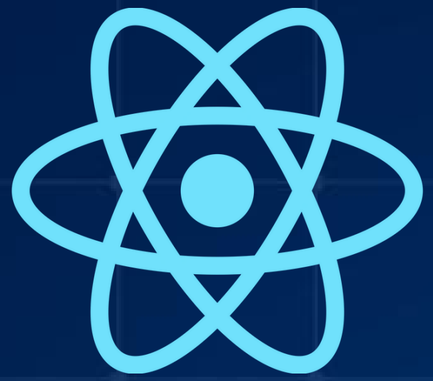
Secretaría de Desarrollo Económico Sustentable
Grandeza de México

Universidad Nacional Autónoma de México
Escuela Nacional de Estudios Superiores

IECAGto
Instituto Estatal de Capacitación



DESARROLLO WEB CON REACT

Temario

Fundamentos de desarrollo web

Javascript

Aplicaciones con React

Hooks y navegabilidad

Consumo de Web APIs

Herramientas en la nube

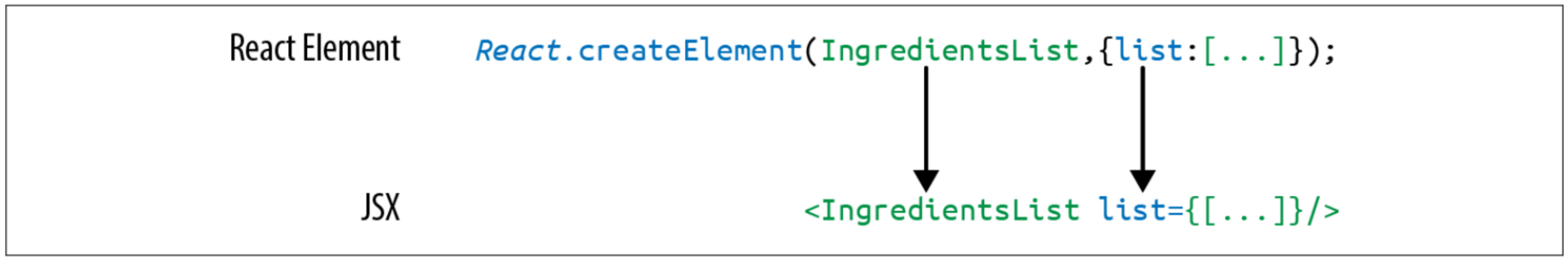
Sitios estáticos y SSR

Aplicaciones en tiempo real

Aplicaciones con React

JavaScipt XML (JSX)

JSX es una extensión de JavaScript que nos permite definir elementos de React usando una notación basada en etiquetas directamente en nuestro código **JavaScript**. Evita el uso excesivo del método **createElement**. Su parecido con HTML lo hace una herramienta útil a la hora de crear interfaces.



Características de JSX

Componentes anidados: Los componentes creados pueden añadirse como hijos de otros componentes y pueden ser añadidos más de una vez.

```
<IngredientsList>
  <Ingredient />
  <Ingredient />
  <Ingredient />
</IngredientsList>
```

Expresiones de JavaScript: Se pueden añadir expresiones de JavaScript incluyéndolas dentro de llaves **{}** para los atributos de cada elemento y en el cuerpo del componente.

```
<h1>{title}</h1>
```

```
<input type="checkbox" defaultChecked={false} />
```

Características de JSX

Uso de className: A diferencia de HTML, no se puede utilizar la etiqueta class debido a que es una palabra reservada de JS; en su lugar se usa **className**.

Mapeo de arreglos: Los arreglos se pueden mapear en forma de elementos de **JSX** directamente.

```
<IngredientsList>
  <Ingredient />
  <Ingredient />
  <Ingredient />
</IngredientsList>
```

```
<ul>
  {props.ingredients.map((ingredient, i) => (
    <li key="{i}">{ingredient}</li>
  ))}
</ul>
```

