NODE JS

Funciones

adn POTENCIÁ TU FUTURO

Nación Servicios

IT Patagonia

Codeki™

Clase 13

01 Repaso Funciones

02 Alcance en funciones

03 Funciones de orden superior

04 Expresiones Lambda

Nación Servicios


Repaso - Funciones

01

Funciones en JavaScript

Las funciones son elementos básicos dentro del desarrollo informático, ya que nos permiten:

- Agrupar código de manera organizada.
- llevar a cabo operaciones de manera reiterada sin necesidad de reiteración de código.
- Deben brindar solución a un conjunto extenso de casos.

01

Anatomía de una funcion en JavaScript

```
function nombreDeLaFuncion(parametro1, parametro2) {
  var variableLocal = 10;
  return resultado;
}
var resultado = nombreDeLaFuncion(valor1, valor2);
```

01

Nomenclatura en funciones

Declaración de función:

Comienza con la palabra clave function, seguida del nombre de la función, paréntesis que pueden contener los parámetros separados por comas (opcional) y luego abre una llave { para definir el cuerpo de la función.

01

Nomenclatura en funciones

Nombre de la función:

Es el identificador único de la función. Puedes usar este nombre para invocar la función desde otras partes del código.

01

Nomenclatura en funciones

Parámetros (opcional):

Son variables que actúan como marcadores de posición para los datos que se pasan a la función cuando se la invoca. Los parámetros se definen entre paréntesis y separados por comas. Dentro del cuerpo de la función, puedes utilizar estos parámetros como variables locales.

01

Nomenclatura en funciones

Cuerpo de la función:

Aquí es donde defines la lógica de la función. Puedes realizar operaciones, manipular datos, utilizar condicionales, bucles, etc. dentro del cuerpo de la función.

01

Nomenclatura en funciones

Instrucción de retorno (opcional):

Si quieres que la función devuelva un valor, puedes utilizar la palabra clave return seguida del valor que deseas devolver. Si no hay una declaración de retorno o si no se proporciona un valor, la función devolverá undefined por defecto.

01

Nación Servicios

Nomenclatura en funciones

Llamada a la función:

Para ejecutar el código dentro de la función y obtener un resultado (si la función devuelve algo), simplemente escribe el nombre de la función seguido de paréntesis que contengan los valores que deseas pasar como argumentos a los parámetros de la función.

