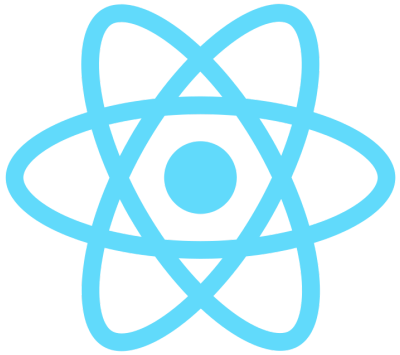
React



Introducción a React

Introducción

Introducción

- Librería JS

Introducción

- Librería JS
- UI reactivas

Introducción

- Librería JS
- UI reactivas
- SPA: Single Page Applications

Introducción

- Librería JS
- UI reactivas
- SPA: Single Page Applications
- Data binding

Introducción

- Librería JS
- UI reactivas
- SPA: Single Page Applications
- Data binding
- Web components

Introducción

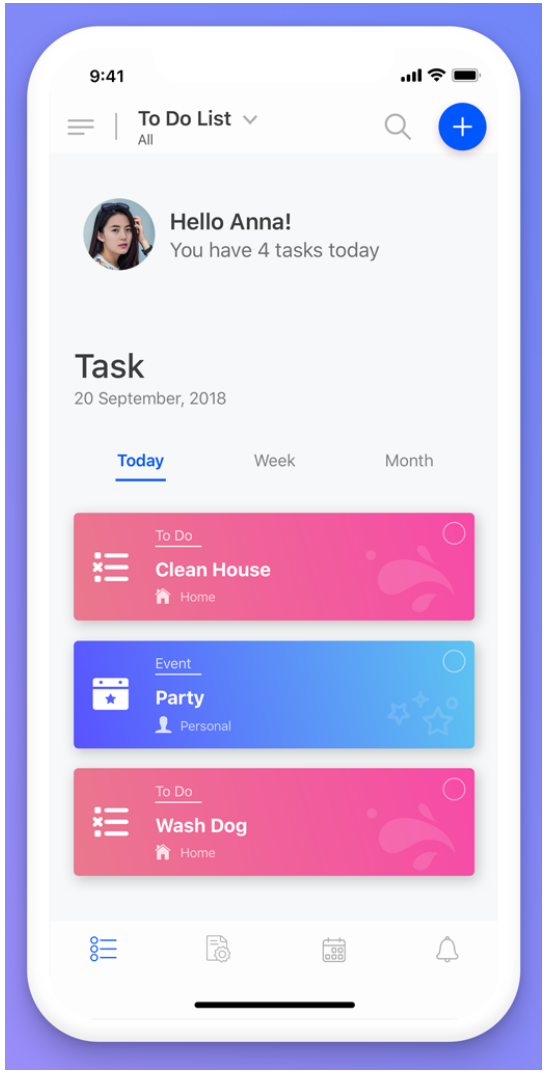
- Librería JS
- UI reactivas
- SPA: Single Page Applications
- Data binding
- Web components
- Microservicios / Jamstack

Introducción

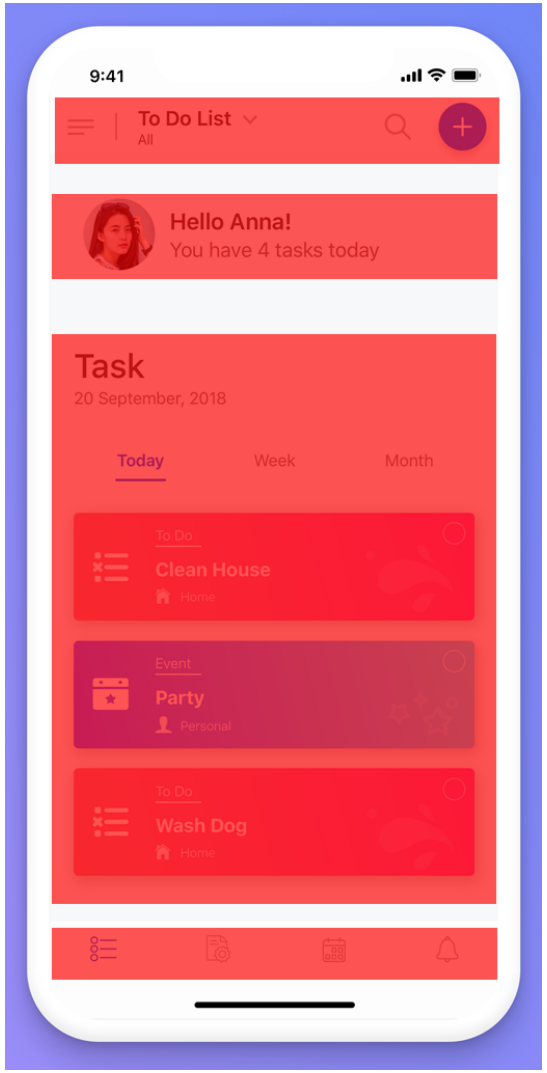
- Librería JS
- UI reactivas
- SPA: Single Page Applications
- Data binding
- Web components
- Microservicios / Jamstack
- Código fuente y código compilado
- JSX

