realiza un programa que mediante eventos y el uso del objeto `event` te muestre en todo momento la posición actual del ratón en pantalla. Para mostrarlo modificaremos de forma dinámica un elemento HTML (Ejemplo un `<p>`) que nos muestre la posición actual del ratón.

2. EJERCICIO 2: ELEMENTOS AUTODESTRUCTIBLES

Realiza un programa que tenga 10 elementos `<p>` y al hacer clic sobre ellos desaparezcan (se oculten) y al hacer click con el botón derecho los elimine del DOM (no se debe mostrar el menú de contexto). También deberá tener un botón “Reaparecer” que hará que aparezcan todos los elementos desaparecidos (pero no los eliminados).

3. EJERCICIO 3: CAMBIO DE COLOR

Crea una página HTML con un DIV que contenga en su interior al menos tres DIV de diferentes colores. Al pulsar uno de los `<div>` el padre se podrá del color de ese elemento.

Debes usar la misma función para manejar los eventos en los tres `<div>` (no vale que hagas una función específica para cada uno)

4. EJERCICIO 4: ÍNDICE DE UNA PÁGINA WEB

Prepara un código JavaScript que, añadido a una página web, genere una tabla de contenido a partir de las etiquetas `<h1>`, `<h2>`...`<h6>` El índice lo generará dentro de una etiqueta `<pre>`, añadiendo espacios para que quede identado.

El resultado de este ejercicio debería ser una página html con el script, pero el script ha de ser genérico y debería poder funcionar en cualquier otra página.

Por ejemplo, dada la siguiente página:

```
<html>
<body>
    <h1>Tema 1</h1>
    <div>
        <h2>Apartado 1</h2>
    </div>
    <h2>Apartado 2</h2>
    <h1>Tema 2</h1>
    <h2>Introducción</h2>
</body>
</html>
```

Debería generar una etiqueta `<pre>` con el siguiente contenido:

```
<pre>
1.- Tema 1
    1.1.- Apartado 1
    1.2.- Apartado 1
2.- Tema 2
    2.1.- Introducción
</pre>
```

Pista:

Divide el ejercicio en dos: primero, la obtención de datos: intenta crear una estructura con los títulos que puedas manejar como te interese. Luego, a partir de esa estructura, genera el contenido del `<pre>`.



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 02. MANEJO DEL NAVEGADOR

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2021/2022

Versión:231010.0643

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

□ Importante

□ Atención

□ Interesante

ÍNDICE DE CONTENIDO

5. Validación de email.....	3
6. Confirmación de email.....	3
7. Selección de localidad.....	3

UD02. MANEJO DEL NAVEGADOR

5. VALIDACIÓN DE EMAIL

Realiza un formulario que pida una dirección de email y la valide antes de enviarla:

- Debe validar si el email sigue el formato `texto@servidor.loquesea`
- Además de validar el formato anterior, debe comprobar que servidor.loquesea este como servidor admitido en un array de servidores llamado “listaServidores”.

Dicho array debe ser definido a mano en el código. Por ejemplo

```
let listaServidores=[ "terra.es" , "myspace.com" , "arrakis.es" , "tuenti.es"];
```

6. CONFIRMACIÓN DE EMAIL

Realiza un formulario que tenga dos campos de email con las mismas validaciones que el ejercicio anterior. Deberá validar antes del envío si los dos emails son iguales.

7. SELECCIÓN DE LOCALIDAD

Realiza un formulario con dos elementos “select”. Al cambiar el primero, se actualizará el segundo. Al enviar el formulario, se comprobará que ambos han sido marcados.

Cuando no tengan ninguna selección previa, los “select” mostrarán “Select no seleccionado”.

Los valores del primer “select” serán “Alicante”, “Castellón”, “Valencia”. Por defecto no habrá ninguno seleccionado.

Los valores del segundo “select” son:

- Para Alicante : “Alicante Capital”, “Elche”, “Orihuela”.
- Para Castellón : “Castellón Capital”, “Oropesa”, “Vinaroz”.
- Para Valencia : “Valencia Capital”, “Torrent”, “Mislata”. (Aquí saldrá por defecto seleccionado “Mislata”).

Aquí tienes información adicional sobre cómo modificar el contenido de selects dinámicamente:
<http://www.javascriptkit.com/javatutors/selectcontent.shtml>

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- García Barea, Sergi



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 03. MÓDULOS Y HERRAMIENTAS DE DESARROLLO

Desarrollo Web en entorno cliente
CFGs DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2024/2024

Versión:241013.2011

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 1: Incrementar contador.....	3
2. Ejercicio 2: Puntuación de tenis.....	3
3. Ejercicio 3: Publicación de un módulo npm.....	3

UD03. MÓDULOS Y HERRAMIENTAS DE DESARROLLO

Para cada uno de los ejercicios de este módulo deberás crear un entorno de desarrollo con Vite. Los fuentes estarán en el directorio `src` excepto `index.html` que estará en el raíz.

1. EJERCICIO 1: INCREMENTAR CONTADOR

Crea un módulo que implemente una función `incrementar()` que devuelva un número empezando por 1 y que se vaya incrementando en cada llamada, y otra función `reset()` para reinicializarlo.

Crea una página web que tenga un único `<p>` con el contenido inicial 0, un botón “[Incrementar](#)” y otro botón “Reset”.

