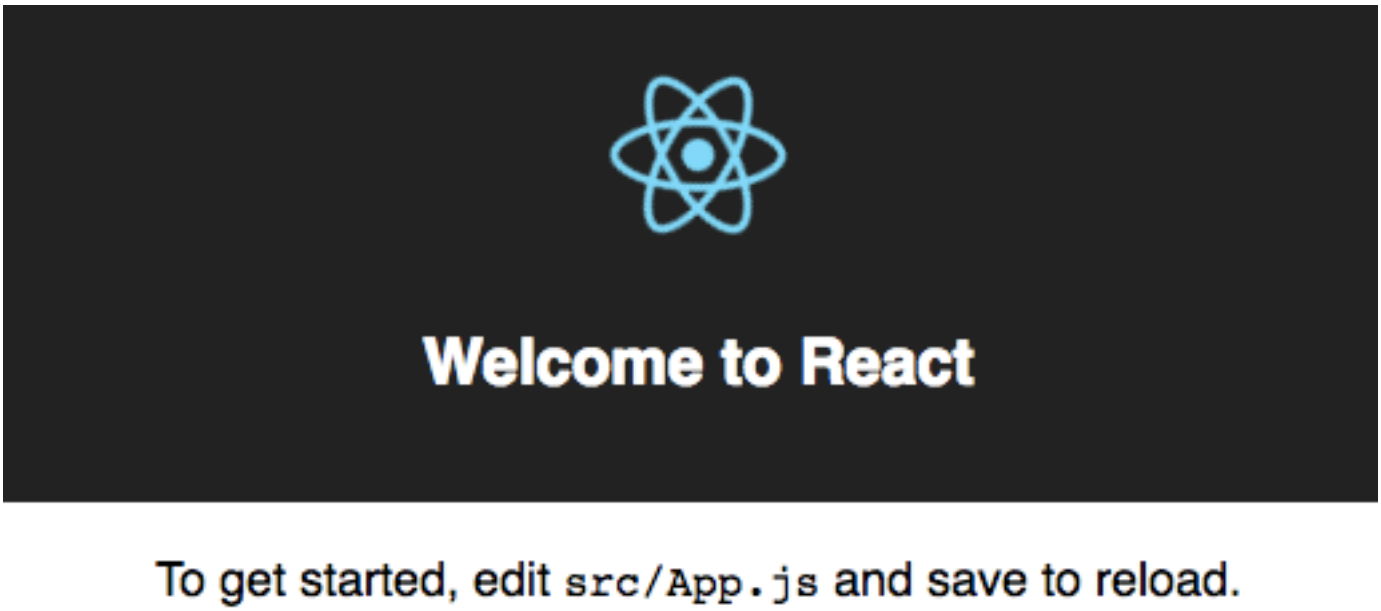
Ejercicios del Tema 1

Vamos a comenzar creando la típica aplicación de *Hola Mundo*. Crea un nuevo proyecto de ReactJS desde la consola tal y como se ha explicado en el tema 1 y, una vez creado, sitúate en la nueva carpeta. Prueba a ejecutar el proyecto que acabas de crear y verás que se abre una ventana de tu navegador. Deberías visualizar algo similar a esto:



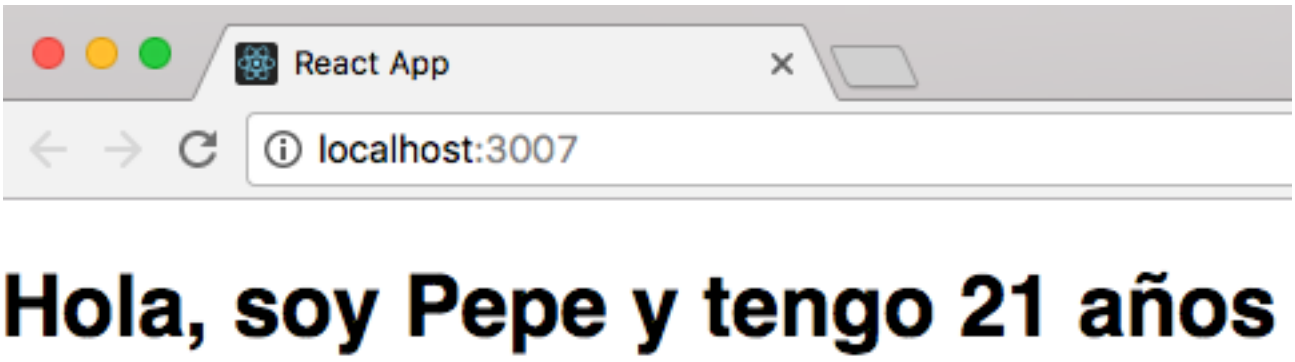
Esta es la visualización por defecto para cualquier proyecto React recién creado. Como podrás comprobar, el archivo App.js es cargado por defecto para mostrar la vista principal. Esto se debe a que el fichero index.js lo importa y lo renderiza.