Integración con Babel

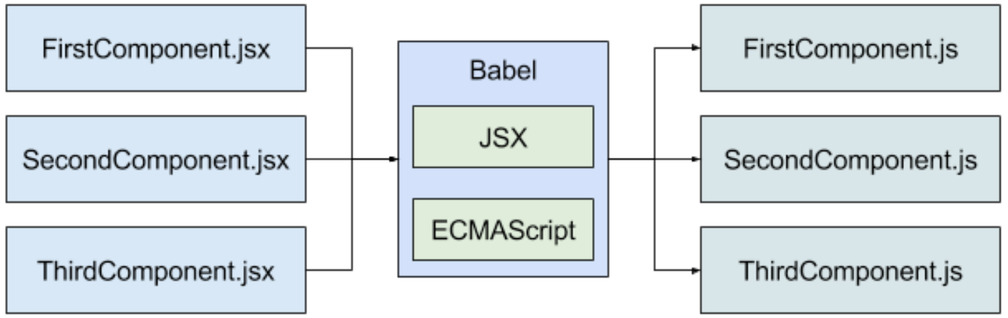
JSX no puede ser interpretado directamente por ningún navegador web en la actualidad, por lo que es necesario utilizar un **compilador de JS** para traducir el contenido JSX a algo que los navegadores puedan entender. La manera más sencilla de integrarlo es mediante el CDN de Babel.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>React Examples</title>
  </head>
  <body>
    <div id="root"></div>

    <!-- React Library & React DOM -->
    <script
  src="https://unpkg.com/react@16.8.6/umd/react.development.js">
  </script>
    <script
  src="https://unpkg.com/react-dom@16.8.6/umd/react-dom.development.js">
  </script>
    <script
  src="https://unpkg.com/@babel/standalone/babel.min.js">
  </script>

