
# Framework para construir aplicaciones web modernas en React

# El objetivo del curso

Que puedas construir una app con React y Next.JS de principio a fin

# ¿Qué tengo que saber?

HTML, CSS, Javascript

No hace falta React :D

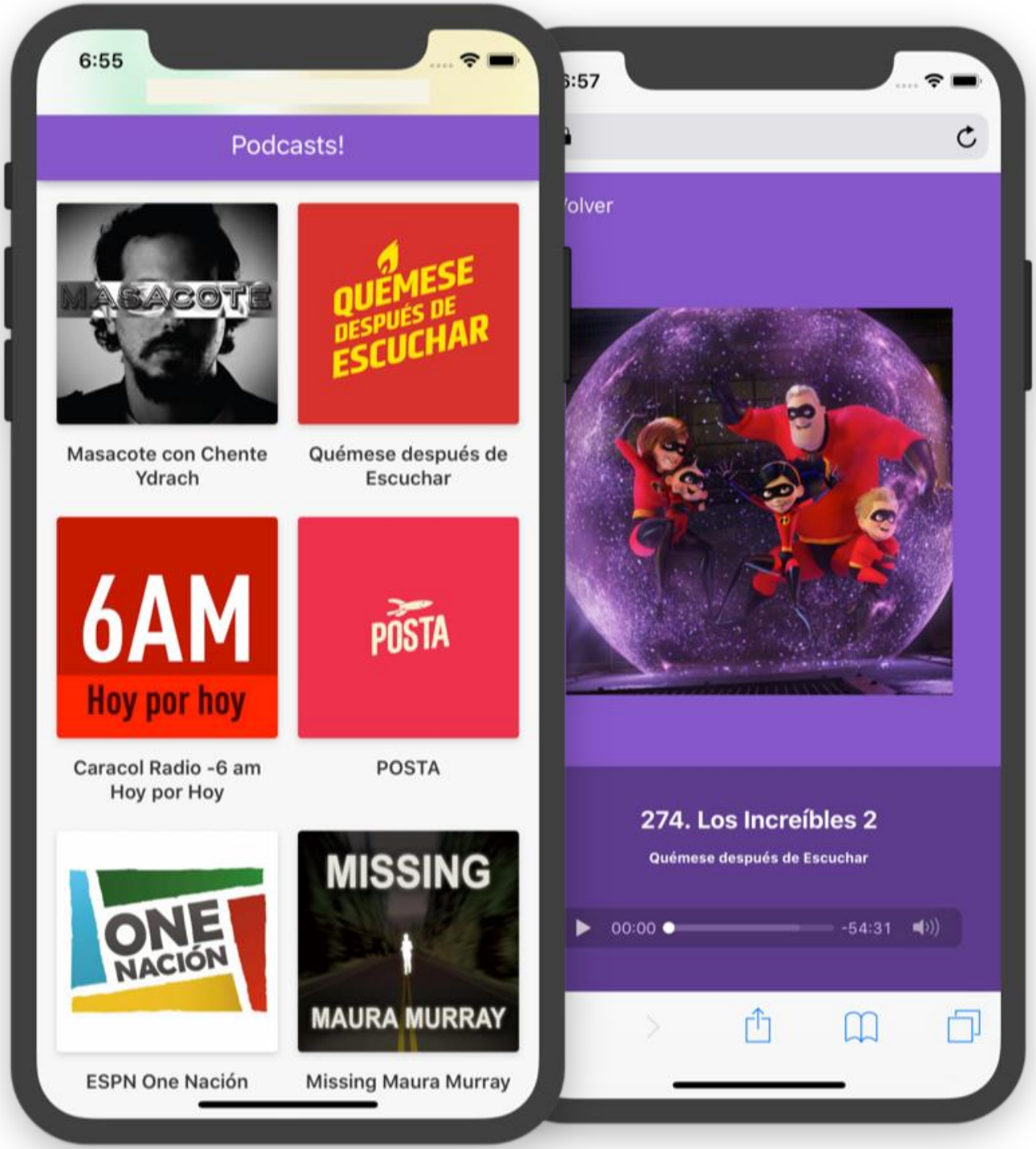
# ¿Qué vamos a aprender?

Next.JS, React, Styled JSX

Arquitectura

# Platzi Podcasts

App para escuchar Podcasts
Utiliza la API de Audioboom


6:55
Podcasts!
MASACOTE
Masacote con Chente Ydrach
QUÉMESE DESPUÉS DE ESCUCHAR
Quémese después de Escuchar
6AM
Hoy por hoy
Caracol Radio -6 am Hoy por Hoy
POSTA
POSTA
ONE NACIÓN
ESPN One Nación
MISSING
MAURA MURRAY
Missing Maura Murray
olver
274. Los Increíbles 2
Quémese después de Escuchar
00:00
-54:31

# Nuestra primera página

¿Cómo funciona Next?