Componentes

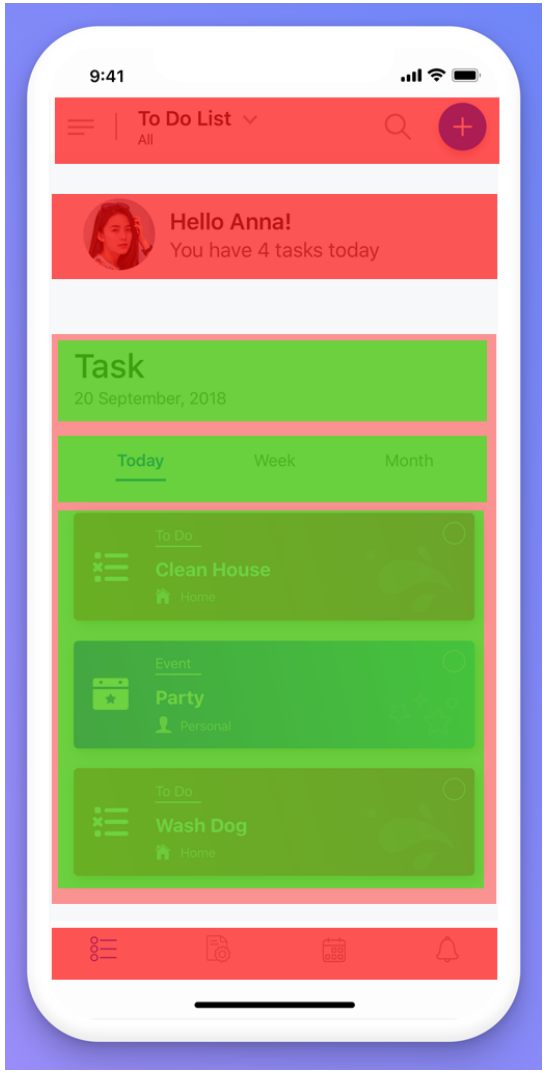
Componentes



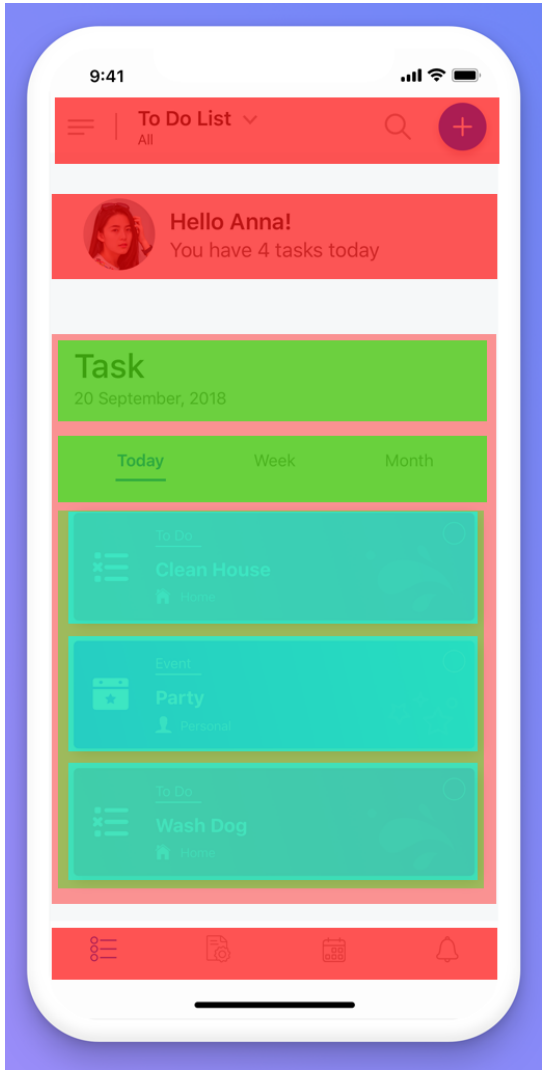
Componentes



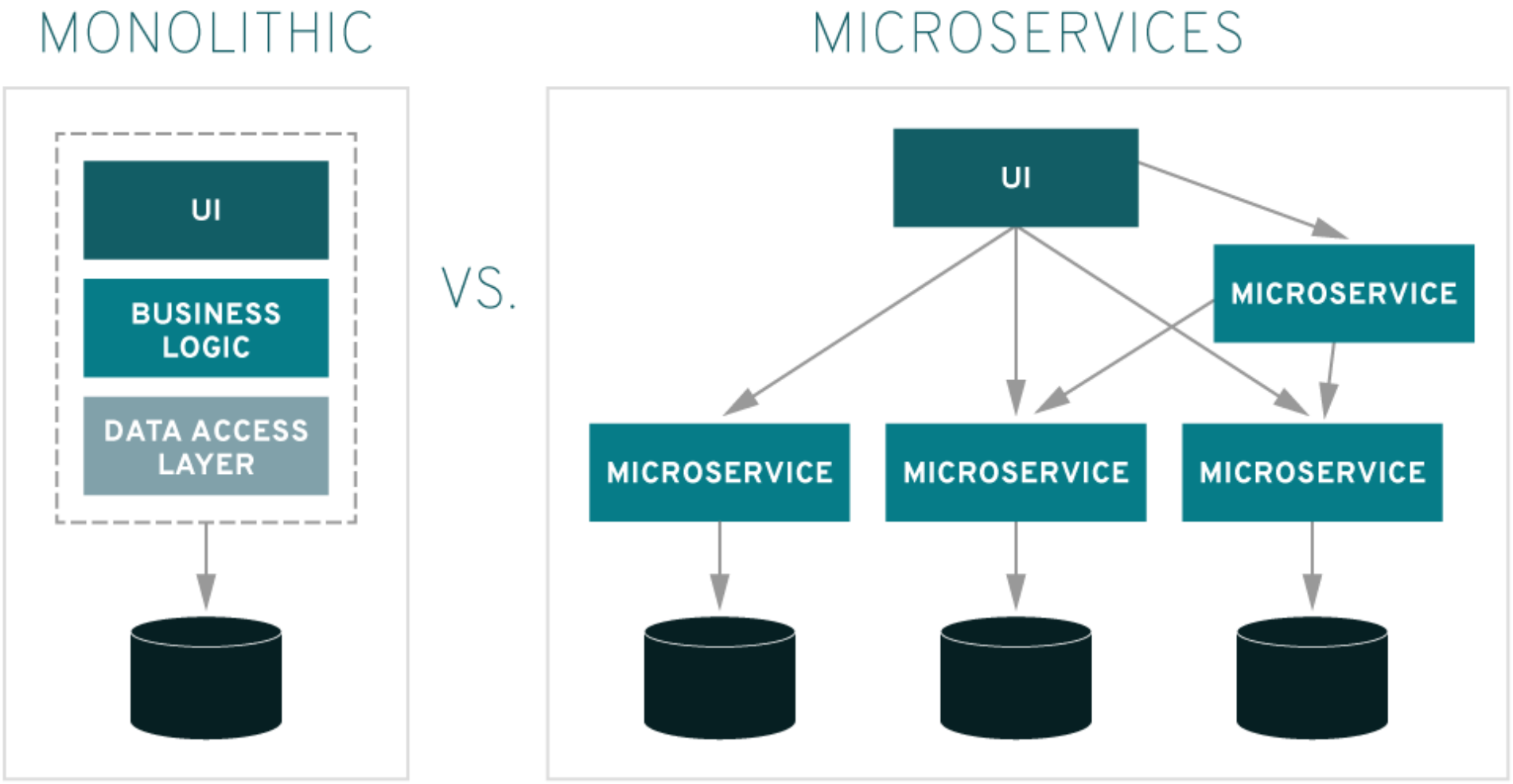
Componentes



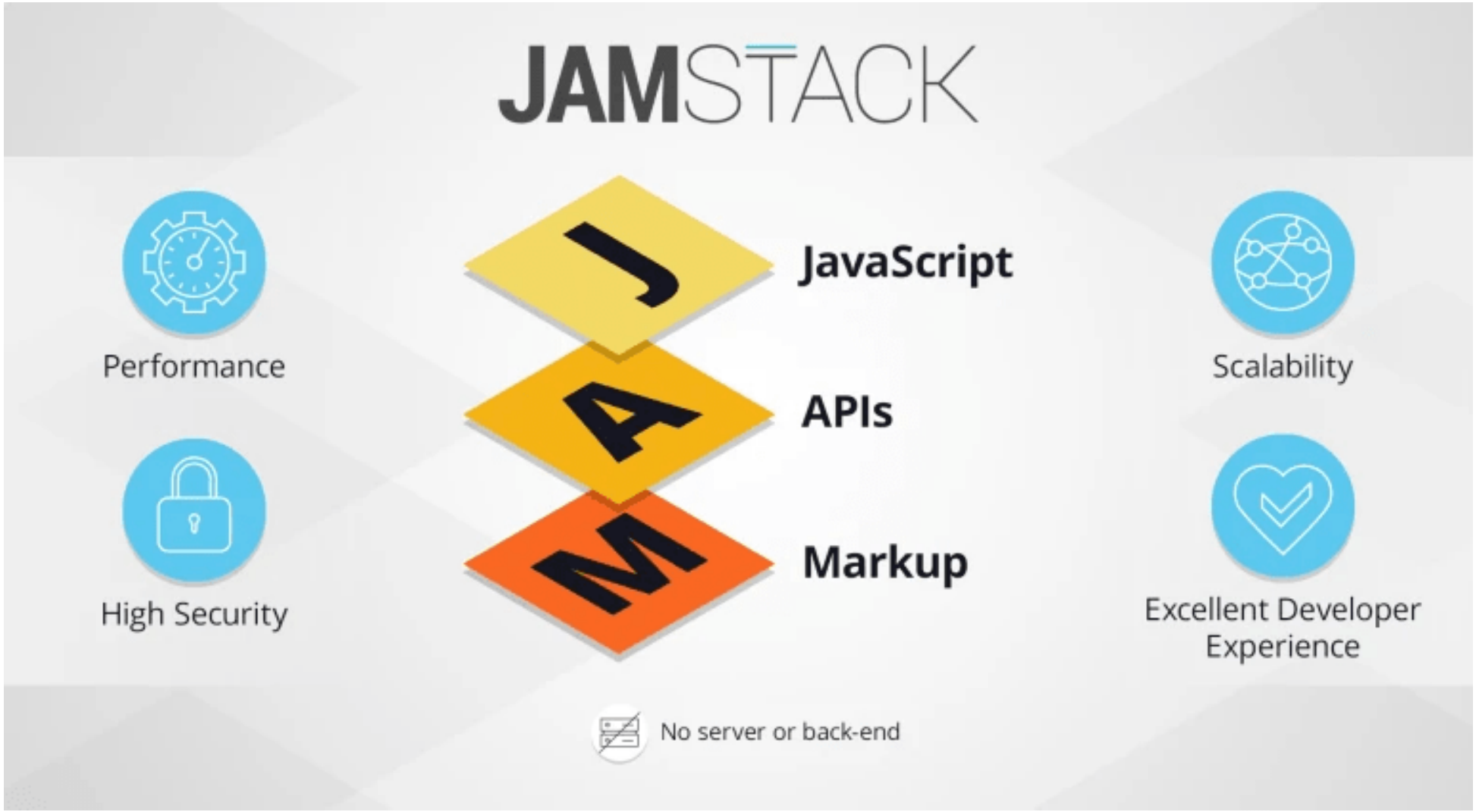
Componentes



Microservicios

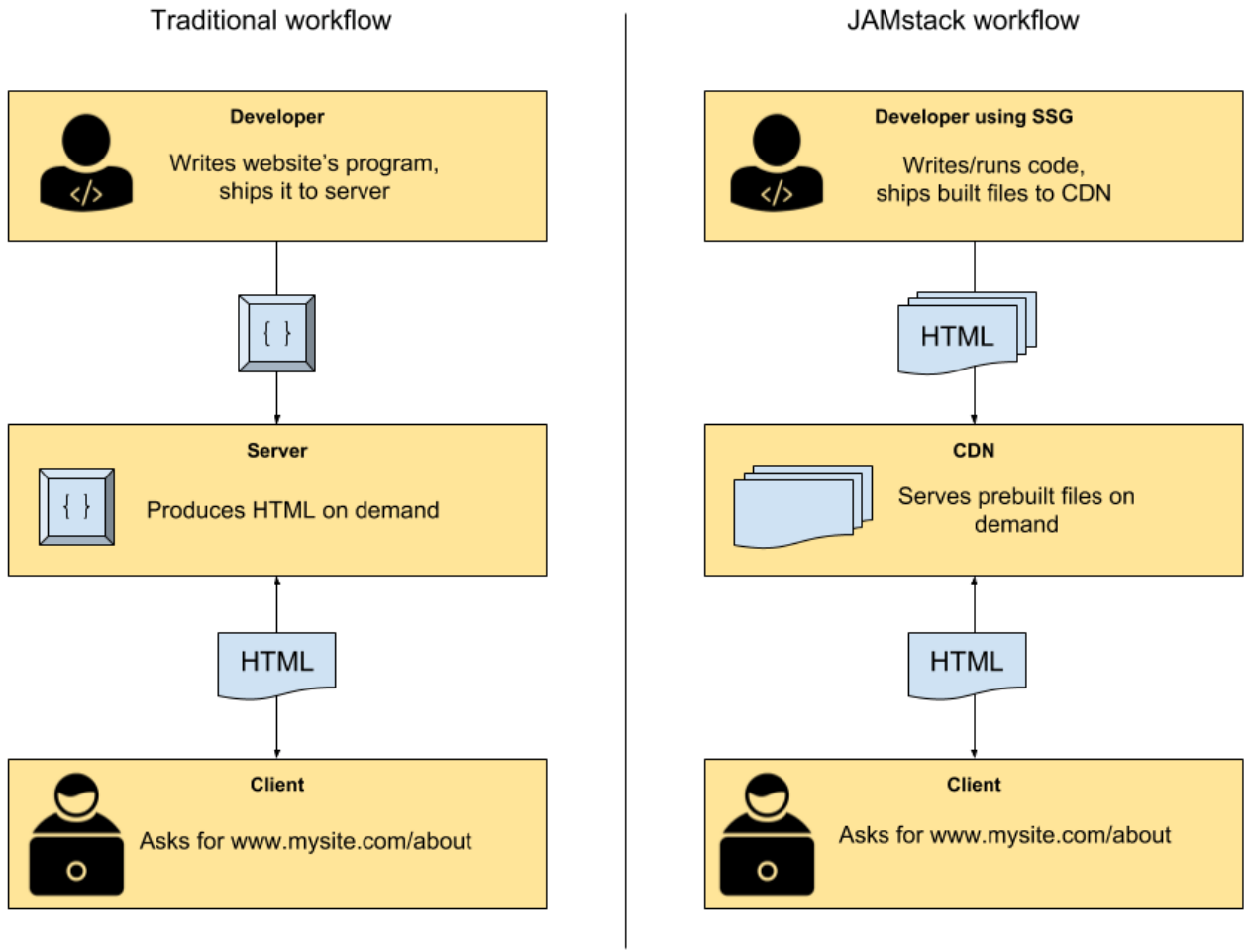


Jamstack



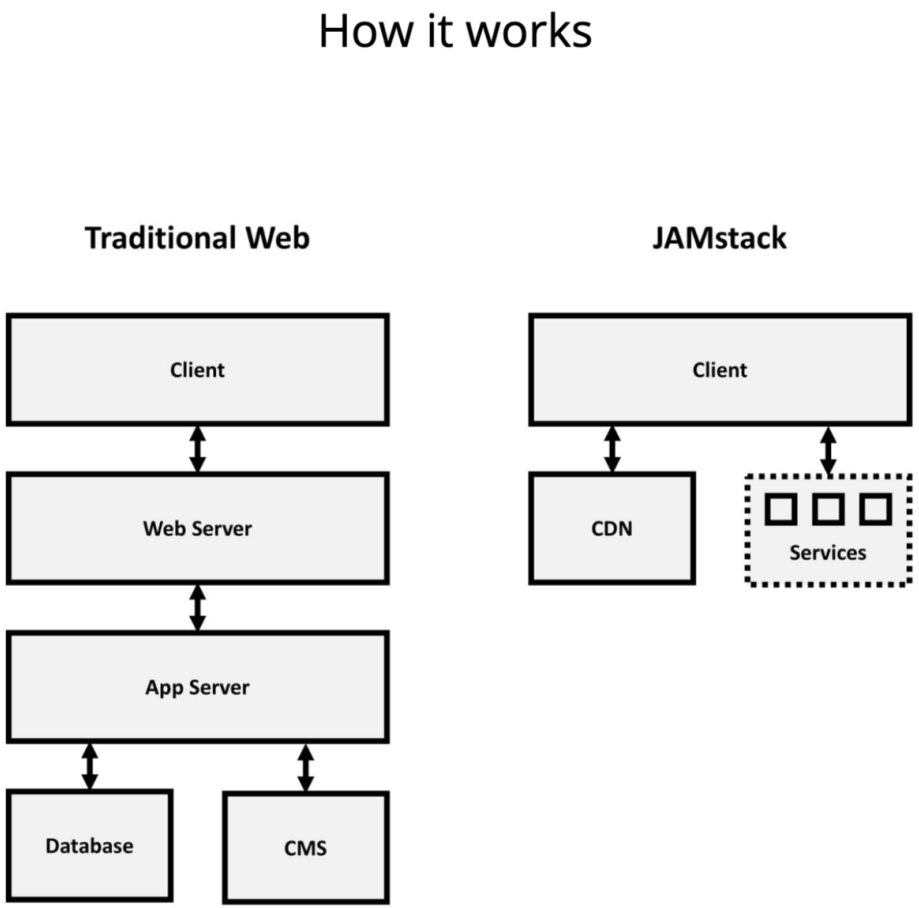
Fuente: https://cloudinary.com/blog/developer_experience_for_a_modern_web_jamstack_delivers

