Fase 6: Hooks, Callbacks y Formularios

Hooks, Callbacks y Formularios

Partimos de...

- Árbol de componentes *presentacionales*
- Todo el contenido dinámico se carga desde JSON
- Literales y precios internacionalizados
- *Storybook* funcionando y *tests verdes*

Hooks, Callbacks y Formularios

Queremos...

- Introducir *interactividad* en la aplicación
  - Botones
  - Formularios
  - Select
- Recibir *callbacks* para enlazar con la *lógica de negocio*

Hooks, Callbacks y Formularios

Callbacks

- Algunos átomos reciben funciones como props
  - onClick (*button*)
  - onChange (*input*, *select*)

```
import React from 'react'
import PropTypes from 'prop-types'
import cx from 'classnames'
import { identity } from 'lodash'

const Button = ({ action, children, fullwidth, className }) => (
  <button className={cx("button", { fullwidth }, className)}
          onClick={() => action()}>
    <span>{ children }</span>
  </button>
)

Button.propTypes = {
  children: PropTypes.node.isRequired,
  fullwidth: PropTypes.bool,
  className: PropTypes.string,
  action: PropTypes.func
}
```

Hooks, Callbacks y Formularios

Callbacks

- Las *vistas* también reciben *callbacks*
  - Las *operaciones* que se pueden hacer en la página
  - Delegan en la *capa de servicios* de la app
  - Responsables de *invocar* en el *momento* adecuado...
  - ...y con los *parámetros* adecuados

```
const ProductDetails = ({ data, shoppingCart, onAddToCart }) => {
  return (
    <AppLayout shoppingCart={shoppingCart}>
      <ProductInfo product={data}
                   quantity={1}
                   onAddToCart={
                     () => onAddToCart(data.id, 1)
                   }/>
      <RelatedProducts products={data.relatedProducts} />
    </AppLayout>
  )
}
```

```
const ProductInfo = ({ product, quantity, onAddToCart }) => (
  <div className="product-details">

    <ProductTitle product={product} />
    <ProductPrice product={product} />
    <ProductPicture product={product} />
    <ProductAvailability product={product} />
    <ProductDescription product={product} />

    <div className="product-actions">
      <ProductQuantity quantity={quantity}/>
      <Button action={onAddToCart}>
        <FormattedMessage id="button:add-to-cart"/>
      </Button>
    </div>
  </div>
)
```

Hooks, Callbacks y Formularios

Para *testear* que funciona correctamente...

1. Creamos una función *espía* con jest.fn()
2. La pasamos como callback al componente
3. Lanzamos el *evento* correspondiente con *enzyme*
4. Comprobamos si *espía* se ha invocado correctamente

```
import React from 'react'
import { intlMount } from '../../lib/intl-mount'
import ProductDetails from '../../../src/ui/views/ProductDetails'
import data from '../../data/product-details.json'

describe('<ProductDetails/>', () => {

  it('invokes the onAddToCart callback when clicked', () => {
    const spy = jest.fn()
    const wrapper = intlMount(
      <ProductDetails data={data}
                      shoppingCart={shoppingCart}
                      onAddToCart={spy}/>
    )
    wrapper.find('Button').simulate('click')
    expect(spy).toHaveBeenCalledWith(data.id, 1)
  })

})
```

Hooks, Callbacks y Formularios

Para visualizar los callbacks en Storybook...

- Vamos a instalar *@storybook/addon-actions*
  - Nos permite visualizar invocaciones y parámetros

Hooks, Callbacks y Formularios

Añadir a *.storybook/addons.js*

```
import '@storybook/addon-knobs/register'
import '@storybook/addon-actions/register'
```

```
import React from 'react'
import { storiesOf } from '@storybook/react'
import { action } from '@storybook/addon-actions'
import ProductInfo from '../organisms/ProductInfo'
import productDetailsData from
  '../../../.storybook/data/product-details.json'

storiesOf('Organisms/ProductDetails', module)

  .add('ProductInfo', () => (
    <ProductInfo
      product={productDetailsData}
      quantity={1}
      onAddToCart={action('addToCart')}
    />
  ))
```