Clase 13

01

Repaso Funciones

02

Alcance en funciones

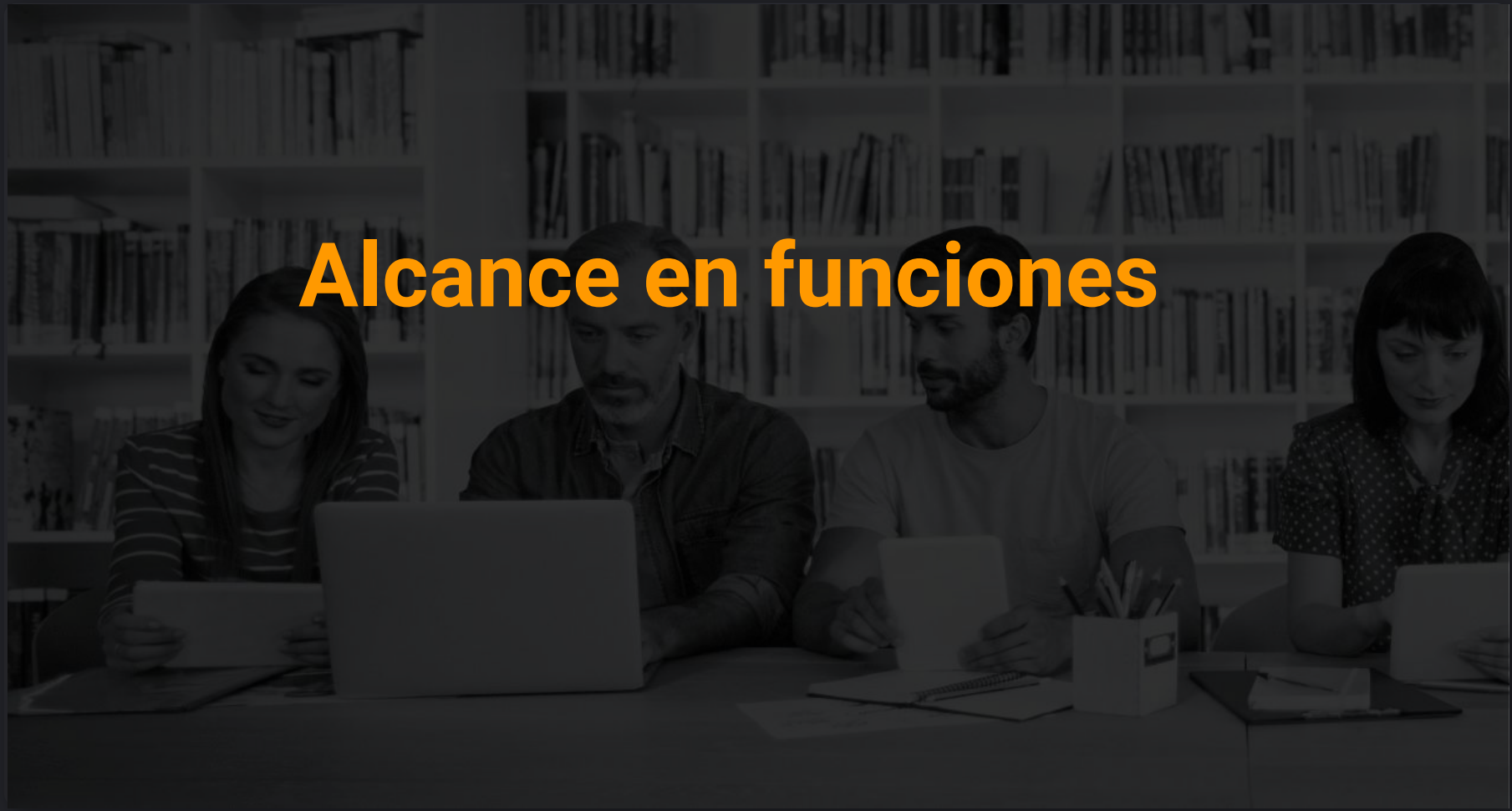
03

Funciones de orden superior

04

Expresiones Lambda

Nación Servicios


Alcance en funciones

02

Definición de alcance (Scope)

La traducción lineal de esta palabra, significa **“alcance”** . Este término, da cuentas del entorno donde nuestra función o variable, está disponible.

Puntualmente en **JavaScript**, debemos saber que cada función, **genera** su propio **scope**, su propio alcance .

02

Definición de alcance (Scope)

Cuando las distintas variables son definidas, dentro de una función, dejan de ser accesibles desde el exterior de la misma.

Por lo tanto, los comportamientos entre var, let y const, se tornan similares entre sí.

Variables globales

En JavaScript, las funciones (al igual que los objetos) también son variables. Y lo que está por fuera de su alcance, se denomina “global”, permitiendo su acceso, desde múltiples etapas de nuestro archivo.

```
let carName = "Volvo";
myFunction();

function myFunction() {
  document.getElementById("demo").innerHTML = "I can display " + carName;
}
```