Jamstack



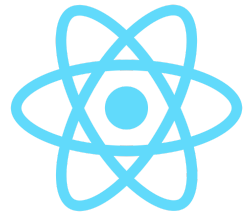
Fuente: https://dinarys.com/blog/jamstack-for-modern-web-development

Jamstack



Fuente: https://dinarys.com/blog/jamstack-for-modern-web-development

Entorno de desarrollo



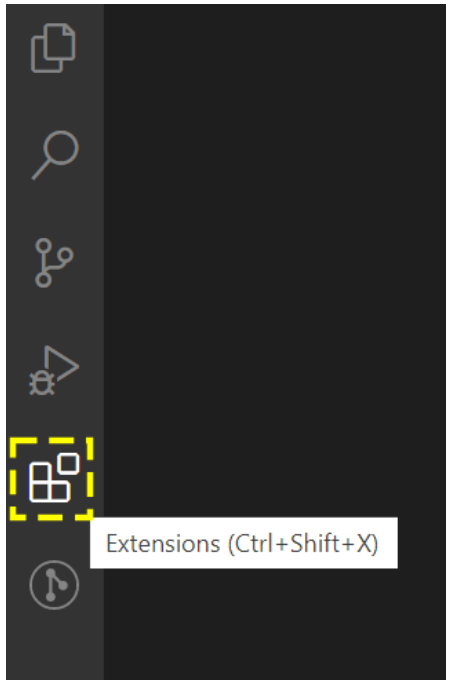
Entorno de desarrollo

Entorno de desarrollo

- IDE: Visual Studio Code

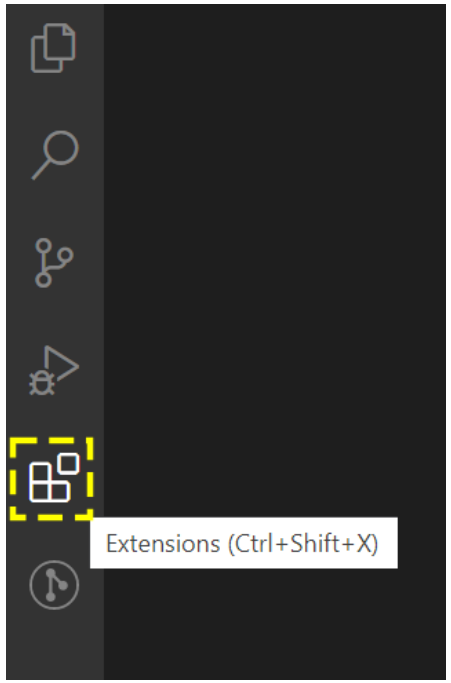
Entorno de desarrollo

- IDE: Visual Studio Code