    <script type="text/babel">
      // JSX code here. Or link to separate JavaScript file that contains JSX.
    </script>
  </body>
</html>
```

Compilación en JavaScript

Conforme se lanzan actualizaciones del lenguaje, es posible que no todos los navegadores lo soporten, por lo que se han creado herramientas para traducir estas características en código compatible a nivel general, este proceso es llamado **compilación**.

En la actualidad existen herramientas para la compilación de JavaScript como **Babel**.



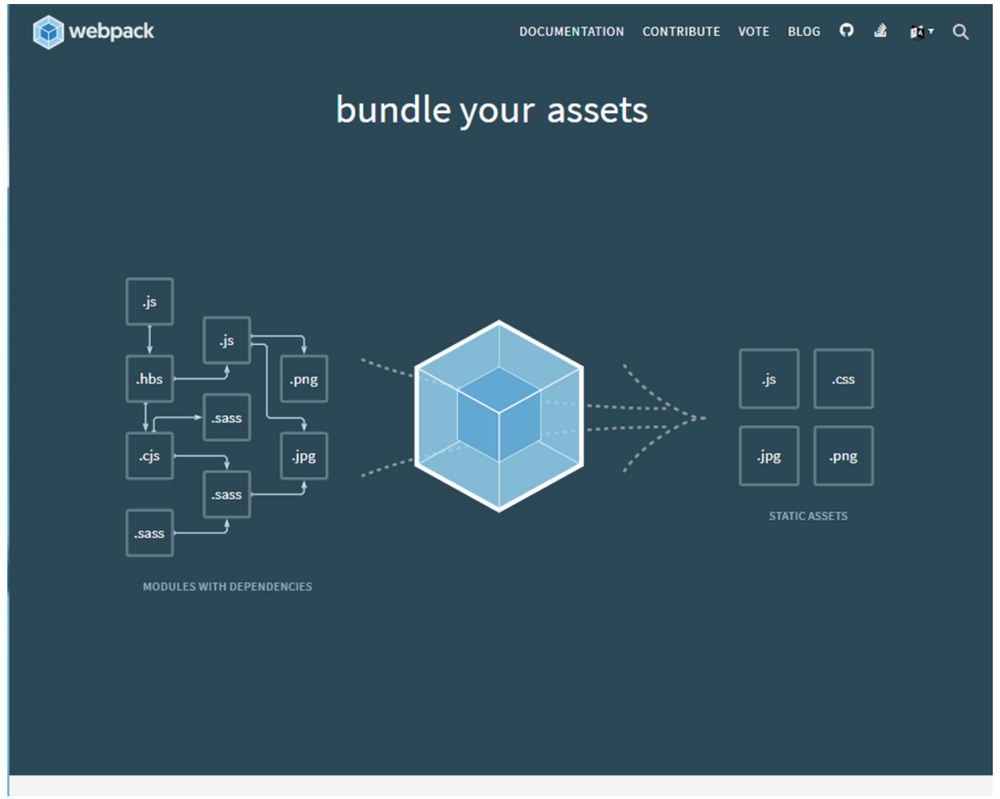
Ejercicio en clase

Primera aplicación usando JSX

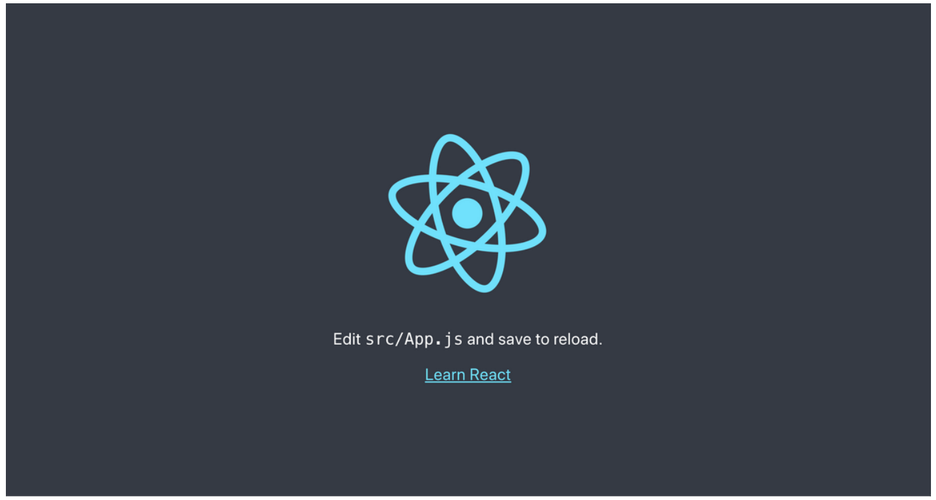
Introducción a webpack

Webpack se define como un **«empaquetador de módulos»**, cuya finalidad es tomar todos los archivos de un módulo específico y convertirlos en uno solo permitiendo mejorar el rendimiento de la red.

Su utilidad es visible en la puesta en producción de proyectos reales de JavaScript, como lo son los de React.



Create React App



Create React App (CRA) es una herramienta de línea de comandos (CLI) desarrollada por Facebook que permite crear rápidamente una aplicación de React con una configuración predefinida y lista para usar. CRA facilita el proceso de configuración y estructura de un proyecto de React, eliminando gran parte del trabajo inicial de configuración.

NPX

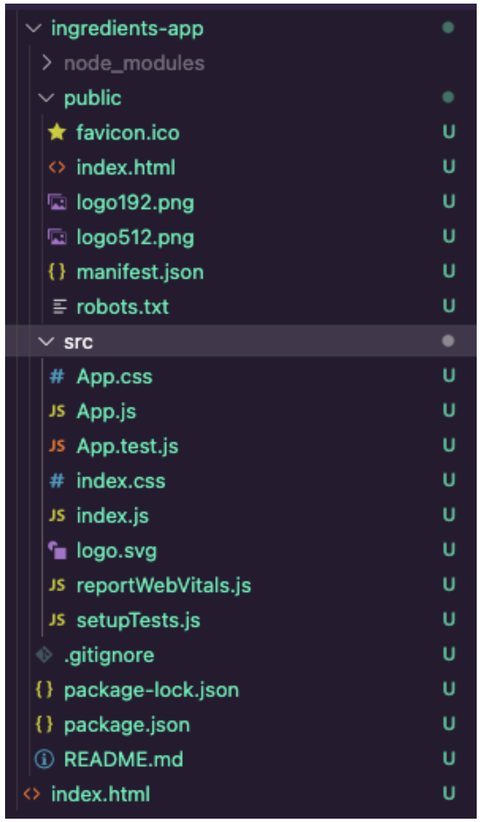
NPX es una herramienta de línea de comandos que viene incluida con npm a partir de la versión 5.2.0. Su propósito principal es permitirte ejecutar paquetes de Node.js sin necesidad de instalarlos globalmente en tu sistema.


