ACÁMICA

TEMA DEL DÍA

Estados y fechas

Los estados en React sirven para agregar comportamiento y dinamismo a nuestros componentes.

Además, veremos cómo trabajar con fechas en Javascript.

Agenda

Daily

Destructuring

Estados

Fechas en informática

Tiempo UNIX

Fechas en Javascript

Daily

Daily

Sincronizando...

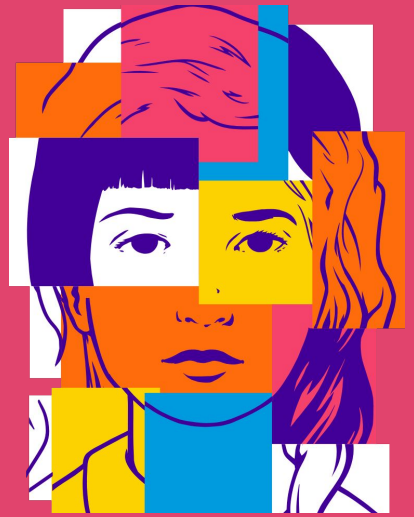
Toolbox

¿Cómo te ha ido?
¿Obstáculos?
¿Cómo seguimos?

Challenge

¿Cómo te ha ido?
¿Obstáculos?
¿Cómo seguimos?

Estados



Para entender el concepto de

ESTADO

pensemos como este se aplica en la vida real.

Imagina que eres un componente:

```
<Persona
  nombre="Luis"
  ojos="marrones"
  piel="morena"
  dni={123456}
/>
```

En React, **las props** se utilizan para definir valores externos de un componente.

Las props son valores que un componente no genera, o decide. Una persona, al nacer, no decide su nombre, numero de identificacion, color de ojos, o de piel.

Son valores que son dados por otras entidades. Esto mismo pasa en React con los componentes. Cuando son inicializados por primera vez, tendremos la oportunidad de decidir ciertas propiedades sobre ellos.

Las
PROPS
se asignan desde afuera

Sin embargo, el

ESTADO

es un valor interno a un componente. Generado y administrado por el mismo.

Por ejemplo, ¿a las personas se nos dice cuando tenemos que tener hambre? No, lo decidimos las mismas personas.

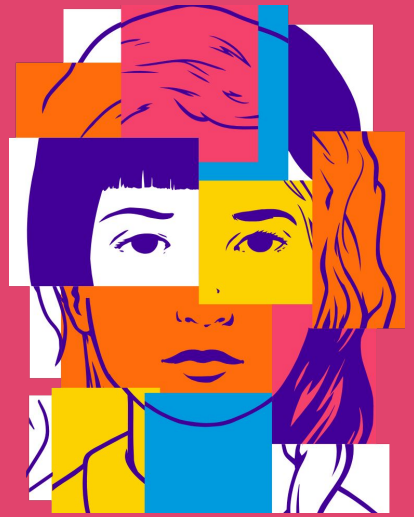
Un estado de ánimo, físico, mental, sentimental, son todos valores que los mismos componentes <Persona /> puede nadministrar.

Veamos cómo se implementan esto en React, pero antes hablemos de un tema: destructuring.

```
function Persona(props) {
  const [hambre, setHambre] = useState(false)

  return <div> ... </div>
};
```

Destructuring



Destructuring

Es la técnica que te permite extraer propiedades específicas de un objeto, array o función.

```
const { 🍇, 🍉, 🍊, 🍋, 🍍 } = 🥗
```

Extraemos y creamos variables con valores automáticos de un objeto, array o función.

Destructuring en objetos

Con el destructuring podemos declarar variables y asignar valores desde un objeto.