  - Extensiones:
    - ESLint
    - Prettier
    - Jest



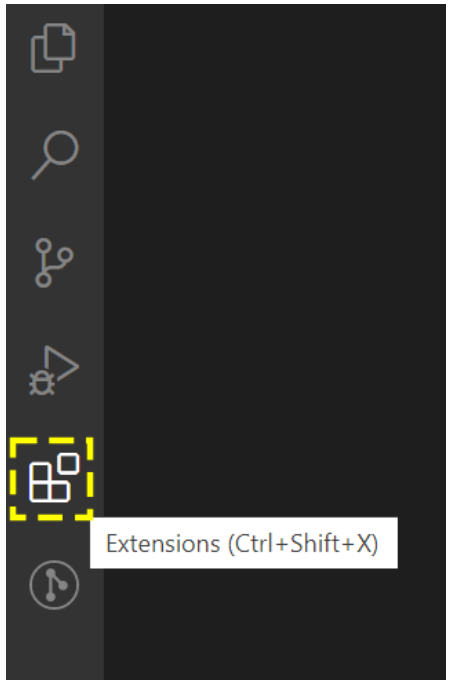
Entorno de desarrollo

- IDE: Visual Studio Code
  - Extensiones:
    - ESLint
    - Prettier
    - Jest
  - Configuración:



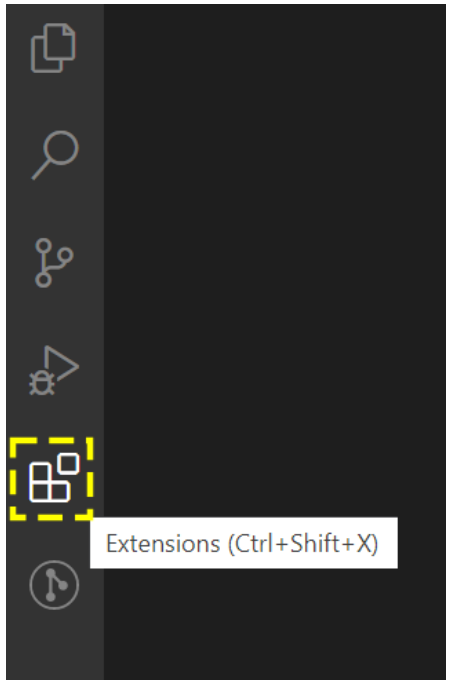
Entorno de desarrollo

- IDE: Visual Studio Code
  - Extensiones:
    - ESLint
    - Prettier
    - Jest
  - Configuración:
    - CTRL-MAYS-P: Preferences: Open settings