```
npx create-react-app my-app
cd my-app
npm start
```

Estructura de un proyecto CRA

Algunos de los archivos y directorios notables en el directorio raíz son:

- **package.json**: Este archivo mantiene las dependencias del proyecto, scripts de desarrollo, configuraciones y otra información relacionada.
- **public**: Este directorio contiene archivos estáticos que se sirven directamente al navegador, como el archivo HTML base, imágenes o íconos.
- **src**: Este directorio es donde se encuentra la mayor parte del código fuente del proyecto.
- **node_modules**: Este directorio es donde se almacenan las dependencias del proyecto instaladas a través de npm o Yarn.

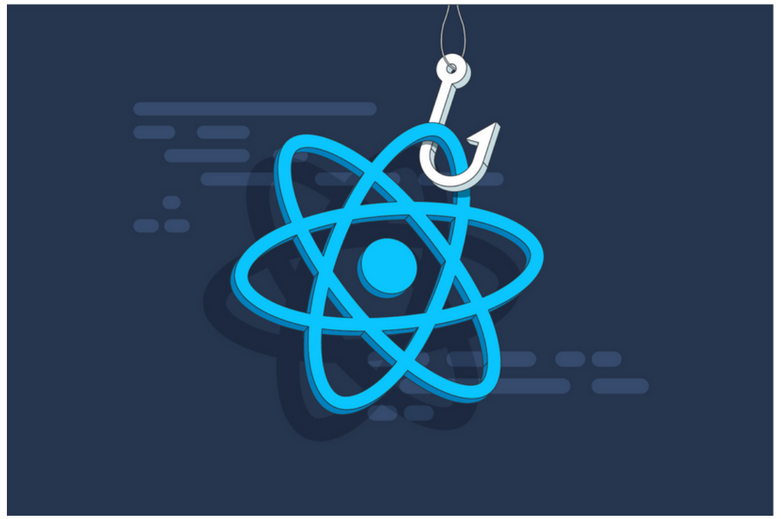


Ejercicio en clase

Aplicación web con React para dar de alta productos usando Create React App

Hooks

Los **hooks** son elementos reutilizables de código que incluyen lógica no incorporada en el árbol de componentes. Permiten **enganchar (hook up)** funcionalidad a nuestros componentes.



React contiene múltiples hooks predefinidos que se pueden utilizar directamente en los componentes con solamente importarlos.

Generalmente los hooks se definen con el prefijo **use**, por ejemplo: **useEffect, useState, useCallback**, etc.

useState

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

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

Este hook permite agregar **estado local** a un componente. Proporciona una forma de declarar y actualizar variables de estado en el cuerpo del componente. Al usar useState, se obtiene una referencia al estado actual y una función para actualizarlo.

useEffect

Este hook permite realizar efectos secundarios en un componente funcional. Se utiliza para ejecutar código después de que el componente se haya renderizado en el DOM y también para limpiar esos efectos cuando el componente se desmonte.

```jsx
import React, { useState, useEffect } from 'react';

function Timer() {
  const [seconds, setSeconds] = useState(0);

  useEffect(() => {
    const timer = setInterval(() => {
      setSeconds(prevSeconds => prevSeconds + 1);
    }, 1000);

    return () => clearInterval(timer);
  }, []);

  return <p>Seconds: {seconds}</p>;
}
```

Dependencias de useEffect

El segundo parámetro del hook **useEffect** hace referencia al **arreglo de dependencias**, este determina cuándo se va a ejecutar el código del efecto secundario señalado.

Sin arreglo: Se ejecuta cada vez que se renderiza el componente.

[]: Se ejecuta solo durante el primer renderizado.

[...params]: Se ejecuta cada vez que cambia el valor de una de las variables pasadas como parámetro

```
const [name, setName] = useState('');
const [age, setAge] = useState(0);

useEffect(() => {
  // Este efecto se ejecuta cuando 'name' o 'age' cambian
  console.log('El valor de name o age ha cambiado');
}, [name, age]);
```

Ejercicio de práctica - JSX

Modificar el ejercicio de recetas para crearlo como una aplicación de Create React App utilizando JSX, y los hooks useEffect y useState.

Secretaría de Desarrollo Económico Sustentable

Grandeza de México

Escuela Nacional de Estudios Superiores

IECAGto

Instituto Estatal de Capacitación

¡GRACIAS!