# Package.json

Documentación del Proyecto
(Junto al Readme)

# Scripts Indispensables

**dev**: Entorno Desarrollo

**build** y **start**: Entorno Producción

# El Router

Cada path va al archivo del mismo nombre en la carpeta **/pages**

**GET /**

/pages/index.js

**GET /platzi**

/pages/platzi.js

```
export default () => (
    <h1>Hola!</h1>
)
```

# Styled JSX

El sistema de estilos de Next.JS

# ¿Por qué usar Styled JSX?

Es más acorde a React

Evitamos problemas al escalar

```
/* BEM */
.block__element--modifier {
    color: red;
}

/* Styled JSX */
.eso { color: red; }
```

# ¿Cómo funciona?

Escribimos CSS3 como siempre

Sólo aplica al componente

```
<style jsx>{`
  /* CSS */
  .clase { color: red; }
`}</style>
```

# Reglas de Styled JSX

Se aplica por componente

Tampoco aplica a componentes internos o externos

# ¿Cómo romper las reglas?

<style jsx **global**>

Operador **:global()**

```
<style jsx global>{`
  /* CSS Global */
  .clase { color: red; }
`}</style>
```

```
<style jsx>{`
      /* Operador :global() */
   :global(.clase) { color: red; }
`}</style>
```

```
<div className=”main”>
    <Navegacion />
</div>
```

---

```
// Qué es <Navegacion />?
<div><a>Home</a></div>
```

```
<style jsx>{`
    /* Escapando Scope */
  .main :global(a) { color: red; }
`}</style>
```

# ¿Y los archivos estáticos?

## Van en la carpeta /static

Se sirven automáticamente

# Reto: About

¡Crea una nueva página!

1. Crear página **/about**
2. Incluir una **imagen**
3. Estilarla con **Styled JSX**
4. Cambiar el fondo del **body**

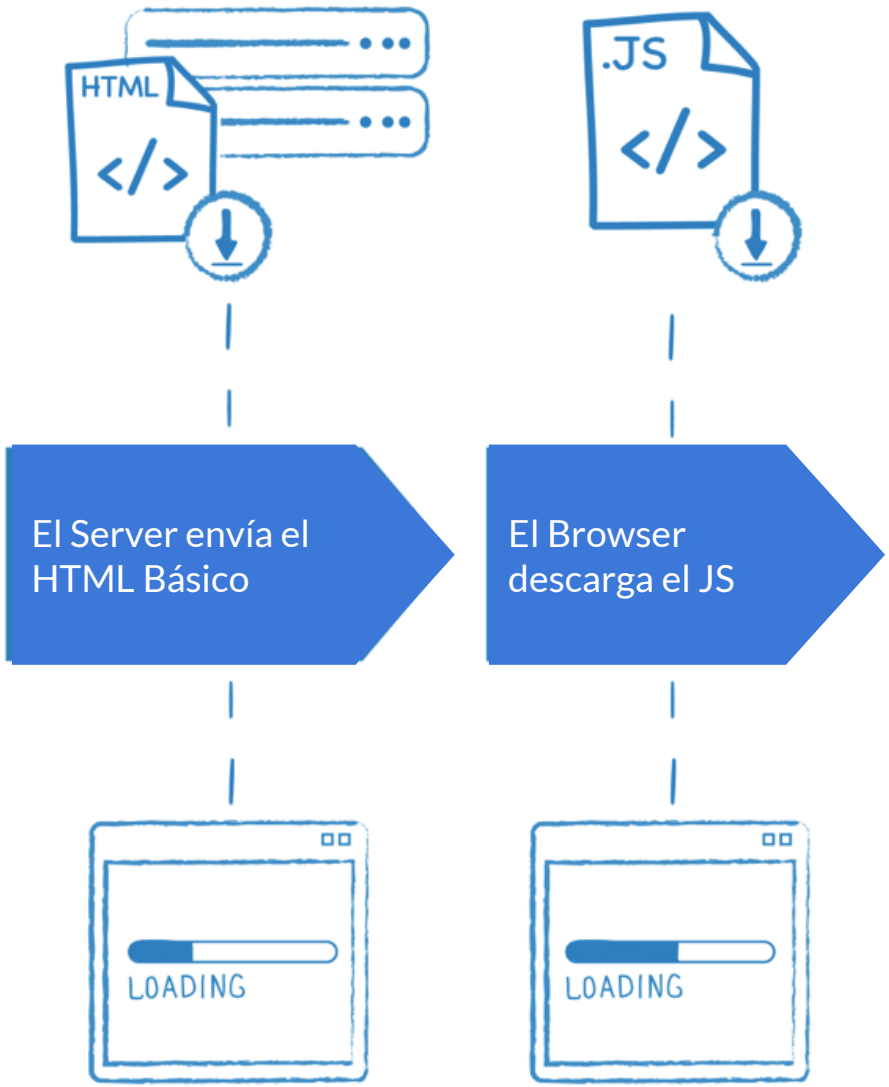
No se puede mostrar la imagen.