Crea otro módulo que implemente la funcionalidad necesaria para hacer que, al pulsar el botón, el párrafo se incremente o se resetee a cero.

2. EJERCICIO 2: PUNTUACIÓN DE TENIS

Construye un módulo para mantener la puntuación de un juego en un partido de tenis.

El módulo exportará tres funciones:

- `iniciarJuego()`: Inicializa la puntuación del juego.
- `puntoJugador(jugador)`: Indica que uno de los jugadores ha ganado el punto. Si el juego había finalizado, lanzará una excepción. El parámetro jugador será `1` para el primer jugador y `2` para el segundo. Si se pasa un número de jugador diferente lanzará una excepción con el mensaje ‘[Jugador inválido](#)’. Si el juego había finalizado se lanzará la excepción ‘[El juego ya ha finalizado](#)’.
- `resultado()`: Devuelve una cadena indicando el resultado actual del juego. Según el caso debe devolver (atención a las mayúsculas y espacios):
 - Deuce: [Deuce](#)
 - Ventaja de un jugador: [Ventaja jugador 1](#)
 - En otro caso: [Jugador 1: 15 Jugador 2: 40](#)

La puntuación de un jugador se incrementa cada vez que gana un punto siguiendo la siguiente secuencia: 0, 15, 30, 40. Si el jugador tiene una puntuación de 40 y gana el punto, gana el juego.

Si los dos jugadores tienen 40 puntos se pasa a “deuce”. Quien gane el punto tiene “ventaja”. Si el jugador que no tiene ventaja gana el punto, se pasa de nuevo a “deuce”. Si el jugador que tiene ventaja gana el punto, ganará el juego.

Crea dos versiones: una versión que utilice CommonJS y otra versión que utilice módulos ES6 (las dos versiones deben estar en el mismo proyecto)

3. EJERCICIO 3: PUBLICACIÓN DE UN MÓDULO NPM

Crea un módulo que exporte una función. La función debe admitir un array como entrada y transformar todos sus elementos en la cadena '[banana](#)'.

Sigue la guía que hay en los enlaces para publicar dicho módulo en el repositorio de npm. Utiliza scopes para publicarlo como [@\[TU USUARIO\]/bananize](#).

Una vez publicado, crea un proyecto nuevo y comprueba que puedes utilizarlo. Mira qué es lo que se ha descargado en [node_modules](#) cuando instalas el módulo en otro proyecto.

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 04. CLASES Y OBJETOS

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:221024.0656

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 1: La célula.....	3
2. Ejercicio 1: El gato.....	3
3. Ejercicio 1: Tres en ralla.....	4

UD04. CLASES Y OBJETOS

Los ejercicios 2 y 3 de esta semana los puedes hacer utilizando clases o utilizando prototipado. Para aprobar el módulo tendrás suficiente con saber trabajar con clases, pero si quieres aprender JavaScript más a fondo te recomiendo que hagas el ejercicio utilizando los dos métodos.

✓ Recuerda crear los tests para los ejercicios

1. EJERCICIO 1: LA CÉLULA

Crea un objeto célula mediante literales de objetos. El objeto debe contener valores primitivos (números, booleanos y cadenas, por ejemplo) y una función `mitosis()` que le permita dividirse en dos, copiando sus propiedades a los hijos y eliminando las propiedades propias.

El objeto debe funcionar aunque se le añadan nuevas propiedades de tipo primitivo (no hace falta que hagas ninguna comprobación sobre esto)

No crees constructores, clases... la célula ha de ser un único objeto JavaScript y estar autocontenido.

☞ Los hijos también deben ser capaces de realizar la mitosis para que pueda continuar la línea celular

2. EJERCICIO 2: EL GATO

Construye un programa que te permita crear gatos virtuales.

Los gatos virtuales deben tener tres propiedades: hambre, cansancio y felicidad. Crea métodos para alimentar al gato, ponerlo a dormir y jugar con él. Debe haber un método que imprima el estado del gato. Crea también otro método para indicarle el paso del tiempo (debe admitir un parámetro que serán los milisegundos de tiempo que han pasado)

El hambre y el cansancio se incrementan con el tiempo y se incrementan al comer y dormir. La felicidad se decrementa con el tiempo y aumenta al jugar. En las funciones debe haber cierta aleatoriedad, la puedes generar con `Math.random()`

Ten en cuenta que el gato no debe imprimir nada por consola: puedes preparar código que "use" el gato y muestre sus propiedades, pero eso no formará parte del gato.

Recuerda utilizar módulos.

3. EJERCICIO 3: TRES EN RALLA

Implementa un módulo para jugar al tres en ralla mediante clases y/u objetos. Debe tener una función para jugar donde se le indique la posición a jugar. El juego debe recordar el turno, de modo que a la hora de jugar sólo se indicará dónde, no quién juega.

Cuando se introduzca una jugada inválida, el juego lanzará una excepción.

Debe haber algún método para comprobar si el juego ha finalizado y quién es el ganador, así como para imprimir el estado del juego.

El módulo del tres en ralla no puede imprimir nada por consola.

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 04. CLASES Y OBJETOS

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2021/2022

Versión:221101.0824

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 4: Polígonos.....	3
2. Ejercicio 5: Procesador de textos.....	3
3. Ejercicio 6: Roombas.....	4

UD04. CLASES Y OBJETOS

Realiza los ejercicios de esta semana intentando usar clases o prototipado.