Ejercicio 1 - *Cambia el título principal por el de "¡Hola Mundo!" y el texto inferior por "Esta es mi primera aplicación con React".*

Ejercicio 2 - *Crea una constante Persona que tenga dos atributos: nombre y edad. Implementa una función que para una Persona dada devuelva la frase siguiente:*

```
¡Hola! Soy nombre y tengo edad años.
```


Ejercicio 3 - *Modifica el código para que al cargar la aplicación se muestre únicamente la frase anterior. El resultado final deberá parecerse al siguiente:*



Ejercicio 4 -

Ahora que ya hemos alcanzado cierta confianza con la estructura básica de nuestra aplicación ReactJS, vamos a trabajar con las posibilidades que nos ofrece JSX para el diseño de nuestras interfaces.

Para este nuevo ejercicio vamos a generar un nuevo proyecto de React y vamos a trabajar con el fichero **/src/index.js**

Como todavía no hemos entrado en detalle en la definición de componentes en React, vamos a modificar el código de dicho fichero para que quede de la siguiente manera:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';

ReactDOM.render(
  <div>
    Contenido
  </div>,
  document.getElementById('root')
);
```

De esta manera, tendremos un script sencillo con el que podemos experimentar.

Si ejecutamos la aplicación con **npm start**, veremos la palabra *Contenido* en el navegador.

En este caso, aprovecharemos algunas de las ventajas que nos ofrece ES6, la nueva versión de Javascript para practicar con elementos un poco más complejos.

- El primer paso será crear una clase Persona, mediante la cual podremos generar instancias que posteriormente visualizaremos en nuestra aplicación.

Para ello, generamos el fichero **Persona.js** en el mismo directorio donde estamos trabajando.

El contenido podría ser el siguiente

```
class Persona{
  constructor (nombre, apellidos, edad){
    this.nombre = nombre;
    this.apellidos = apellidos;
    this.edad = edad;
  }

  nombreCompleto(){
    return this.nombre + " " + this.apellidos;
  }

  hablar(){
    return "La persona " + this.nombreCompleto() + " está
hablando";
  }
}

export default Persona;
```

Dentro de la clase Persona definimos su **constructor**, donde podemos inicializar todas las propiedades de nuestros objetos. En este caso, incluimos 3 propiedades (nombre, apellidos, edad).

Definimos dos métodos dentro de la clase, los cuales devuelven diferentes cadenas de caracteres representativas de nuestros objetos de tipo Persona.

Es muy importante incluir la línea con el export, la cual nos va a permitir a posteriori importar esta clase para su uso en los diferentes componentes de nuestro proyecto.

Si volvemos a nuestro fichero **index.js** e importamos la clase, podemos utilizarla para visualizar los datos de la Persona creada.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';

import Persona from './Persona';

let persona = new Persona('Mario', 'Girón', 33);

ReactDOM.render(
  <div>
    {persona.hablar()}
  </div>,
  document.getElementById('root')
);
```

Importamos la clase, generamos una instancia y dentro del elemento a renderizar, llamamos al método hablar de la clase, obteniendo como resultado la cadena que nos muestra los datos de la Persona.

Aparte de revisar la exportación de las clases dentro de un proyecto, demostramos con este ejemplo cómo podemos lanzar llamadas a métodos específicos dentro de la vista generada para nuestra aplicación.

Otra de las características más llamativas de Javascript es el hecho de poder definir **herencia** entre diferentes clases. Nos será de bastante utilidad más adelante.

En este pequeño ejemplo vamos a implementar la clase **Adulto**, la cual va a heredar todas las propiedades y métodos de Persona y además va a disponer de dos propiedades nuevas (trabajo, coche) y el método **mostrar**.

Generamos el fichero **Adulto.js** con la siguiente implementación.


```
import Persona from './Persona';

class Adulto extends Persona{

  constructor(nombre, apellidos, edad, trabajo, coche){
    super(nombre, apellidos, edad);
    this.trabajo = trabajo;
    this.coche = coche;
  }

  mostrar(){
    return "La Persona " + this.nombreCompleto() + " es un
Adulto y trabaja en '" + this.trabajo +"'. ¿Tiene coche? "
+ this.coche;
  }
}

export default Adulto;
```

En este caso, es importante el uso que le damos a la palabra reservada **extends** para definir cual es la clase de la que estamos heredando.

Mediante la palabra reservada **super** llamamos al método correspondiente de la clase padre. (Si usamos super dentro del constructor, estamos invocando al constructor de la clase superior).

Si importamos la clase nueva dentro de nuestro script principal, podremos generar instancias y posteriormente usarlas en nuestra vista.

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';

import Adulto from './Adulto';

let adulto = new Adulto('Pepe', 'Martinez', 29, 'Desarrollador', true);

ReactDOM.render(
  <div>
    {adulto.mostrar()}
  </div>,
  document.getElementById('root')
);
```


AMPLIACIÓN

- Genera la clase **Joven**, extendiendo de la clase Persona.
- Define nuevos atributos:
    - **colegio**: nombre del colegio
    - **numJuguetes:** número de juguetes que tiene.
- Crea un nuevo método:
    - **mostrar:** método que nos permita mostrar los datos del Joven

- Dentro de **index.js** genera un array con varios objetos de la clase Adulto y de la clase Joven. Genera un elemento para poder recorrer dicho array y mostrar en la vista todos los objetos.