# Server Side Rendering

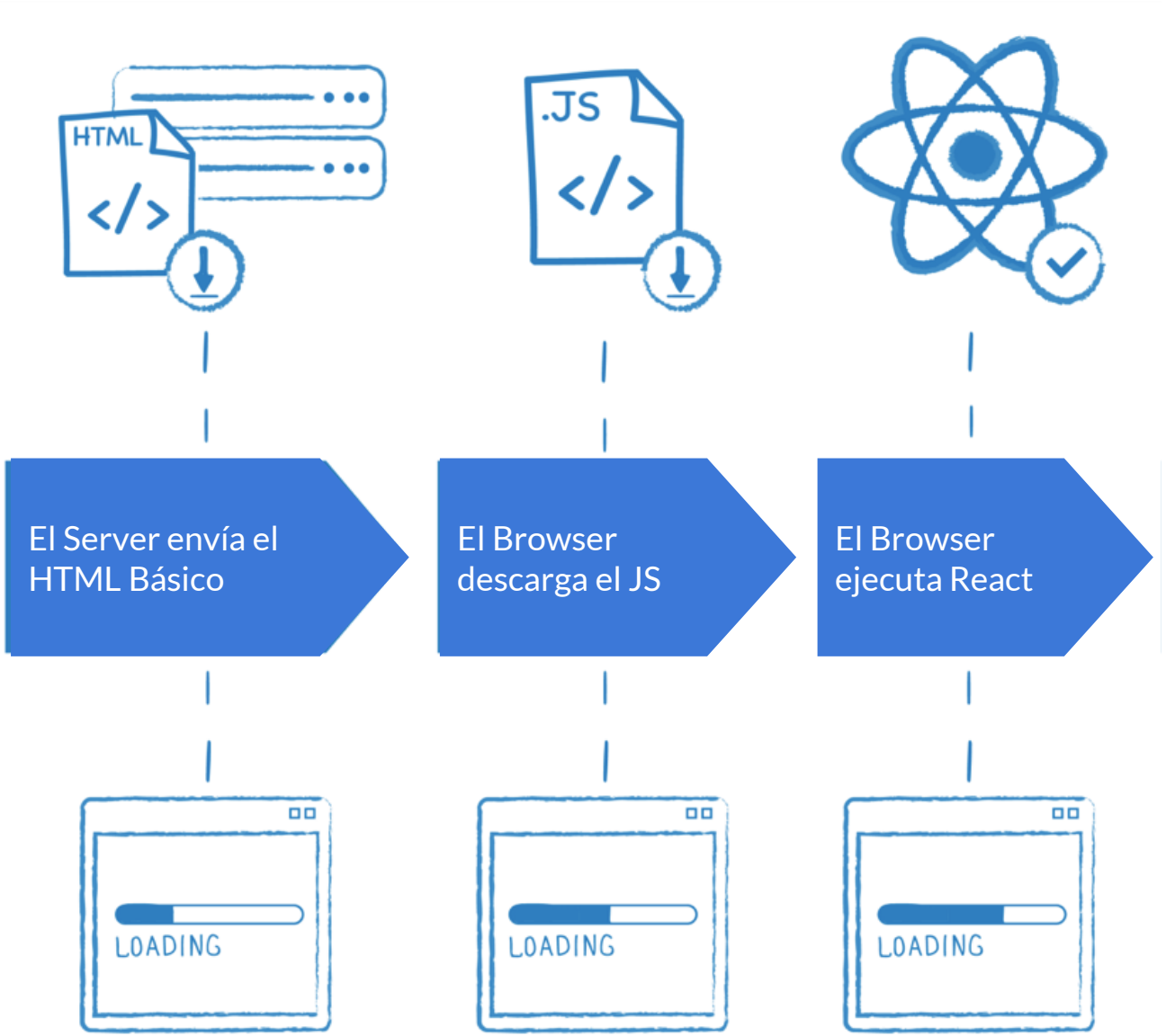
Mejora Performance y SEO
Automático en Next.JS!