Recuerda crear los tests para los ejercicios

1. EJERCICIO 4: POLÍGONOS

Implementar una clase [Polígono](#) que se componga de [n](#) puntos. Implementa las clase [Cuadrado](#), [Rectángulo](#) y [Triángulo](#) que tengan una función [perímetro\(\)](#) que calcule el perímetro de los mismos.

El perímetro de un polígono se calcula sumando la longitud de sus lados.



No todos los conjuntos de 3 puntos forman un triángulo. No todos los conjuntos de 4 puntos forman cuadrados o rectángulos.

2. EJERCICIO 5: PROCESADOR DE TEXTOS

Crea un mini-procesador de textos. El procesador de textos debe admitir párrafos que pueden estar justificados al centro, a la izquierda o a la derecha.

Dado un ancho en caracteres debe generar un array de cadenas donde cada elemento será una cadena que tendrá como longitud el ancho en caracteres dado, completando con espacios para justificar.

Por ejemplo, dados estos párrafos:

- Párrafo 1: "Tres tigres blancos", justificado izquierda
- Párrafo 2: "Comían", justificado centro
- Párrafo 2: "El trigal", justificado derecha

La salida con ancho 15 sería (la primera línea es para contar los caracteres):

```
123456789012345
Tres.tigres.....
blancos.....
.....Comían ...
.....El trigal
```

Si hay una palabra más larga que el ancho dado puedes partirla por donde quieras. En la tercera línea el espacio de más tiene que estar al principio.

3. EJERCICIO 6: ROOMBAS

En una estación espacial en forma de anillo han comprado unos cuantos aspiradores automáticos para ayudarles mantener la limpieza.

La estación está dividida en $N \times M$ casillas. Los aspiradores se mueven en línea recta por turnos, avanzando una casilla cada turno. Ten en cuenta que la estación es cilíndrica, así que cuando los aspiradores se “salen” por la derecha vuelven a “entrar” por la izquierda y viceversa; lo mismo sucede arriba y abajo.

En un momento dado puede coincidir más de un aspirador en una casilla. Se considera entonces que han chocado. Cuando sucede esto, los aspiradores reaccionarán de diferentes maneras:

- Si el aspirador es ruso, al chocar con un obstáculo girará a la izquierda
- Si el aspirador es estadounidense, girará a la derecha
- Si el aspirador es europeo esperará dos turnos a recibir instrucciones de la central y luego cambiará de dirección aleatoriamente
- China no ha participado —tiene su propia estación espacial— pero ha hecho todos los aspiradores.

Implementa un simulador de limpieza de la estación que indice en cada momento dónde están los robots.

Bonus: registra qué partes de la estación se han limpiado ya y detén los robots cuando toda la estación esté limpia.

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 05. PROGRAMACIÓN FUNCIONAL

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2023/2024

Versión:231106.1140

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 1: Purificación.....	3
2. Ejercicio 2: Idés.....	3
3. Ejercicio 3: Multiplicación perezosa.....	4
4. Ejercicio 4: Pipas.....	4

UD05. PROGRAMACIÓN FUNCIONAL

Esta semana los ejercicios son más cortos, pero puede que conceptualmente te resulten un poco más complejos.

1. EJERCICIO 1: PURIFICACIÓN

Modifica la función `mergeValues` para que sea una función pura.

```
const MINIMUM = 15;

function mergeValues(arrayOfIntegers) {
  let element;
  let sum = 0;

  while(element = arrayOfIntegers.pop()) {
    sum += element
  }

  sum = Math.max(sum, MINIMUM)
  arrayOfIntegers.push(sum);

  return arrayOfIntegers;
}

console.log(mergeValues([10,20,30,40])) // [100]
console.log(mergeValues([1,2,3,4])) // [15] (MINIMUM)
```

2. EJERCICIO 2: IDÉS

Crea una función que permita crear funciones para generar IDs. Las funciones devueltas generarán una cadena de la longitud definida cuando se invoquen. La cadena se irá incrementando con cada invocación.

Ejemplos:

```
const len3Id = createIDGenerator(3);
len3Id() // 001
len3Id() // 002
len3Id() // 003

const len5Id = createIDGenerator(5);
len5Id() // 00001
```

 Date cuenta de que las funciones que crea `createIDGenerator` no son funciones puras, ya que devuelven diferentes valores con los mismos parámetros de entrada

3. EJERCICIO 3: MULTIPLICACIÓN PEREZOSA

Crea una función llamada `lazyMultiply` para multiplicar dos números. Si a la función se le pasan dos parámetros devolverá inmediatamente la solución. Si a la función se le pasa un único parámetro devolverá una función que, al pasarle un segundo parámetro, devolverá el resultado de la multiplicación.

Ejemplos:

```
lazyMultiply(7,4) // 28  
  
const perTwo = lazyMultiply(2)  
perTwo(3) // 6  
  
lazyMultiply(5)(10) // 50
```

4. EJERCICIO 4: PIPAS

Crea una función llamada `doublePipe` que admita un número variable de funciones y devuelva una función nueva que aplique esas funciones en el mismo orden en que se han pasado, pero aplicando dos veces cada una de las funciones.