Por ejemplo:

```
const hamburguesa = {
  lechuga: true,
  tomate: true,
  pepinillo: false,
  carne: 1
}

const { lechuga, carne } = hamburguesa;
console.log(lechuga) // → true
console.log(carne) // → 1
```

Destructuring en arrays

Este concepto aplica también arrays, de la siguiente forma:

```
const nombres = ["javier", "gustavo", "lara"]

const [javi, gustavo] = nombres;
console.log(javi) // → "javier"
console.log(gustavo) // → "gustavo"

const [ , ,lara] = nombres;
console.log(lara) // → "lara"
```

Programo

Squad lead

Checkbox refactor

Vamos a refactorizar un ejercicio, aplicando el concepto de destructuring.

Punto de partida.

Programamos

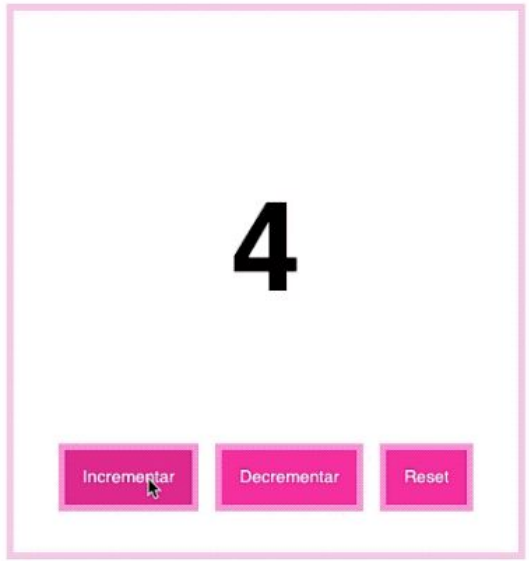
¡Todas las personas!

Contador

El estado puede ser cualquier tipo de valor válido en Javascript, por ejemplo, números, strings, booleanos, etc. Todo depende de qué clase de información queramos mostrar en el componente.

Vamos a construir una aplicación que nos permita manejar un estado numérico, en un contador.

Quien se anima a empezar desde este punto de partida ?



Paso 1: hook useState

Lo primero que debemos hacer para manejar el estado en un componente es importar el hook useState en los archivos que lo necesitemos.

En Contador.jsx modifica la primera línea a esto:

```
import React, {useState} from "react"
```

Luego, utiliza el hook para crear un valor inicial, y obtener la función actualizadora del valor. Sin destructuring hariamos algo asi:

```
const unValorInicial = 0;
const infoEstado = useState(unValorInicial);
const valorEstado = infoEstado[0];
const actualizarEstado = infoEstado[1];
```

Pero, con la técnica del destructuring, podemos escribir lo anterior en solo una línea:

```
const [valorEstado, actualizarEstado] = useState(0);
```

Paso 2: Eventos

Lo siguiente será aplicar los eventos que queremos que actualicen nuestro estado, en este caso, el click de un botón.

En Contador.jsx, agrega 3 onClick a los botones, y pasale el nombre de una función correspondiente. (aún no las hemos creado)

```
<section className="controls">
  <button onClick={incrementar}>Incrementar</button>
  <button onClick={decrementar}>Decrementar</button>
  <button onClick={reset}>Reset</button>
</section>
```

Paso 3: Manejadores de eventos

Crea la función “incrementar” como una función de flecha. Esta función, al ejecutarse, creará una nueva variable para calcular el “nuevoEstado”, el cual será igual al valor inicial, (originalmente seteado en 0), más el número 1, ya que el contador se actualiza en una unidad..

Luego, ejecutaremos la función actualizadora que nos da el hook useState, la cual toma como parámetro el nuevo valor para el estado.

```
const incrementar = () => {
  let nuevoEstado = valorEstado + 1;
  actualizarEstado(nuevoEstado)
};
```

Paso 4: Manejadores de eventos

Ahora deberemos crear los otros dos manejadores de cambio para las operaciones de decrementar y reset.

```
const decrementar = () => {
  let nuevoEstado = valorEstado - 1;
  actualizarEstado(nuevoEstado)
};
const reset = () => {
  let nuevoEstado = 0;
  actualizarEstado(nuevoEstado)
};
```

Paso 5: Asociar el 0 al estado

Para finalizar, tendremos que sustituir el valor 0 (*hardcodeado*)
a que lea de la variable “infoEstado”