# Client Side Rendering

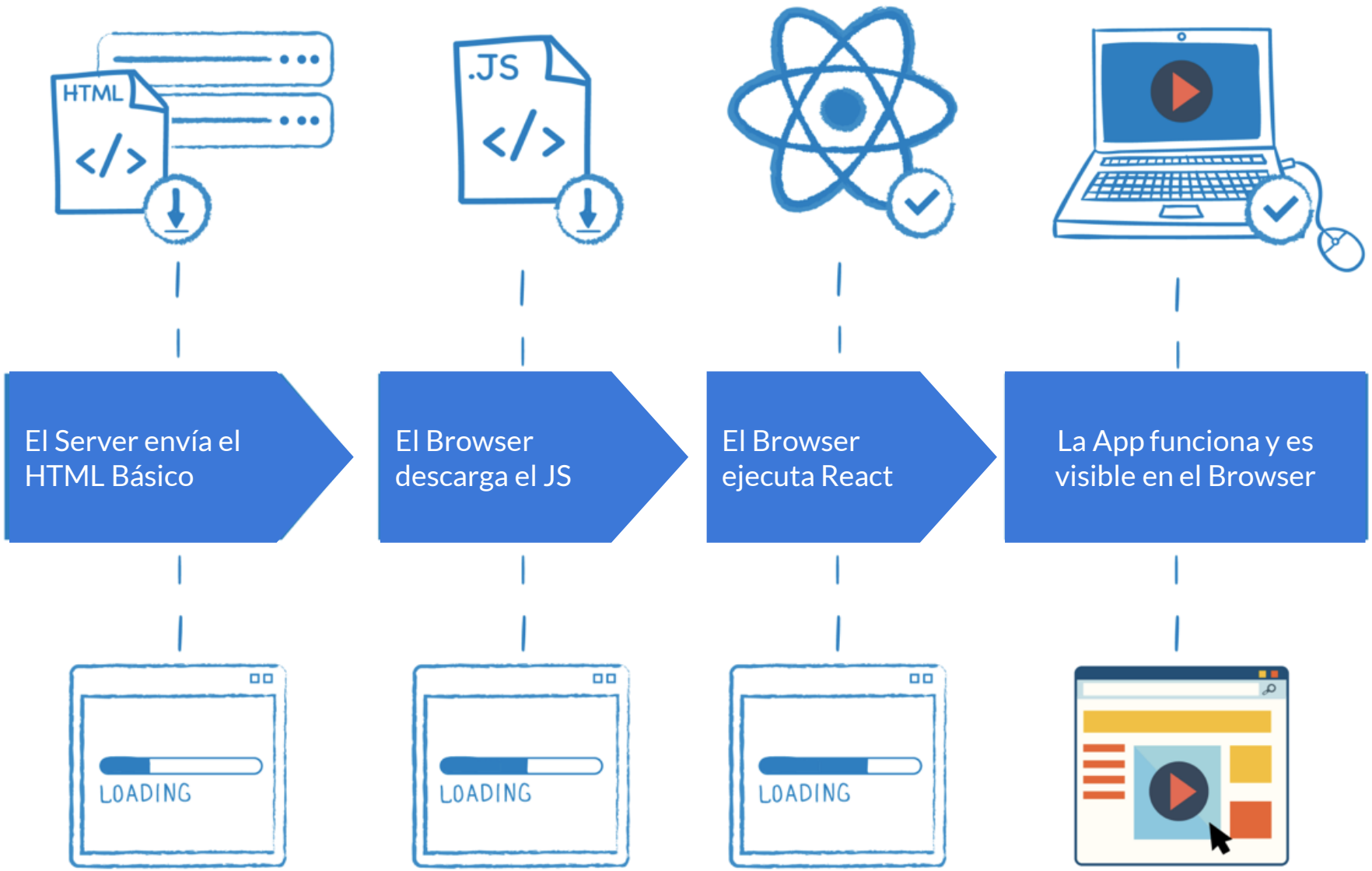
# Client Side Rendering



# Client Side Rendering

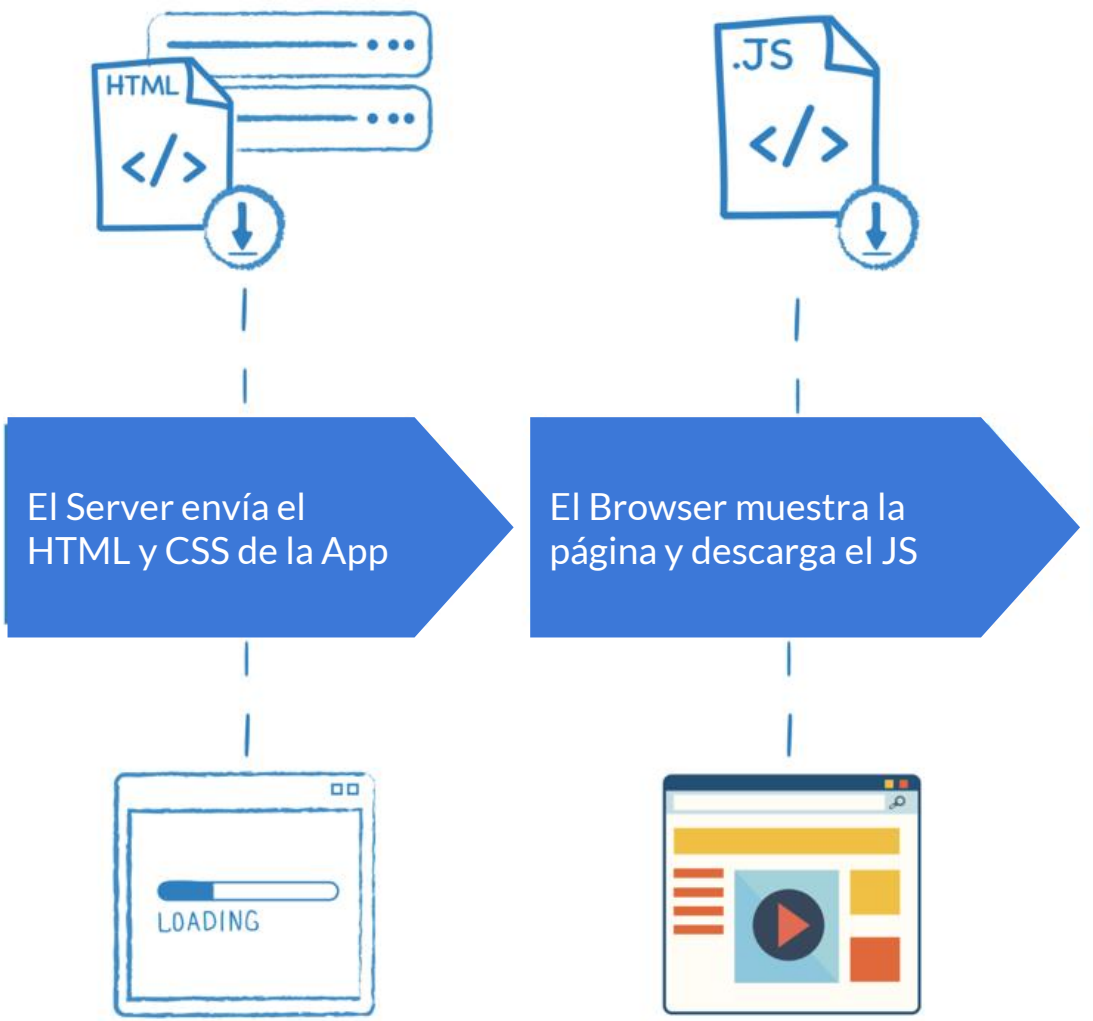


# Client Side Rendering

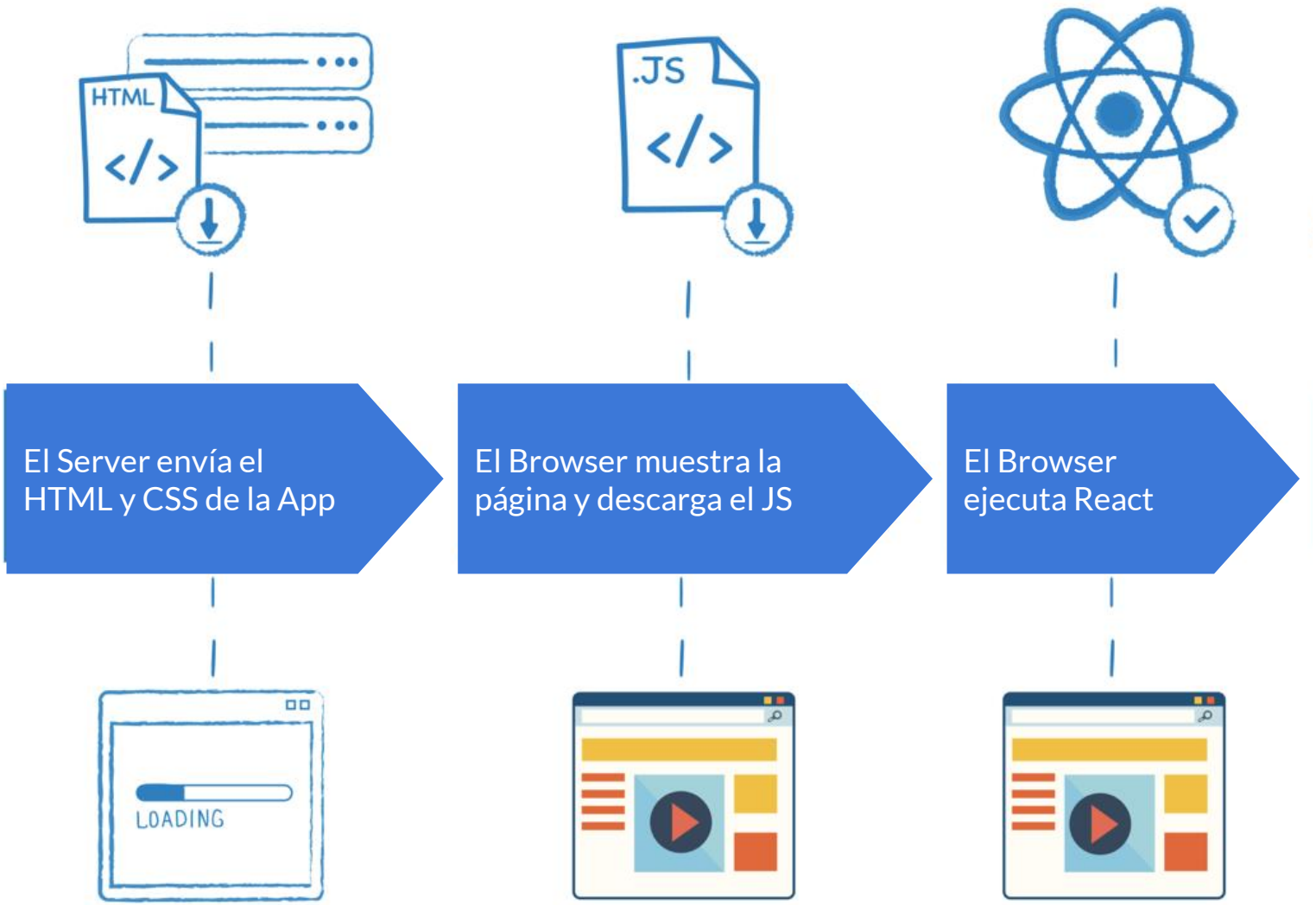


# Server Side Rendering

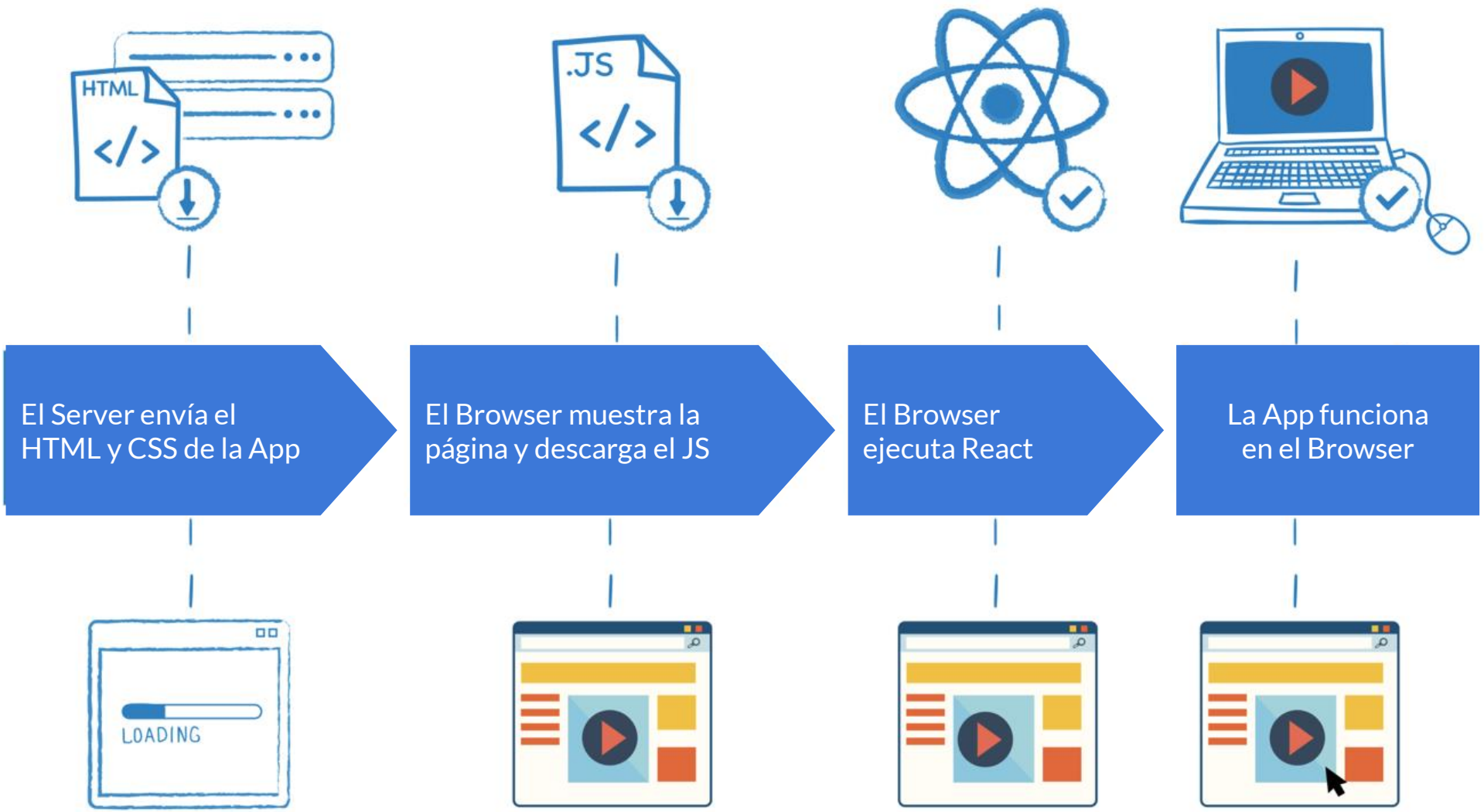
# Server Side Rendering



# Server Side Rendering



# Server Side Rendering



# ¿Qué ventajas tiene?

Mejor Performance y UX

Indexa en todos los servicios

# ¡En Next.JS es Automático!

Todas las páginas son
Renderizadas Server Side

# ¿Pero qué pasa si tengo que llamar un API?

¿Cómo se entera Next.JS que tenemos los datos?