Por ejemplo:

```
function double(x) { return x*2 }  
function add3(x) { return x+3 }  
  
let multiplyPerFourAndAddSix = doublePipe(double, add3)  
console.log(multiplyPerFourAndAddSix(10)) // 46 = (10*2*2+3+3)  
  
let addSixAndMultiplyPerFour = doublePipe(add3, double)  
console.log(addSixAndMultiplyPerFour(10)) // 64 = (10+3+3)*2*2
```

Las funciones deben admitir un sólo parámetro.

👉 Recuerda que puedes utilizar el operador spread para manejar un número variable de parámetros en una función

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 05. PROGRAMACIÓN FUNCIONAL

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:221114.1316

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 5: Validación de direcciones.....	3
2. Ejercicio 6: Remap.....	4
3. Ejercicio 7: Evolución de Pokémon.....	4

UD05. PROGRAMACIÓN FUNCIONAL

Los ejercicios están pensados para que intentes aplicar las técnicas de programación funcional vistos en la unidad didáctica.



Recuerda crear los tests para los ejercicios

1. EJERCICIO 5: VALIDACIÓN DE DIRECCIONES

Escribe un programa que filtre las direcciones de una lista para obtener únicamente las direcciones válidas. Para que una dirección sea válida debe cumplir las siguientes condiciones:

- Debe tener informados el país, la ciudad y la dirección
- Debe tener informado o el móvil o el fijo
- Debe incluir una región o un código postal (cp)

El filtrado se debe hacer en una única sentencia que ocupe una sola línea de código. Además, **no puedes utilizar if en ninguna parte del programa**.

Por ejemplo, con la siguiente lista de direcciones debería devolver únicamente la primera:

```
let direcciones = [
  {
    // Válido
    pais: 'España', region: '', cp: '46014',
    ciudad: 'Valencia', direccion: 'Carrer Misericòrdia, 34',
    complemento: '',
    movil: '', fijo: '961 20 69 90'
  },
  {
    // Inválido: no tiene móvil o fijo
    pais: 'España', region: '', cp: '46960',
    ciudad: 'Aldeia', direccion: 'C/ Montcabrer, 22',
    complemento: 'Pol. Ind. La Lloma',
    movil: '', fijo: ''
  },
  {
    // Inválido: no tiene país
    pais: '', region: 'Alicante', cp: '',
    ciudad: 'Petrer', direccion: 'Los Pinos, 7',
    complemento: '',
    movil: '', fijo: '965 37 08 88'
  }
]
```

2. EJERCICIO 6: REMAP

Implementa una función `map(array, función)` que reciba un array y aplique la función a cada elemento, devolviendo un nuevo array.

No puedes usar bucles de ningún tipo ni las funciones `forEach` o `map` de los arrays

Por ejemplo:

```
map([1,2,3], x => x * 2) // [2,4,6]
```

3. EJERCICIO 7: EVOLUCIÓN DE POKÉMON

Dada la cadena de evolución de un pokémon en formato JSON obtenida del PokéAPI (<https://pokeapi.co/>) obtén una cadena con la lista de los diferentes pokémon en los que evoluciona. Si un pokémon puede tener varias cadenas de evolución (por ejemplo, [Eevee](#)) devuelve sólo el contenido de la primera.

Por ejemplo, para Bulbasaur, se puede obtener la cadena de evolución de esta dirección: <https://pokeapi.co/api/v2/evolution-chain/1>. El programa debería devolver la cadena “Bulbasaur - Ivysaur - Venusaur” (ojo, que la primera letra está en mayúsculas)

Si quieras ver otras cadenas de evolución puedes cambiar el número final en la URL. Aquí tienes la documentación sobre la información de devuelve el endpoint:

<https://pokeapi.co/docs/v2#evolution-section>

☞ No es necesario que hagas la llamada para obtener los datos, sólo debes programar el proceso de la cadena. Ten en cuenta que para las pruebas no necesitarás la cadena entera sino sólo las partes relevantes para el ejercicio.

☞ Aunque este ejercicio te parezca complicado, se puede implementar con menos de 10 líneas de código. Usa la recursividad.



Bonus: devuelve un array con todas las posibles cadenas de evolución

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 06.

PROGRAMACIÓN ASÍNCRONA

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2021/2022

Versión:221127.1848

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 1: Callbacks secuenciales.....	3
2. Ejercicio 2: Callbacks paralelos.....	3
3. Ejercicio 3: cutriFetch.....	3
4. Ejercicio 4: Fizz asíncrono.....	4
5. Código de asyncRequest.....	4

UD06. PROGRAMACIÓN ASÍNCRONA



El código de `asyncRequest()` está al final del enunciado de los ejercicios

1. EJERCICIO 1: CALLBACKS SECUENCIALES

Utilizando la función `asyncRequest()` realiza de forma secuencial peticiones para obtener `recurso1`, `recurso2` y `recurso3`, en ese orden. Cuando hayas obtenido los tres recursos debes imprimir “¡Completado!” en la consola.

2. EJERCICIO 2: CALLBACKS PARALELOS

Utilizando la función `asyncRequest()` realiza de forma simultánea peticiones para obtener `recurso1`, `recurso2` y `recurso3`.

Debes imprimir el contenido de cada recurso en orden y lo antes posible. Cuando se hayan obtenido los tres recursos debes imprimir “¡Completado!” en la consola.