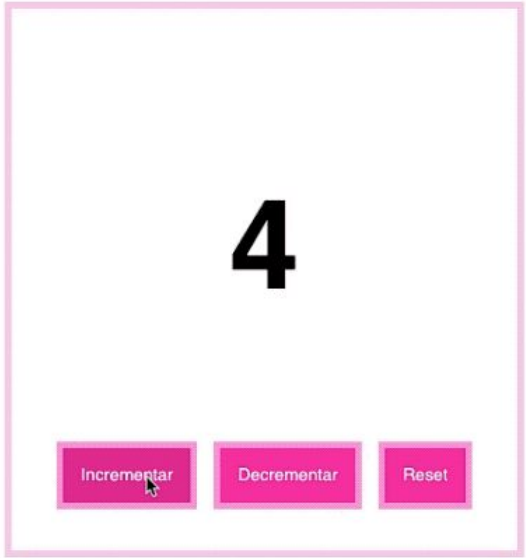
```
<p className="count">0</p>
```

↓

```
<p className="count">{valorEstado}</p>
```

¡Con estos cambios tendremos un contador funcionando!

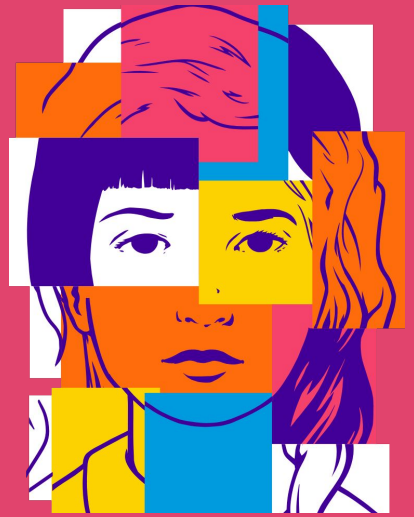
[Acá te dejamos el ejercicio finalizado.]



¡BREAK!

Ph. Credit: Drew Coffmann

Fechas en programación



Fechas en programación

Las encontramos dentro de cualquier rincón de un paquete de software. Se utilizan para registrar el punto exacto de una acción, por ejemplo, transacciones bancarias, pedidos de delivery o para denotar un rango, como la disponibilidad de un hotel.

La estructura completa de una fecha incluye:

```
14/02/2020   10:04:09:999   GMT-0300
```

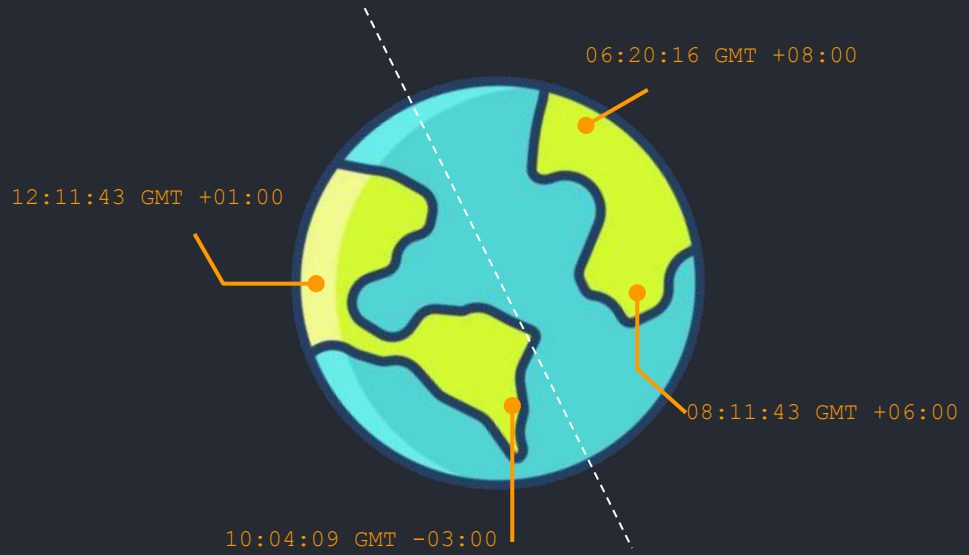
día mes año hs min seg ms timezone

Complejidad en fechas

Husos horarios, segundos bisiestos y horarios de verano son factores que complejizan el manejo de fechas en sistemas informáticos.