# getInitialProps()

Específico de Next.JS

Cargar el Contenido Principal

```
static async getInitialProps() {
  // Aquí traemos los datos
  let request = await fetch('...')
  let datos = await request.json()
  return { datos }
}
```

# ¡Importante!

## getInitialProps()
## Sólo funciona en Pages

Audioboom API
**github.com/audioBoom/api**
**api.audioboom.com**

Channels Recomendados
**/channels/recommended**

# Protip: CSS Grids

Con CSS Grids podemos armar el layout en un componente

# Enlazando Páginas

Aprovechando una gran feature de Next.JS

# <Link />

Debe contener un <a>
u otro elemento

```
<Link href="/url">
  <a>Texto del Link</a>
</Link>
```

# ¿Cómo funciona <Link />?

Next.JS mezcla Client Side Rendering con Server Side Rendering

# Si clickeamos en un Link

## Client Side Rendering

Carga sólo lo que falta de esa página

Cambia la ruta en el browser

# Si abrimos un nuevo Tab

## Server Side Rendering

Carga todo el HTML, CSS y JS

Crea una nueva sesión

# ¿Cómo funciona prefetch?

Precarga sólo HTML, CSS y JS

**¡No precarga getInitialProps!**

```
<Link href=”/url” prefetch>
   <a>Texto del Link</a>
</Link>
```

# ¿Qué pasa si repito prefetch?

Cada página sólo se precarga una vez

Lo podemos usar en listas

# ¡Importante!

**Prefetch sólo funciona en producción**

(npm run build && npm start)

# Recibiendo Parámetros

¿Cómo implementamos la página de cada Channel?

```
static async getInitialProps({query}) {
    let id = query.id
    // Obtenemos los datos...
    return { datos }
}
```

# Protip: Performance

Si hay que hacer varias requests,
**paralelízalas con Promise.all()**

```
// Hacer 2 requests en paralelo
let [req1, req2] = await Promise.all([
    fetch(url_1),
    fetch(url_2)
])
```