Esto es un ejemplo de cómo se podrían recibir los datos y qué tendría que imprimirse por pantalla:

Datos recibidos	Impreso en consola
resource2	(No se imprime nada ya que aún no tenemos el resultado de resource1)
resource1	The first resource The second resource
resource3	The third resource ¡Completado!

3. EJERCICIO 3: CUTRIFETCH

Crea un módulo con una función llamada `cutriFetch()` que reciba como parámetro un nombre de recurso, llame a `asyncRequest()` y devuelva una promesa que se resuelva cuando se han obtenido los datos del recurso.

Realiza el ejercicio 1 utilizando esa nueva función.

4. EJERCICIO 4: Fizz ASÍNCRONO

Crea una función llamada `fizz()` que devuelva si un número es divisible por 3 o contiene un 3. Sin embargo, debe devolver ese resultado al cabo de un tiempo aleatorio entre 100 y 10.000 ms (puedes reducir este número mientras estás probando el ejercicio)

Imprime los números del 1 al 300. Si `fizz()` devuelve `false` debes imprimir el número. Si `fizz()` devuelve `true`, debes imprimir "fizz" en lugar del número.

Debes imprimir los números en orden pero tan rápido como puedas.

5. CÓDIGO DE ASYNCREQUEST

```
const DEFAULT_RESOURCES = {  
    "resource1": "The first resource",  
    "resource2": "The second resource",  
    "resource3": "The third resource",  
    "whatINeed": "resource2"  
}  
  
function asyncRequest(resource, callback, resources = DEFAULT_RESOURCES) {  
    const MIN_DELAY = 1_000;  
    const RANDOM_DELAY = 2_000;  
  
    var randomDelay = Math.round(Math.random() * RANDOM_DELAY) + MIN_DELAY;  
  
    console.log("**Requesting: " + resource + "***");  
  
    setTimeout(function(){  
        console.log("**Returning: " + resources[resource] + "***");  
        callback(resources[resource]);  
    }, randomDelay);  
}  
  
module.exports = asyncRequest;
```

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 06. PROGRAMACIÓN ASÍNCRONA

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:221207.0939

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 5: Movimientos de pokémon.....	3
2. Ejercicio 6: Gravity.....	3

UD06. PROGRAMACIÓN ASÍNCRONA

1. EJERCICIO 5: MOVIMIENTOS DE POKÉMON

Crea una función `movimientos()` que, dado un nombre de pokémon en inglés, devuelva de forma asíncrona el nombre en castellano de todos los movimientos de dicho pokémon en orden alfabético.

Por ejemplo, `movimientos('bulbasaur')` debería devolver:

['Abatidoras', 'Bomba Germen', ...]

Los datos de un pokémon pueden obtenerse con una llamada al API:

[https://pokeapi.co/api/v2/pokemon/\[NOMBRE DEL POKEMON EN INGLÉS\]](https://pokeapi.co/api/v2/pokemon/[NOMBRE DEL POKEMON EN INGLÉS])

💡 Puedes utilizar una herramienta como este formateador de JSON para ver mejor las respuestas a las peticiones: <https://jsonformatter.curiousconcept.com/>

En esta herramienta basta con que pongas la URL de la petición y te devolverá el JSON que responda la petición formateado.

En los datos del pokémon hay enlaces para obtener datos de los diferentes apartados, entre ellos los movimientos. Debe obtener los datos de los movimientos lo más rápidamente posible. Recuerda gestionar los posibles errores.

2. EJERCICIO 6: GRAVITY

Utilizando el API de localización de la estación espacial internacional crea una página web que muestre su posición sobre el mapa cada diez segundos:

<http://open-notify.org/Open-Notify-API/ISS-Location-Now/>

Para los mapas puedes utilizar `leafletjs`. En la página inicial tienes un ejemplo de cómo mostrar un mapa utilizando los datos de OpenstreetMap: <https://leafletjs.com/examples/quick-start/>

Extra: Añade un botón para centrar el mapa en la posición actual de la ISS.

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 06. PROGRAMACIÓN ASÍNCRONA

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:231214.1858

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio A1: Información de usuario simple.....	3
2. Ejercicio A2: Información de usuario completa.....	4
3. Ejercicio A2: Información de múltiples usuarios.....	5

UD06. PROGRAMACIÓN ASÍNCRONA

1. EJERCICIO A1: INFORMACIÓN DE USUARIO SIMPLE

En este ejercicio, tu tarea es simular la obtención de información de un usuario mediante una función asíncrona que devuelve una promesa. No utilices `fetch` en este caso, simplemente crea una función `obtenerInformacionUsuario` que simula una solicitud asíncrona.

- Crea una función llamada `obtenerInformacionUsuario` que toma un parámetro `idUsuario` (número entero) y devuelve una promesa.
- Dentro de la función, simula una demora asíncrona aleatoria entre 0.5 y 3 segundos antes de resolver o rechazar la promesa. Puedes usar `setTimeout` para simular esta demora.
- Si `idUsuario` es un número par, resuelve la promesa con un objeto que contenga la información del usuario, por ejemplo:

```
{ id: idUsuario, nombre: 'Usuario Par', tipo: 'Regular' }
```

- Si `idUsuario` es un número impar, rechaza la promesa con un mensaje de error, por ejemplo:

```
new Error('Error: Usuario Impar no permitido')
```

5. Implementa un ejemplo de uso de esta función. Llama a `obtenerInformacionUsuario` con un número par e imprime la información del usuario si la promesa se resuelve correctamente. Luego, llama a la función con un número impar e imprime el mensaje de error si la promesa es rechazada.