Encontramos un mayor reto cuando hay que mantener una uniformidad horaria en un mismo sistema que puede ser consultado en distintas partes del mundo, al mismo momento.

¿La solución para esto? **El tiempo UNIX.**



Tiempo UNIX

Es un valor, expresado en segundos, que empezó a contarse desde el **1 de Enero de 1970**, específicamente desde las **12:00am** de ese día.

Cuando un sistema informático (por ejemplo, aplicación web) quiere saber la fecha en un punto especifico, solo tiene que saber el tiempo UNIX actual y usar herramientas para traducir esa cantidad de segundos a una fecha estructurada.

`8/12/2010 7:44:28 P.M = 1.291.837.468 seg desde 1/1/1970`

Para el 8 de Diciembre del 2010, han pasado 1.291.837.468 segundos desde el 1 de Enero de 1970

Tiempo UNIX

1 de Enero de 1970
Segundo 0

11 de Mayo de 2003
1052700292 segundos desde tiempo UNIX

¿Cuál es el tiempo UNIX de hoy?

8 de Diciembre de 1996
850005892 segundos desde tiempo UNIX

24 de Septiembre de 2015
1443055492 segundos desde tiempo UNIX

Fechas en Javascript

Constructor `new Date();`

```
let fechaActual = new Date();
```

Produce un date string cuyo valor es la fecha del momento de su creación, según la hora del navegador o sistema operativo sobre el que se haya consultado.

El date string es un string especial que ofrece métodos que un string comun no tendria.

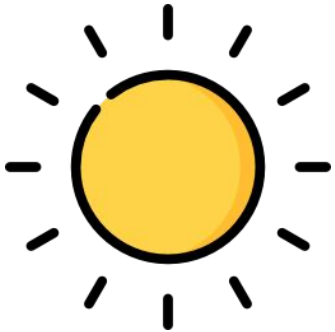
Exploremos esto, codeando.

```
//Tue Dec 08 2020 00:27:15 GMT-0300 (Argentina Standard Time)
```

Programamos

Una persona por ejercicio.

1 - El día de hoy



En un sandbox nuevo, de tipo “vanilla” (Javascript) construye una variable que almacene el día de hoy. Borra el código que Code Sandbox deja en index.js, y sustituyelo por:

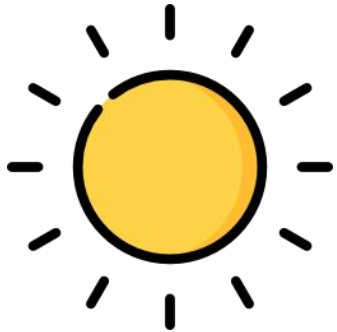
```
let hoy = new Date();
```

Haz un console.log de la variable “hoy”.

Lo que ves, es el date string. Es un string especial que tiene información sobre la hora en el momento que new Date() fue ejecutada.

```
Sat Aug 14 2021 11:28:52 GMT-0300 (Argentina Standard Time)
```

2 - El día de hoy, en UNIX



Ahora que tenemos la fecha de hoy, respondamos esta pregunta:

¿Cuál es el valor del tiempo UNIX en segundos, para esa fecha?

Para obtener el tiempo UNIX (la cantidad de segundos transcurridos desde el 1 de Enero 1970) utilizamos la función “getTime()” . En el sandbox anterior, agrega lo siguiente debajo del new Date()

```
let hoyEnUNIX = hoy.getTime();
```

Sin embargo, tenemos un problema. Javascript “razona” el tiempo en **milisegundos**, por lo que tendremos que hacer el cálculo para llevar este valor a **segundos**. Este cálculo se hace de esta manera:

```
let hoyEnUNIX_SEG = Math.floor(hoyEnUNIX_MS / 1000);
```

Loguea este valor por consola e introducelo en este convertidor para verificar que la fecha arrojada por “hoyEnUNIX_SEG” es la fecha en este momento.