Clase 13

01

Repaso Funciones

02

Alcance en funciones

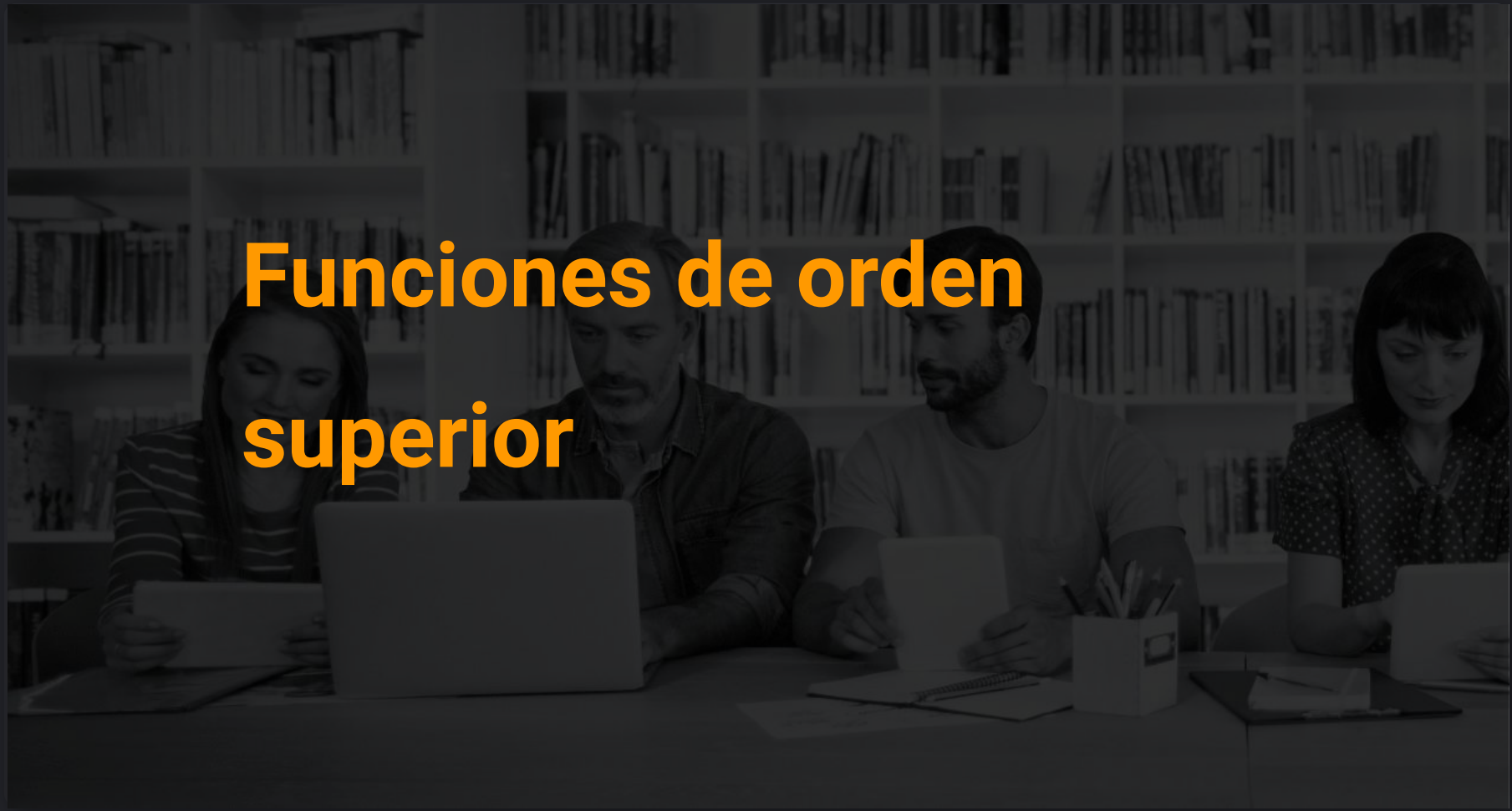
03

Funciones de orden superior

04

Expresiones Lambda

Nación Servicios


Funciones de orden
superior

03

Definición

Son funciones que operan a partir de otras funciones. Esto implica dos detalles importantes:

- Habrá funciones que acepten otras funciones como argumento.
- Habrá funciones que retornen una función.

03

Definición

Si repasamos lo recientemente enunciado, veremos qué esto está estrechamente relacionado al detalle de qué JavaScript trata a las funciones como variables.

Funciones de orden superior y callback

Se denomina “callback” a una función que es pasada como argumento a otra función.

Mientras que una función de orden superior es aquella qué toma, al menos, una función como entrada y/o retorna una funcion como salida.