2. EJERCICIO A2: INFORMACIÓN DE USUARIO COMPLETA

En este ejercicio, practicaremos el uso de promesas para realizar llamadas paralelas a múltiples endpoints de una API ficticia. Supongamos que tenemos tres endpoints que proporcionan información sobre usuarios, posts y comentarios respectivamente. Tu tarea es obtener datos de estos tres endpoints en paralelo.

Crea dos funciones llamadas `obtenerInfoUsuarios` y `obtenerInfoPosts`. Cada una de ellas toma un parámetro `id` y devuelve una promesa.

Cada función simulará una llamada asíncrona a un endpoint diferente de una API ficticia. Usa `setTimeout` para simular una demora en cada llamada.

- `obtenerInfoUsuarios(id)`: Simula obtener información sobre un usuario con el `id` proporcionado. Los usuarios tendrán este esquema:

```
{  
  id: 1, nombre: 'Usuario 1'  
}
```

- `obtenerInfoPosts(id)`: Simula obtener información sobre los posts del usuario con el `id` proporcionado. Cada post tendrá los campos `titulo` y `contenido`:

```
[  
  {id: 1, titulo: 'Post 1', contenido: 'Lorem ipsum dolor sit amet'},  
  {id: 2, titulo: 'Post 2', contenido: 'consectetur adipisicing elit.'},  
  ...  
]
```

Cada una de estas funciones fallará aleatoriamente con una probabilidad del 5%.

Una vez tengas esas funciones crea una función llamada `obtenerInformacionCompleta` que toma un `idUsuario` y utiliza las funciones anteriores para obtener información de usuario, posts y comentarios en paralelo. La función debe devolver una promesa que se resuelva con un objeto que contenga la información completa. Por ejemplo:

```
{  
  usuario: { id: idUsuario, nombre: 'Usuario Ejemplo' },  
  posts: [...], // Array de posts  
}
```

Implementa un ejemplo de uso de `obtenerInformacionCompleta`. Llama a esta función con un `idUsuario` y maneja la promesa resultante utilizando `.then` para imprimir la información completa si la promesa se resuelve correctamente, y utiliza `.catch` para manejar cualquier error.

3. EJERCICIO A2: INFORMACIÓN DE MÚLTIPLES USUARIOS

Utiliza la función `obtenerInformacionCompleta` del apartado anterior para obtener los datos de cinco usuarios al mismo tiempo. Cuando hayas obtenido la información de todos los usuarios muéstralos con sus datos completos por orden alfabético.



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 07. APIS DEL NAVEGADOR

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2023/2024

Versión:240215.1600

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 1: Formulario.....	3
2. Ejercicio 2: Lista de tareas.....	3

UD07. APIs DEL NAVEGADOR

1. EJERCICIO 1: FORMULARIO

Crea un formulario con un PIN donde se puedan introducir hasta cinco caracteres. Al escribir el quinto carácter debe enviar el PIN a <https://validate-pin.fly.dev/validate> en formato `form-urlencoded` con el método POST y mostrar si el PIN es o no correcto.

👉 En el Mundo Exterior® usarías una función debounce para no lanzar demasiadas peticiones al servidor y tomaríais el valor de la última promesa resuelta. En esta URL tenéis algunos ejemplos de cómo implementarlo en JavaScript:

<https://webdesign.tutsplus.com/es/tutorials/javascript-debounce-and-throttle--cms-36783>

👉 Puedes acceder al código fuente del servidor en <https://github.com/CEED-2023/validate-pin> y lanzarlo en local si te es más cómodo

2. EJERCICIO 2: LISTA DE TAREAS

Crear una lista de tareas en el navegador. Debe tener un campo para escribir el texto de la tarea y un botón añadir. Al pulsar el botón se añadirá la tarea a una lista.

No se pueden eliminar ni modificar las tareas, sólo añadir. Si se cierra el navegador y se vuelve a abrir debe conservar la lista de tareas al acceder a la misma página.

✓ Recuerda crear un entorno de desarrollo con Vite

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 07. APIS DEL NAVEGADOR

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:230109.1224

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:



Importante



Atención



Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 3: Chat.....	3
2. Ejercicio 4: Efectos Especiales.....	3

UD07. APIs DEL NAVEGADOR

1. EJERCICIO 3: CHAT

Mejora el ejemplo del chat para permitir introducir un nombre de usuario al conectar. Los mensajes deben de indicar el usuario que los ha realizado.

Tendrás que modificar tanto la parte cliente como la parte servidor. Piensa de qué manera puede hacerle llegar cada cliente su nombre al servidor y cómo guardarlos en el mismo. No hace falta que compruebes que no haya nombre duplicados, pero sí que no se puedan enviar mensajes hasta que el servidor haya dado el OK.

Aunque no hemos visto nada de express ni de Websockets en el servidor lo único que tienes que modificar es la función que se ejecuta cuando el cliente recibe una conexión nueva: es código JavaScript y puedes utilizar todo lo que hemos visto del lenguaje hasta el momento.