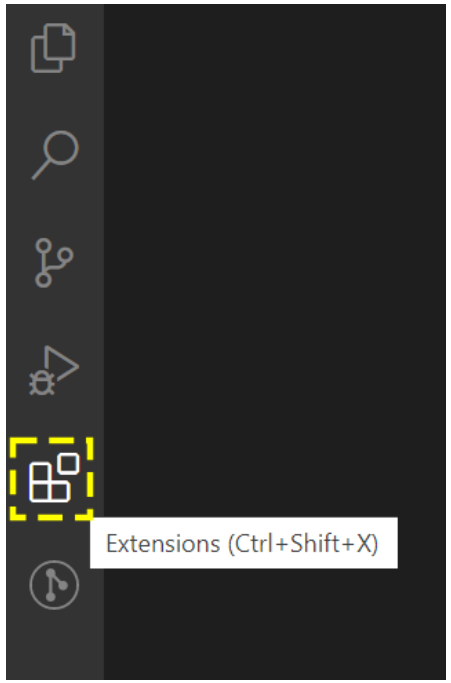
Entorno de desarrollo

- IDE: Visual Studio Code
  - Extensiones:
    - ESLint
    - Prettier
    - Jest
  - Configuración:
    - CTRL-MAYS-P: Preferences: Open settings




```
"editor.formatOnSave": true,
"editor.formatOnPaste": true,
"editor.codeActionsOnSave": {
  "source.fixAll.eslint": true
},
"editor.defaultFormatter": "esbenp.prettier-vscode"
```

Entorno de desarrollo



- IDE: Visual Studio Code
  - Extensiones:
    - ESLint
    - Prettier
    - Jest
  - Configuración:
    - CTRL-MAYS-P: Preferences: Open settings

```
"editor.formatOnSave": true,
"editor.formatOnPaste": true,
"editor.codeActionsOnSave": {
  "source.fixAll.eslint": true
},
"editor.defaultFormatter": "esbenp.prettier-vscode"
```

- Extensión para navegador: React Developer Tools

Entorno de desarrollo

Entorno de desarrollo

- Node.js y npm

Node.js y npm

npm

npm

- Repositorio de paquetes distribuibles

npm

- Repositorio de paquetes distribuibles
- La carpeta node_modules

npm

- Repositorio de paquetes distribuibles
- La carpeta node_modules
- El archivo package.json

Comandos npm

Comandos npm

- Instalar un paquete:
  npm install paquete [--save-dev]

Comandos npm

- Instalar un paquete:
  npm install paquete [--save-dev]
- Instalar todas las dependencias registradas:
  npm install

Comandos npm

- Instalar un paquete:
  npm install paquete [--save-dev]
- Instalar todas las dependencias registradas:
  npm install
- Instalar sólo las dependencias de producción:
  npm install --production

Comandos npm

- Instalar un paquete:
  npm install paquete [--save-dev]
- Instalar todas las dependencias registradas:
  npm install
- Instalar sólo las dependencias de producción:
  npm install --production
- Ejecutar un paquete:
  npx ejecutable

Configuración proxy

npm config set proxy http://username:password@host:port

npm config set https-proxy http://username:password@host:port

Introducción a JavaScript

Conceptos iniciales

Conceptos iniciales

- Interpretado, compilado y ejecutado en el navegador

Conceptos iniciales

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS

Conceptos iniciales

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**

Conceptos iniciales

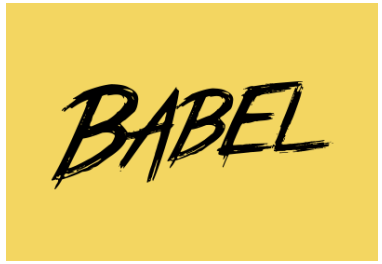
- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**
- La versión **ES6** o **ES2015**

Conceptos iniciales

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**
- La versión **ES6** o **ES2015**
- Transpiladores: Babel, TypeScript

Conceptos iniciales

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**
- La versión **ES6** o **ES2015**
- Transpiladores: Babel, TypeScript



Conceptos iniciales

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**
- La versión **ES6** o **ES2015**
- Transpiladores: Babel, TypeScript
- Módulos:

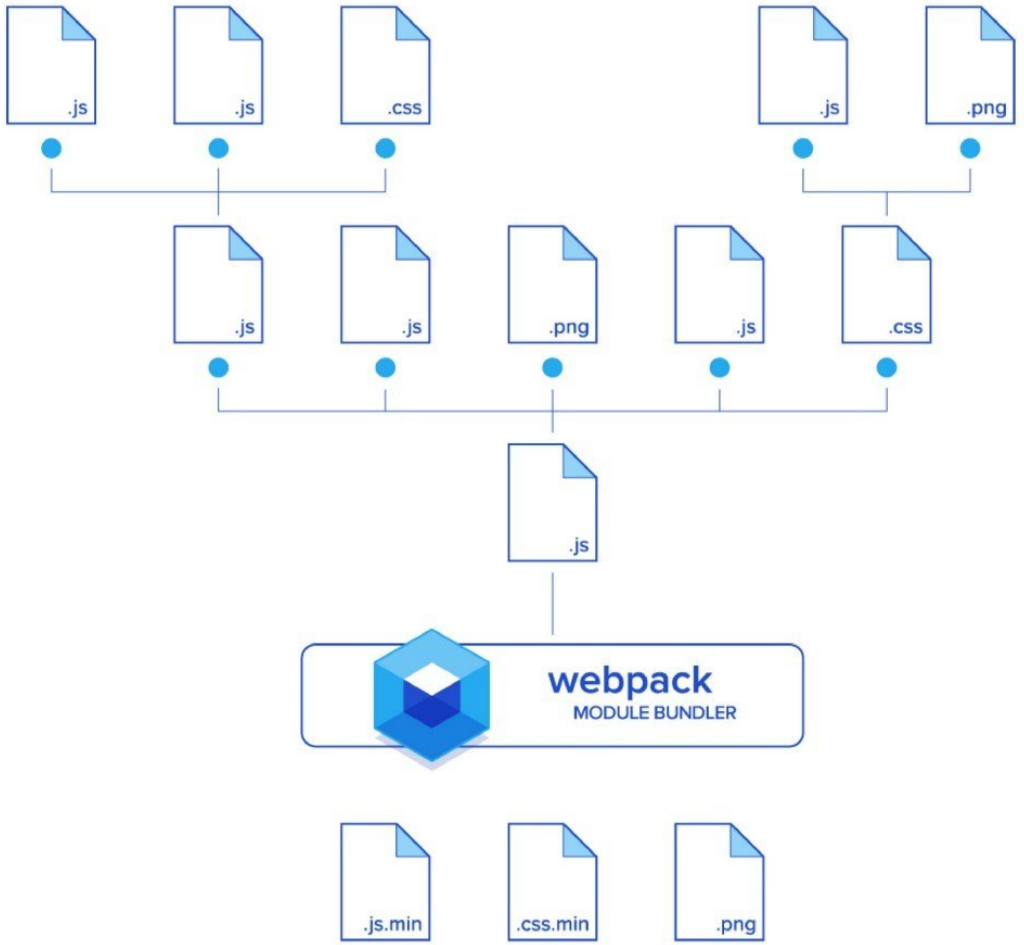
Conceptos iniciales

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**
- La versión **ES6** o **ES2015**
- Transpiladores: Babel, TypeScript
- Módulos:
  - import / export