3 - Tiempo UNIX de...

Ya sabemos cómo calcular el tiempo UNIX en segundos para la fecha del momento. Pero, ¿cómo podemos calcular el tiempo UNIX de una fecha específica, distinta de ahora?

Para esto, le pasamos un string al constructor `new Date()` que representa esa fecha en particular. En el sandbox escribe:

```
let diaDeLaTierra2021 = new Date("2021-4-22");
// sacamos el tiempo UNIX en milisegundos
let diaDeLaTierra2021_MS = diaDeLaTierra2021.getTime();
// convertimos de MS a SEG
let diaDeLaTierra2021_SEG = Math.floor(diaDeLaTierra2021_MS / 1000);
// logueamos por consola
console.log(diaDeLaTierra2021_SEG);
```

En JS, debemos especificar AÑO - MES - DIA

Verifica el valor de `diaDeLaTierra2021_SEG` en el convertidor de UNIX sea igual al 22 de Abril de 2021.

¡Atajo!

Si te parecen muchos pasos el cálculo de una fecha en segundos, lo puedes hacer directamente así:

```
let diaDeLaTierra2021_SEG = Math.floor(new Date("2021-4-22").getTime() / 1000);
```

4 1 2 3

Internamente, Javascript calcula:

1- El date string con el constructor new Date, a partir de una fecha específica
2- El tiempo UNIX con .getTime()
3- Divide el tiempo UNIX entre 1000 para calcular segundos
4- Redondea hacia el número entero más bajo con Math.floor

4 - ¡Atajo, funcional!

Alternativamente, puedes crear una función que te retorne los segundos UNIX.
Escribe esta función en el sandbox.

```
// recuerda que la fecha debe ser "AAAA-MM-DIA"
function calcularUNIXsegundos(fecha) {
  return Math.floor(new Date(fecha).getTime() / 1000);
}

let diaDeLaTierra2021_SEG = calcularUNIXsegundos("2021-4-22");
```

5 - Operaciones con fechas

Teniendo la posibilidad de calcular el tiempo UNIX de una fecha, vamos a determinar, con Javascript, si una fecha es menor, o mayor que otra. Declara estas dos fechas en el sandbox:

```
let fechaA = "1992-3-28";
let fechaB = "2020-6-13";
```

¿Cómo podemos determinar si una es mayor o menor que la otra? Lo primero es calcular el tiempo UNIX de cada una:

```
let fechaA_SEG = calcularUNIXsegundos(fechaA);
let fechaB_SEG = calcularUNIXsegundos(fechaB);
```

En estas dos variables tendremos dos ***timestamps***, cada una con el valor transcurrido desde el 1 de Enero de 1970. Por lo que una comparación numérica será suficiente:

```
if(fechaA_SEG > fechaB_SEG) {
  console.log(fechaA + " es mayor que " + fechaB);
} else {
  console.log(fechaB + " es mayor que " + fechaA);
}
```

Extra:

Métodos de date string

El date string tiene varios métodos para obtener información de la fecha creada, estos son algunos.

```
let fechaActual = new Date();

fechaActual.getDate() // Número del día
fechaActual.getDay() // Día de la semana (0 es Domingo, 1 es Lunes)
fechaActual.getFullYear() // Año
fechaActual.getMonth() // Mes (Enero es 0)
fechaActual.getHours() // Horas
fechaActual.getMinutes() // Minutos
fechaActual.getSeconds() // Segundos
fechaActual.getTime() // Tiempo UNIX en MILISEGUNDOS
```

Para la próxima

1) Termina el ejercicio de la meeting de hoy.
2) Lee la toolbox 30 y prepárate para la próxima meeting.
3) Resuelve el challenge

ACÁMICA