2. EJERCICIO 4: EFECTOS ESPECIALES

Crea un proyecto con tres páginas:

1. Un índice `index.html` que tenga dos enlaces: uno a la página `nieve.html` y otro a `confetti.html`
2. Una página `nieve.html` en la que utilices el paquete npm `pure-snow` (<https://www.npmjs.com/package/pure-snow.js>) para mostrar nieve en la página. Debe tener un botón en el centro de la página para volver al índice.
3. Una página `confetti.html` en la que utilices el paquete npm `js-confetti` (<https://www.npmjs.com/package/js-confetti>) para mostrar confeti cada vez que se pulse un botón situado en el centro de la página.

No se podrá lanzar más confeti hasta que el confeti haya desaparecido.

Se debe poder volver a la página anterior (que será el índice si has accedido desde allí) pulsando el botón atrás del navegador.

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a

este documento:

- *Álvaro Maceda Arranz*



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 08. REACT

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:230115.1747

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 1: Gradient.....	3
2. Ejercicio 2: Picker.....	4
3. Ejercicio 3: SuperPicker.....	4

UD08. REACT

1. EJERCICIO 1: GRADIENT

Crea un componente **React** llamado **Gradient** que admita tres propiedades:

- **hue**: Valor entre 0 y 360 que indicará el matiz (hue) en un modelo de color HSL
- **saturation**: Valor entre 0 y 100 que indicará la saturación en un modelo de color HSL
- **number**: Indicará el número de cuadrados a crear en el componente

El componente creará tantos **divs** como los indicados en **number**. Los **divs** tendrán el color **hsl(hue, saturation, ligthness)** donde **hue** y **saturation** serán las pasadas como parámetro y **ligthness** se incrementará desde 0 a intervalos de **100/number**.

Por ejemplo, si se muestra el componente con:

```
<Gradient hue={23} saturation={100} number={5}>
```

Se mostraría esto en pantalla:



Los colores de los **divs** serían:

- **hsl(23, 100%, 0%)**
- **hsl(23, 100%, 20%)**
- **hsl(23, 100%, 40%)**
- **hsl(23, 100%, 60%)**
- **hsl(23, 100%, 80%)**

Puedes utilizar este CSS para los elementos HTML:

```
.squares {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  align-items: center;
  height: 100%;
}

.square {
  height: 100px;
  flex-grow: 1;
}
```

 En los ejercicios puedes crear tantos componentes React como creas necesario para realizar la tarea; normalmente necesitarás utilizar más de uno.

2. EJERCICIO 2: PICKER

Crea un componente **React** llamado **Picker** que tenga este aspecto:



Hue: 0

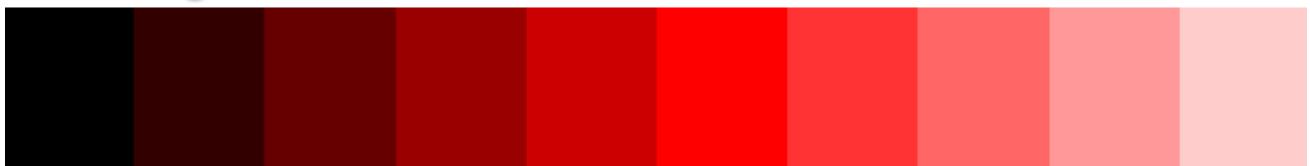
Saturation: 100

Steps: 10

El componente tendrá tres sliders creados con `<input type="range">`. El primero tendrá valores entre 0 y 360, el segundo entre 0 y 10 y el tercero entre 5 y 100. Al cambiar los valores de los sliders se mostrará el valor de los mismos en el recuadro inferior.

3. EJERCICIO 3: SUPERPICKER

Modifica el componente del ejercicio 2 para que, en vez de los valores, se muestre un degradado utilizando el componente del ejercicio 1:



A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 08. REACT

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2022/2023

Versión:230129.1341

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 4: PokeData.....	3
2. Ejercicio 5: PokeForm.....	3
3. Ejercicio 6: PokeDex.....	4

UD08. REACT

⚠️ Para estos ejercicios puedes utilizar la plantilla disponible en la sección. Deberás modificar las secciones donde se indica y también en otros lugares, según necesites.

1. EJERCICIO 4: POKE DATA

Crea un componente **React** llamado **PokeData**. El componente admitirá un id de especie de pokémon como propiedad (por ejemplo, 1, 4 o 7) y hará lo siguiente:

- Mostrará un spinner utilizando el componente **Loading** (que tendrás que completar)
- Obtendrá la URL del sprite para esa especie utilizando la función **getSpeciesSprite**
- Cuando se haya obtenido la URL se mostrará la imagen en vez del spinner

Si cambia el id de especie se debe recargar la imagen, mostrando el spinner hasta que esté cargada la nueva URL de la imagen.

⚠️ Puedes obtener más información sobre el API en <https://pokeapi.co/>, pero no lo necesitarás para estos ejercicios.