Conceptos iniciales

- Interpretado, compilado y ejecutado en el navegador
- Cada navegador programa su propio motor de JS
- Estandarización: **ECMAScript**
- La versión **ES6** o **ES2015**
- Transpiladores: Babel, TypeScript
- Módulos:
  - import / export
  - webpack

Conceptos iniciales



Fuente: https://ana.tap.company/how-to-build-an-npm-package/

ES6

ES6

- let y const

ES6

- let y const

```
let a = 3;

let a = 10;  // Error
var a = 12;  // Error

const b = 10;

b = 3; // Error

const obj = {
    x: 10,
    y: 12
}

obj.x = 15;  // OK

obj = {    // Error
    x: 15,
    y: 12
}
```

ES6

- let y const

ES6

- let y const
- Template literals

```
let nombre = "Antonio";

let cuadrado = function(x) {
    return x * x;
}

let n = Math.floor(Math.random() * 10);

let saludo1 = "Hola, " + nombre + ". El cuadrado de " + n + " es " + cuadrado(n) + ".";
let saludo2 = `Hola, ${nombre}. El cuadrado de ${n} es ${cuadrado(n)}.`;
```

ES6

- let y const
- Template literals

ES6

- let y const
- Template literals
- for ... of

ES6

```
let nombres = ["Patricia", "Zacarías", "Miguel", "Maite"];

for (let i in nombres) {
    console.log(nombres[i]);
}

for (let nombre of nombres) {
    console.log(nombre);
}

let obj = {
    x: 3,
    y: 4
}

for (let i in obj) {
    console.log(obj[i]);
}

let nombre = "Antonio Jesús";

for (let c of nombre) {
    console.log(c);
}
```

ES6

- let y const
- Template literals
- for ... of

ES6

- let y const
- Template literals
- for ... of
- Funciones
  - Parámetros por defecto

ES6

```
function potencia(x, y = 2) {
    return Math.pow(x, y);
}

console.log(`10 elevado a 8 es ${potencia(10, 8)}`);
console.log(`El cuadrado de 5 es ${potencia(5)}`);
```

ES6

- let y const
- Template literals
- for ... of
- Funciones
  - Parámetros por defecto

ES6

- let y const
- Template literals
- for ... of
- Funciones
  - Parámetros por defecto
  - Función arrow:

    (parámetros) => expresión_devuelta

ES6

```
const potencia = function (x, y = 2) {
    return Math.pow(x, y);
}

const potencia = (x, y = 2) => Math.pow(x, y);

setTimeout(() => console.log("pausa"), 2000);
```

ES6

ES6

- Operador spread

ES6

- Operador spread
  - Parámetros en funciones

ES6

- Operador spread
  - Parámetros en funciones
  - Copiar un array en otro

ES6

- Operador spread
  - Parámetros en funciones
  - Copiar un array en otro
  - push inmutable

ES6

- Operador spread
  - Parámetros en funciones
  - Copiar un array en otro
  - push inmutable
  - Copiar un objeto en otro

ES6

- Operador spread
  - Parámetros en funciones
  - Copiar un array en otro
  - push inmutable
  - Copiar un objeto en otro
  - Mergear objetos

ES6

```
// function(a, b, c)
function sumar(a, b, c) {
  console.log(a + b + c);
}
let nums = [1, 3, 6];
sumar(...nums);

// function(n parámetros)
function sumar(...nums) {
  let suma = 0;
  for (n of nums) {
    suma += n;
  }
  console.log("La suma es " + suma);
}
let a = 3;
let b = 7;
let c = 8;
sumar(a, b, c);

// push y unshift
```

Métodos de los arrays

- Métodos:
  - map

Métodos de los arrays

```
let nombres = ["juan", "luisa", "amparo", "arturo"];

let nombresMays = nombres.map(nombre => nombre.toUpperCase());

console.log(nombresMays);
```

Métodos de los arrays

- Métodos:
  - map
  - filter

Métodos de los arrays

```
let personas = [
    {
        nombre: "juan",
        edad: 15
    },
    {
        nombre: "luisa",
        edad: 35
    },
    {
        nombre: "amparo",
        edad: 17
    },
    {
        nombre: "arturo",
        edad: 32
    }
];

let mayoresEdad = personas.filter(persona => persona.edad >= 18);

console.log(mayoresEdad);
```

Métodos de los arrays

- Métodos:
  - map
  - filter
  - reduce

Métodos de los arrays

```
let nums = [2, 4, 10, 15, 12];

let suma = nums.reduce((x, y) => x + y);

let objs = [
    {
        x: 3,
        y: 2
    },
    {
        x: 8,
        y: 10
    },
    {
        x: 10,
        y: 15
    }
]

let sumaX = objs.reduce((acc, obj) => acc + obj.x, 0);            // Mé
let sumaX = objs.map(obj => obj.x).reduce((obj1, obj2) => obj1 + obj2);
```

Desestructuración

Desestructuración

- Asignar desde un array

Desestructuración

```
function medidasMueble(mueble) {
  // ...

  return [100, 70, 20];
}

let [ancho, alto, profundo] = medidasMueble(mesa);

console.log(ancho, alto, profundo);

// 100, 70, 20
```

Desestructuración

- Asignar desde un array
- Intercambiar variables

Desestructuración

```
let a = 10;
let b = 20;

[a, b] = [b, a];

console.log(a, b);

// 20, 10
```

Desestructuración

- Asignar desde un array
- Intercambiar variables

Desestructuración

- Asignar desde un array
- Intercambiar variables
- Asignar desde un objeto

Desestructuración

```
1  function getRGB(colorHex) {
2    // ...
3
4    return {
5      alias: 'deeppink',
6      red: 255,
7      green: 20,
8      blue: 147,
9      alpha: 0.8
10   }
11 }
12
13 let { red, green, blue } = getRGB("#ff1493");
14
15 console.log(red, green, blue);
16
17 // 255, 20, 147
```

Desestructuración

```
let personas = [{
  nombre: "Luis",
  apellido: "Herrera",
  edad: 23
},
{
  nombre: "Marta",
  apellido: "Nieto",
  edad: 29
}];

for (let {nombre, edad} of personas) {
  console.log(`Me llamo ${nombre} y tengo ${edad} años
}

// Me llamo Luis y tengo 23 años
// Me llamo Marta y tengo 29 años
```

Desestructuración

```
let persona = {
  nombre: "Luis",
  edad: 23
}

let { nombre, edad, estado = "soltero" } = persona;

console.log(nombre, edad, estado);

// Luis, 23, soltero
```

Desestructuración

```
let notas = {
  mat: 8,
  fis: 6,
  dib: 5,
  tec: 6
}

let { mat: matematicas, fis: fisica, dib: dibujo, tec: tecnologia } = notas;

console.log(matematicas, fisica, dibujo, tecnologia);

// 8, 6, 5, 6
```