STORYBOOK ⌘

Filter

- Atoms
  - ProductAvailability
  - ProductDescription
  - ProductInfo
  - ProductPicture
  - ProductPrice
  - ProductThumbnail
  - ProductTitle
  - ProductTotal
  - VerticalSeparator
  - Subtitle
  - BetaHeader
  - CardRow
  - FormField
  - Button
  - PaginationItem
  - Money
- Layouts
- Molecules
- Organisms
  - AppFooter

Los músicos de Bremen

17,00 $

Disponibilidad: Disponible

Nuevo concepto de libro digital diseñado para su uso en dispositivos electrónicos. Se ha desarrollado con una navegación optimizada para enriquecer el aprendizaje en estos soportes.

KNOBS ACTION LOGGER

3 addToCart: []

CLEAR

Hooks, Callbacks y Formularios

Formularios

- Por regla general, las *vistas* gestionan los formularios
  - gestional el valor de los *input* y *select*
  - reaccionan a los *onChange*
  - aplican *validaciones* y controlan *errores*

```
const ProductDetails = ({ data, shoppingCart, onAddToCart }) => {
  return (
    <AppLayout shoppingCart={shoppingCart}>
      <ProductInfo product={data}
                   onChange={newQuantity => void 0}
                   quantity={1}
                   onAddToCart={
                     () => onAddToCart(data.id, 1)
                   }/>
      <RelatedProducts products={data.relatedProducts} />
    </AppLayout>
  )
}
```

```
class ProductDetails extends React.Component {
  constructor() {
    this.state = { quantity: 1 }
    this.updateQuantity = this.updateQuantity.bind(this)
  }
  updateQuantity(quantity) {
    this.setState({ quantity })
  }
  render() {
    const { quantity } = this.state
    return (
      <AppLayout shoppningCart={shoppingCart}>
        <ProductInfo product={data}
                     quantity={quantity}
                     setQuantity={this.updateQuantity}
                     onAddToCart={
                       () => onAddToCart(data.id, quantity)
                     }/>
        <RelatedProducts products={data.relatedProducts} />
      </AppLayout>
    )
  }
}
```

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

const ProductDetails = ({ data, shoppingCart, onAddToCart }) => {
  const [quantity, setQuantity] = useState(1)
  return (
    <AppLayout shoppingCart={shoppingCart}>
      <ProductInfo product={data}
                   quantity={quantity}
                   setQuantity={setQuantity}
                   onAddToCart={
                     () => onAddToCart(data.id, quantity)
                   }/>
      <RelatedProducts products={data.relatedProducts} />
    </AppLayout>
  )
}
```

Hooks, Callbacks y Formularios

Callbacks de la App

ProductList ⇒ ninguno

ProductDetails ⇒ *onAddToCart*

ShoppingCart ⇒ *onModifyItemQuantity*, *onRemoveItem*

Checkout ⇒ *onCheckout*

Hooks, Callbacks y Formularios

onAddToCart(*productId*, *quantity*)

- *ProductDetails*
- Invocación...
  - El usuario pulsa en botón *Add To Cart*
  - *productId*: id del producto mostrado
  - *quantity*: cantidad seleccionada en el selector

Hooks, Callbacks y Formularios

onModifyItemQuantity(*itemId*, *quantity*)

- *ShoppingCart*
- Invocación...
  - El usuario modifica un selector de cantidad
  - *productId*: id del producto mostrado
  - *quantity*: cantidad seleccionada en el selector

Hooks, Callbacks y Formularios

onRemoveItem(*itemId*)

- *ShoppingCart*
- Invocación...
  - El usuario pulsa el botón (x)
  - *productId*: id del producto a eliminar

Hooks, Callbacks y Formularios

onCheckout(*values*)

- *Checkout*
- Invocación...
  - El usuario pulsa el botón *Purchase*
  - *values*: los valores del formulario (con delivery)
  - **IMPORTANTE:** sólo se invoca **si no hay errores**

Ejercicio

Implementa la invocación a los 4 callbacks

- Utiliza hooks para gestionar formularios y selectores
- Valida los campos de Checkout (y muestra errores)
- Testea los callbacks de vistas y organismos
- Muestra las acciones en storybook