2. EJERCICIO 5: POKE FORM

Crea un componente **React** llamado **PokeForm** que tenga este aspecto:

PokeForm

Nombre del pokémon: Lenguaje

El funcionamiento será el siguiente:

- Al cargar **por primera vez** el componente se obtendrán los datos de todos los pokémon mediante una llamada a la función **getPokemonData**
- El **datalist** contendrá el nombre de todos los pokémon en el idioma seleccionado en la lista.
- Al pulsar “search”, se invocará a una función pasada en las propiedades con el id de especie correspondiente al pokémon indicado en el campo del formulario.
- Si el texto introducido en el campo no se corresponde a ningún nombre de pokémon en ese idioma, no se llamará a dicha función.

3. EJERCICIO 6: POKEDEX

Combina **PokeForm** y **PokeData** para montar una **PokeDex**:

Pokedex

PokeForm

Nombre del pokémon: Lenguaje

PokeData



¡Es muy efectivo!

A continuación ofrecemos en orden alfabético el listado de autores que han hecho aportaciones a este documento:

- Álvaro Maceda Arranz



CENTRE ESPECÍFIC
D'EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 08. REACT

Desarrollo Web en entorno cliente
CFGS DAW

Ejercicios

Álvaro Maceda Arranz

alvaro.maceda@ceedcv.es

2024/2025

Versión:250207.1029

Licencia



Reconocimiento - NoComercial - CompartirIgual (by-nc-sa): No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Importante

Atención

Interesante

ÍNDICE DE CONTENIDO

1. Ejercicio 1: Form.....	3
2. Ejercicio 2: Filas.....	4
3. Ejercicio 3: Cuadrados.....	4
4. Ejercicio 4: Redux.....	4
5. CSS para la aplicación.....	5

UD08. REACT

Para estos ejercicios, utiliza el CSS que hay al final del documento. Respeta las clases del HTML de ejemplo para que se visualice bien.

1. EJERCICIO 1: FORM

Crea un componente React llamado **Form**. Con este aspecto:



El componente tendrá una constante con una lista de colores con nombre y valor, que serán los que se muestren en el select. Además, deberá usar constantes para el valor mínimo y máximo tanto de **size** como de **number**:

Los datos del componente son para crear ese número de divs con el tamaño seleccionado. El espacio entre divs será de 5px. Debe controlar que no se supere un tamaño total, incluyendo el espacio, de 888 pixels. No se debe poder seleccionar una combinación que supere dicho tamaño. Por ejemplo, no podrán estar **size** a 90 y **number** a 10.

El HTML generado por el componente ha de ser similar a este:

```
<div class="form">
  <label> Size: 78 <input min="20" max="150" type="range" value="78">
  </label>
  <label>Number: 6 <input min="1" max="20" type="range" value="6"></label>
  <label>Color:
    <select>
      <option value="red">Red</option>
      <option value="blue">Blue</option>
      <option value="green">Green</option>
      <option value="yellow">Yellow</option>
      <option value="purple">Purple</option>
    </select>
  </label>
  <button>Añadir Fila</button>
</div>
```

2. EJERCICIO 2: FILAS

Crea un componente Fila que reciba tres datos: tamaño, número y color. Debe crear el número indicado de divs, del tamaño indicado, y con el color de fondo indicado.

Crea un componente Filas que utilice el componente anterior. Debe recibir un array que contenga los datos de cada fila y crear las filas correspondientes.

El HTML generado por los componentes ha de ser similar a este:

```
<div class="filas">
  <div class="fila">
    <div class="cuadro"
      style="width: 121px; height: 121px;
              background-color: red; margin-right: 5px;">
    </div>
    <div class="cuadro"
      style="width: 121px; height: 121px;
              background-color: red; margin-right: 5px;">
    </div>
  </div>
  <div class="fila">
    <div class="cuadro"
      style="width: 150px; height: 150px;
              background-color: green; margin-right: 5px;">
    </div>
  </div>
</div>
```

3. EJERCICIO 3: CUADRADOS

Crea un componente cuadrados que utilice los dos componentes anteriores. Modifica la aplicación para que, al pulsar uno de los cuadrados, el color de fondo de la aplicación se ponga del color del cuadrado seleccionado.

El div de la aplicación debe tener la clase app y se le debe aplicar el estilo de fondo correspondiente:

```
<div id="root">
  <div class="app" style="background-color: white;">
    ...
  </div>
</div>
```

4. EJERCICIO 4: REDUX

Modifica la aplicación para que utilice Redux.

5.CSS PARA LA APLICACIÓN

```
:root {  
  font-family: Inter, system-ui, Avenir, Helvetica, Arial, sans-serif;  
  font-weight: 400;  
}  
  
body {  
  margin: 0;  
}  
  
button {  
  border-radius: 8px;  
  border: 1px solid transparent;  
  padding: 0.6em 1.2em;  
  font-size: 1em;  
  font-weight: 500;  
  background-color: #c2c2c2;  
  cursor: pointer;  
}  
  
button:hover {  
  background-color: rgb(54, 63, 71);  
  color: white;  
}  
  
app {  
  width: 100vw;  
  height: 100vh;  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
  justify-content: flex-start;  
  padding-top: 30px  
}  
  
frm {  
  display: flex;  
  gap: 10px;  
  margin-bottom: 20px;  
  justify-content: center;  
}  
  
form label {  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
}  
  
filas {  
  display: flex;  
  flex-direction: column;  
  align-items: center;  
}  
  
fila {  
  display: flex;  
  margin-bottom: 5px;  
}  
  
.cuadro {  
  cursor: pointer;  
}
```