Clase 13

01

Repaso Funciones

02

Alcance en funciones

03

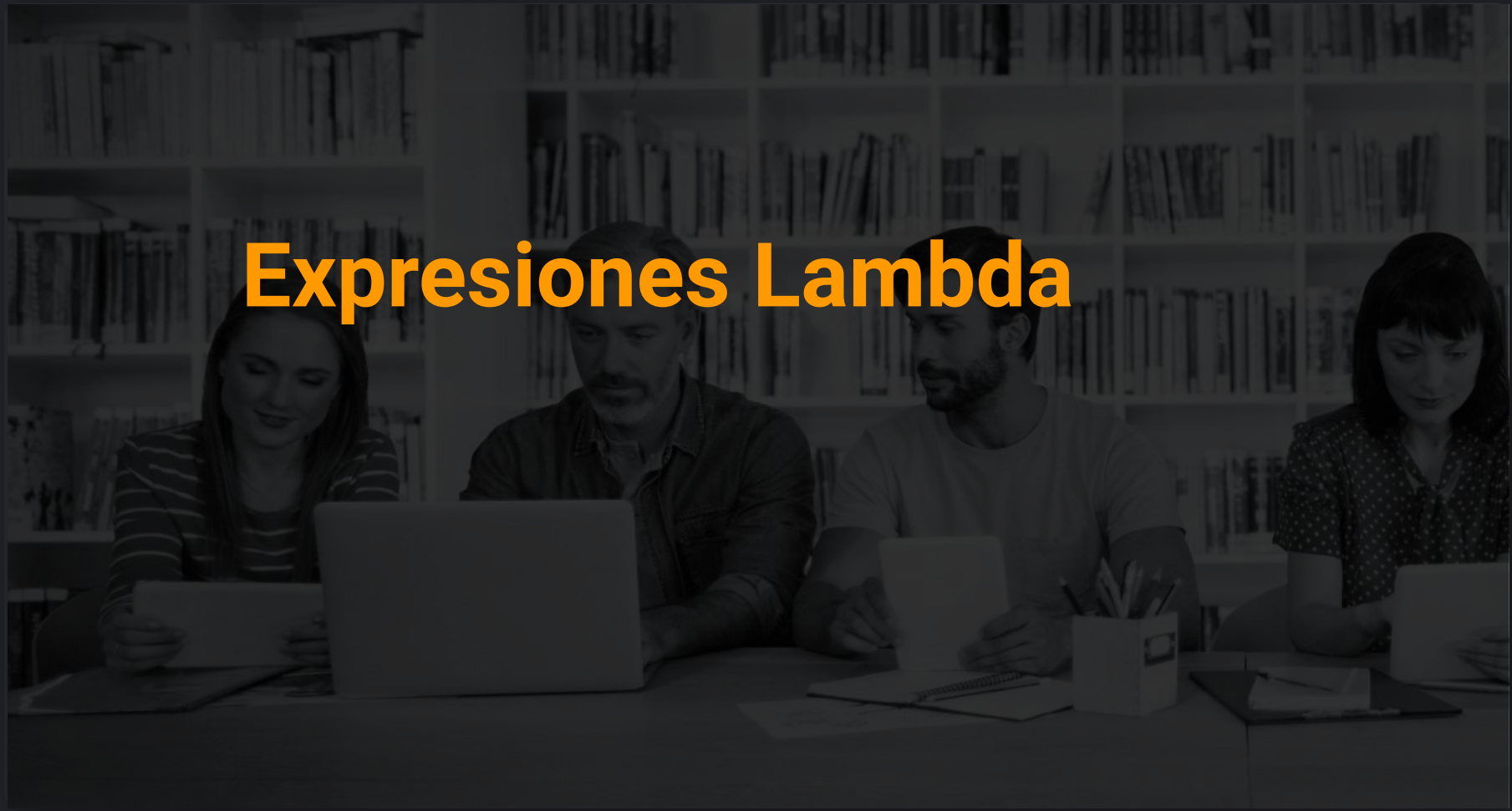
Funciones de orden superior

04

Expresiones Lambda

Nación Servicios

Expresiones Lambda



04

Expresiones Lambda

También conocidas como funciones flecha o **arrow functions**, permiten definir funciones de manera más concisa. Estas expresiones tienen varias ventajas, tales como:

- Sintaxis más breve
- Elemento inicial, hacia la programación funcional
- Comportamiento especial con respecto al alcance de las variables y la palabra clave this
- Usadas en colecciones, permiten procesar, sin modificar el contenido de base

04

Sintaxis en expresiones Lambda

nombreFuncion(parametro1, parametro2, ...) ⇒ expresión

Ejemplo:

```
const saludar = () ⇒ console.log("Hola, mundo!");
saludar();
```

04

Ejemplo con variables

```
const cuadrado = x ⇒ x * x;
console.log(cuadrado(5)); // 25


const suma = (a, b) ⇒ a + b;
console.log(suma(3, 7)); // 10
```

04

Uso en colecciones

La mayor potencialidad, de este tipo de funciones, aparece cuando se las emplea con colecciones, ya qué permite operar de manera agil, llevar adelante múltiples etapas de procesamiento y todo esto sin alterar el contenido de base. Por ejemplo:

```
const numeros = [1, 2, 3, 4, 5];
const duplicados = numeros.map(num ⇒ num * 2);
console.log(duplicados); // [2, 4, 6, 8, 10]
```

04

Nación Servicios

Uso en colecciones

Podemos observar que el contenido de “números[]” no se modificó en absoluto, generamos un nuevo arreglo, llamado duplicados, a partir del uso de map(), sin modificar la información inicial.

Programación funcional

Si bien es una idea bastante avanzada, con los conceptos desarrollados anteriormente, estamos sentando una base en este paradigma. Debemos saber qué (así como también mencionamos para las expresiones lambda) su uso, al momento de trabajar con colecciones es muy difundido.

En este contexto es qué entra la función map(), capaz de correlacionar elementos entre sí (mapear)

Programación funcional

Además de map() encontramos las siguientes:

- filter(): filtra según criterios
- reduce(): permite acortar colecciones
- forEach(): mayormente difundida, empleada para recorrer
- find(): Encontrar
- some(), every(): Empleadas para saber si uno o todos los elementos cumplen con una condición, respectivamente

¡Muchas gracias!