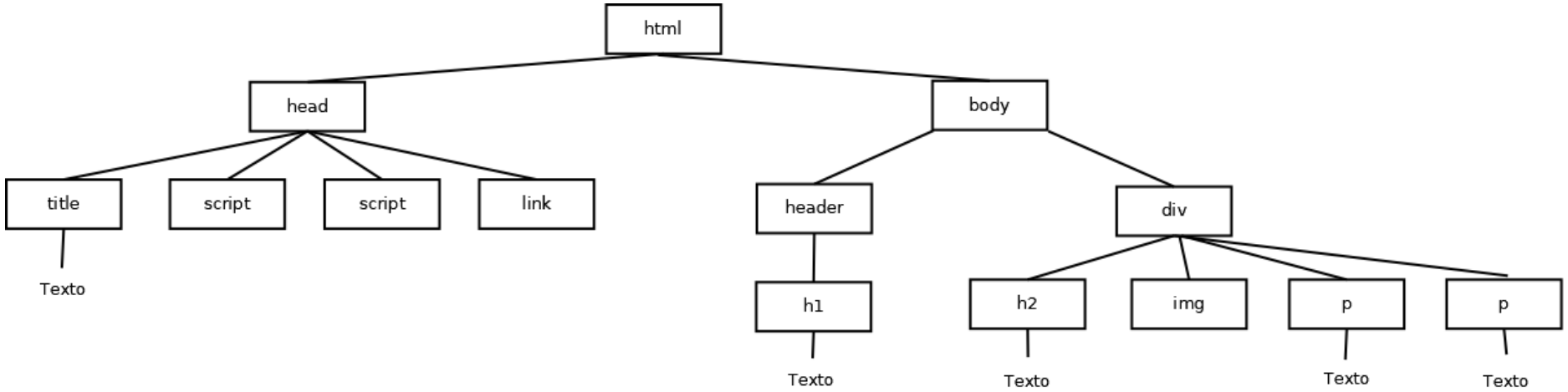
Manipulación del DOM

Manipulación del DOM

- El Document Object Model

Manipulación del DOM

- El Document Object Model



```
<!DOCTYPE html>
<html lang="es-ES">

<head>
  <title>DOM</title>
  <script src="js/jquery.min.js"></script>
  <script src="js/scripts.js"></script>
  <link rel="stylesheet" href="css/estilos.css">
</head>

<body>
  <header>
    <h1>Ejemplo DOM</h1>
  </header>
  <div>
    <h2>Esto es un documento HTML</h2>
    <img src="img/html.jpg" alt="Documento HTML">
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Non consectetur sapiente esse odit dolorem fugiat, iure
      omnis dolore obcaecati veniam necessitatibus sit quia praesentium, nesciunt suscipit. Adipisci illum tempore fuga.
    </p>
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Omnis, consequatur? Nostrum repudiandae qui expedita
      optio exercitationem sapiente rem, quisquam ipsam, sunt veniam dolorum architecto similique! Iusto quibusdam omnis
      tenetur nihil.</p>
  </div>
</body>

</html>
```

Manipulación del DOM

Manipulación del DOM

- El Document Object Model

Manipulación del DOM

- El Document Object Model
- Nodos HTML

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:
  - Genéricas:

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:
  - Genéricas:
    - classList

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:
  - Genéricas:
    - classList
    - className

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:
  - Genéricas:
    - classList
    - className
    - id

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:
  - Genéricas:
    - classList
    - className
    - id
  - Particulares:

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:
  - Genéricas:
    - classList
    - className
    - id
  - Particulares:
    - src

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:
  - Genéricas:
    - classList
    - className
    - id
  - Particulares:
    - src
    - href

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:
  - Genéricas:
    - classList
    - className
    - id
  - Particulares:
    - src
    - href
    - type

Manipulación del DOM

- El Document Object Model
- Nodos HTML
- Propiedades de los nodos:
  - Genéricas:
    - classList
    - className
    - id
  - Particulares:
    - src
    - href
    - type
    - disabled

Manipulación del DOM

```
<header class="cabecera titulo">
  <h1 id="titulo-pagina">Ejemplo DOM</h1>
</header>
<div>
  <h2>Esto es un documento HTML</h2>
  <img src="img/html.jpg" alt="Documento HTML">
  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Non <a href="https://consectetur.com">consectetur</a> sapie
    omnis dolore obcaecati veniam necessitatibus sit quia praesentium, nesciunt suscipit. Adipisci illum tempore fuga.
  </p>
  <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Omnis, consequatur? Nostrum repudiandae qui expedita
    optio exercitationem sapiente rem, quisquam ipsam, sunt veniam dolorum architecto similique! Iusto quibusdam omnis
    tenetur nihil.</p>
  <input type="text" placeholder="Introduce un valor">
  <button type="submit" disabled>
    Enviar
  </button>
</div>
```

Manipulación del DOM

Manipulación del DOM

- Eventos

Manipulación del DOM

- Eventos
- Tipos de eventos

Manipulación del DOM

- Eventos
- Tipos de eventos
- Cancelar la operación por defecto:
  - evento.preventDefault()

Manipulación del DOM

- Eventos
- Tipos de eventos
- Cancelar la operación por defecto:
  - evento.preventDefault()
- Quién ha emitido el evento:
  - evento.target

create-react-app

create-react-app

- Crea el proyecto

create-react-app

- Crea el proyecto
- react-scripts

create-react-app

- Crea el proyecto
- react-scripts
- eslint

create-react-app

- Crea el proyecto
- react-scripts
- eslint
- webpack

create-react-app

- Crea el proyecto
- react-scripts
- eslint
- webpack
- servidor web de desarrollo

create-react-app

- Crea el proyecto
- react-scripts
- eslint
- webpack
- servidor web de desarrollo
- compila el código

Directorios

Directorios

- **/src**: código fuente para desarrollar la app, Webpack lo tomará como origen para crear todo el código final.

Directorios

- **/src**: código fuente para desarrollar la app, Webpack lo tomará como origen para crear todo el código final.
- **/public**: archivos estáticos, Webpack no los procesará, simplemente los copiará en la carpeta de build.

Directorios

- **/src**: código fuente para desarrollar la app, Webpack lo tomará como origen para crear todo el código final.
- **/public**: archivos estáticos, Webpack no los procesará, simplemente los copiará en la carpeta de build.
- **/build**: carpeta donde irá el código final de la app, generado por Webpack.