# Reto: Vista de Podcasts

Hora de aplicar todo lo que aprendimos

1. Crear página **/podcast**
2. Tomar el **id de la query**
3. Pedir los datos al **API**
4. Mostrar el **<audio />**
5. Linkearlo desde **/channel**

5:01

localhost

< Volver

## 274. Los Increíbles 2

**Quémese después de Escuchar**

--:--

# Mejorando el Código

Haciendo que nuestra app sea más mantenible con React

# Protip: Primero Papel

Bocetemos el código como nos gustaría implementarlo

# Crear un Layout

Reutilizamos la navegación

Usamos la prop children de React para usarlo como contenedor

# <Head />

Permite sumar cosas al head

Ideal para <title> y <meta>

```
<Head>
  <title>Podcasts</title>
</Head>
```

# Crear Componentes

Identificar Código Duplicado
Separarlo en Componentes
Definir una API por componente

# Componentes Simples

Listas de Objetos

Componentes con Lógica

# Reto: Reorganizar la App

Aplicar lo que aprendimos al resto de la app

1. Aplicar **<Layout />** con titles
2. Crear **<PodcastList />**
3. Refactorear **/channel**
4. Refactorear **/podcast**

# Manejando Errores

¿Qué pasa si se rompe algo?

# Status 200

¡Está todo bien!
La página funciona

# Error 404

Lo aplicamos si la página no existe

**Sólo si no existe**

# ¡Cuidado!

Devolver errores 404 por accidente puede causar problemas de SEO

# Error 503

Lo aplicamos si hay problemas de red o la API no está funcionando

# ¡Importante!

Siempre hay que cambiar el **res.statusCode**

# ¡Importante!

## Nunca redirigir a una página de /404 o /503.

Los errores son parte de la página

# Personalizando Errores

Unificando el diseño de los errores de nuestra aplicación

# Personalizar <Error />

Utiliza la página **_error.js**

# // https://github.com/zeit/next.js

```
export default class Error extends React.Component {
    static getInitialProps({ res, err }) {
        const statusCode = res ? res.statusCode :
            err ? err.statusCode : null;
        return { statusCode }
    }
    render() {
        return <p>{this.props.statusCode ?
                    `Error ${this.props.statusCode} en el server` :
                    `Error en el browser`}</p>
    }
}
```

# Personalizar <Document />

Extender Server Side Rendering

**¡Sólo si es necesario!**

# ¿Cuándo modificarlo?

Google AMP

Facebook Instant Pages

Plugins como Styled Components

# Diseñando URLs

Hagamos URLs
User Friendly

# Legibilidad

Deben ser entendibles por nuestros usuarios

// Esto no es legible
/channel?id=4702115

// Esto sí
/posta

# Consistencia

Deberíamos poder borrar cualquier fragmento

// Todas deberían andar

**/podcast/un-buen-dia**

**/channel/posta**

**/channel**

**/**

// Todas deberían andar

**/podcast/un-buen-dia**

**/channel/posta**

**/channel**

**/**

// Nuestra estructura de URLs

**/posta/un-buen-dia**

**/posta**

**/**

# Next Routes

Named URLs para Next.JS

Requiere un server.js custom

// Nuestra estructura de URLs

**/posta/un-buen-dia**

**/posta**

**/**

**Ir de esto ...**

```
<Link href={`/canal?id=${id}`}>
  <a>Canal</a>
</Link>
```

**...a esto**

```
<Link route="canal"
 params={{ slug: "canal", id: 123 }}>
 <a>Canal</a>
</Link>
```

# Implementando Next Routes

## Modificando los Links de nuestra aplicación

# Reto: Navegación

Aplicar lo que aprendimos al resto de la aplicación

1. Implementar **Next Routes**
2. Aplicar errores **404** y **503**
3. Personalizar errores

# Vistas Híbridas

Implementando un modal
para tener un link instantáneo

# this.setState()

Cambia el estado del componente

Por ejemplo, para mostrar un modal

# Agregando un loader

Router.onRouteChangeStart

Router.onRouteChangeComplete

# Code!

# Publicando en Github

Algunos consejos y tips
para mejorar tu portfolio

# ¡El Readme!

Incluir información del proyecto

Cómo levantar entornos

Quién lo hizo

# ¡A shipear!

Pongamos el sitio en producción

# now

Nos permite publicar nuestra aplicación con un solo comando

# Protip: Docker

## Estandarizar Deployments

## Curso Fundamentos de Docker

# Documentación Oficial

github.com/zeit/next.js

¡Ver los ejemplos!

# Aprender React

Curso de React en Platzi

reactjs.org

# Docker y Deployments

Curso de Docker en Platzi

zeit.co/now