Despliegue

Despliegue

- La carpeta build
- Netlify

Componentes

Componentes

- Un tipo de elementos React
- Componentes funcionales:
  - función
  - retorna elementos React
  - se exporta

JSX

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**
- Atajo para no escribir la función React.createElement()

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**
- Atajo para no escribir la función React.createElement()
- Elementos React

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**
- Atajo para no escribir la función React.createElement()
- Elementos React
- Un solo padre

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**
- Atajo para no escribir la función React.createElement()
- Elementos React
- Un solo padre
- className, htmlFor

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**
- Atajo para no escribir la función React.createElement()
- Elementos React
- Un solo padre
- className, htmlFor
- Etiquetas autocerradas

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**
- Atajo para no escribir la función React.createElement()
- Elementos React
- Un solo padre
- className, htmlFor
- Etiquetas autocerradas
- Expresiones JS dentro con { }

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**
- Atajo para no escribir la función React.createElement()
- Elementos React
- Un solo padre
- className, htmlFor
- Etiquetas autocerradas
- Expresiones JS dentro con { }
- Condiciones

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**
- Atajo para no escribir la función React.createElement()
- Elementos React
- Un solo padre
- className, htmlFor
- Etiquetas autocerradas
- Expresiones JS dentro con { }
- Condiciones
- Bucles

JSX

- Sintaxis que se parece a HTML pero **es JavaScript**
- Atajo para no escribir la función React.createElement()
- Elementos React
- Un solo padre
- className, htmlFor
- Etiquetas autocerradas
- Expresiones JS dentro con { }
- Condiciones
- Bucles
- Eventos

Componentes dentro de componentes

Componentes dentro de componentes

- Etiqueta propia

Componentes dentro de componentes

- Etiqueta propia
- props

Componentes dentro de componentes

- Etiqueta propia
- props
- children

Componentes dentro de componentes

- Etiqueta propia
- props
- children
- propTypes (npm install prop-types)
  - componente.propTypes = {
    ...
    }

Componentes dentro de componentes

- Etiqueta propia
- props
- children
- propTypes (npm install prop-types)
  - componente.propTypes = {
    ...
    }
  - PropTypes.string, PropTypes.number, PropTypes.bool, PropTypes.arrayOf(), PropTypes.shape()

Componentes dentro de componentes

- Etiqueta propia
- props
- children
- propTypes (npm install prop-types)
  - componente.propTypes = {
    ...
    }
  - PropTypes.string, PropTypes.number, PropTypes.bool, PropTypes.arrayOf(), PropTypes.shape()
  - PropTypes.isRequired

El state

El state

- El hook useState()
- Devuelve un array con [variable, setVariable]
- React vigilará cambios en la variable y hará que la UI reaccione a ellos
- No modifico nunca el valor de la variable directamente
- Cada vez que se modifica el valor, se llama de nuevo al componente

Hooks básicos

Hooks básicos

- useState(valorInicial)
  - Sirve para declarar una variable de estado.
  - Devuelve un array con [variable, setVariable].

Hooks básicos

- useState(valorInicial)
  - Sirve para declarar una variable de estado.
  - Devuelve un array con [variable, setVariable].
- useEffect(callback, dependencias)
  - Sirve para ejecutar una función sólo cuando cambie alguna de las dependencias.
  - No devuelve nada.

Hooks básicos

- useState(valorInicial)
  - Sirve para declarar una variable de estado.
  - Devuelve un array con [variable, setVariable].
- useEffect(callback, dependencias)
  - Sirve para ejecutar una función sólo cuando cambie alguna de las dependencias.
  - No devuelve nada.
- useRef(valor)
  - Sirve para declarar una variable sin que se redeclare en cada llamada al componente.
  - Devuelve un objeto con una propiedad current cuyo valor es el que le hemos pasado al hook.

Hooks básicos

Hooks básicos

- useMemo(callback, dependencias)
  - Sirve para que se calcule un valor y se memorice, y sólo se vuelva a recalcular cuando cambie alguna de las dependencias.
  - Devuelve el valor calculado.

Hooks básicos

- useMemo(callback, dependencias)
  - Sirve para que se calcule un valor y se memorice, y sólo se vuelva a recalcular cuando cambie alguna de las dependencias.
  - Devuelve el valor calculado.
- useCallback(callback, dependencias)
  - Sirve para memorizar una función y que se declare sólo una vez, excepto si cambia alguna de las dependencias.
  - Devuelve la función callback.

Formularios

Formularios

- Vincular cada control de formulario a una variable de estado, a partir de su propiedad value o checked, según el tipo de control.

Formularios

- Vincular cada control de formulario a una variable de estado, a partir de su propiedad value o checked, según el tipo de control.
- Vincular el evento onChange de cada control a una función que modifique la variable del punto anterior a partir del e.target

Formularios

```jsx
<input
  type="text"
  id="nombre"
  name="nombre"
  value={formDatos.nombre}
  onChange={modificarDatos}
  required
/>
```

Formularios

```
const modificarDatos = e => {
  const nombrePropiedad = e.target.name;
  setFormDatos({
    ...formDatos,
    [nombrePropiedad]: e.target.type === "checkbox" ? e.target.checked : e.target.value,
  });
};
```

Context

Context

- Permite poner un valor a disposición de un componente y todos sus descendientes sin tener que pasarlo por props.
- Podemos pasar un objeto para poder poner varios valores a disposición de los componentes.

Context

Context

- Crear un contexto: createContext()

Context

- Crear un contexto: createContext()
- Consumir un valor de un contexto: useContext(NombreContext)

Context

- Crear un contexto: createContext()
- Consumir un valor de un contexto: useContext(NombreContext)
- Colocar un proveedor de contexto englobando a otros componentes y proporcionando el valor: <NombreContext value={valor}>

Context

```
1 // Crear un contexto
2 const NombreContexto = createContext();
```

Context

```
1 // Consumir un valor de un contexto
2 const valor = useContext(NombreContexto);
```

Context

```
1 // Englobar componentes con un proveedor de contexto
2 <NombreContexto.Provider value={valor}>
3   <Componentes>...</Componentes>
4 </NombreContexto.Provider>
```

Reducers

Reducers

- Permiten aislar toda la lógica de modificación del estado

Reducers

- Permiten aislar toda la lógica de modificación del estado
- Funcionan mediante llamado de acciones

Reducers

- Permiten aislar toda la lógica de modificación del estado
- Funcionan mediante llamado de acciones
- Las acciones son objetos

Reducers

- Permiten aislar toda la lógica de modificación del estado
- Funcionan mediante llamado de acciones
- Las acciones son objetos
- Las acciones se llaman mediante dispatch()

Reducers

```
const datosReducer = (state, action) => {
        switch (action.type) {
                case "accion1":
                        return stateModificado1;
                case "accion2":
                        return stateModificado2;
                case "accion3":
                        return stateModificado3;
                default:
                        return state;
        }
}
```

Reducers

Reducers

- Usamos el hook useReducer para obtener el estado y la función dispatch.

Reducers

```
1 const [state, dispatch] = useReducer(functionReductora, estadoInicial);
```

Enrutado

Enrutado

- La librería react-router-dom

Enrutado

- La librería react-router-dom
- Toma el control de los links de navegación y hace que haya una página u otra en base a la URL, sin que se recargue la app

Enrutado

- La librería react-router-dom
- Toma el control de los links de navegación y hace que haya una página u otra en base a la URL, sin que se recargue la app
- Haremos que cada página sea un componente

Enrutado

- La librería react-router-dom
- Toma el control de los links de navegación y hace que haya una página u otra en base a la URL, sin que se recargue la app
- Haremos que cada página sea un componente
- Vincularemos cada ruta con un componente

Enrutado

Enrutado

- Englobaremos todo lo que queramos que esté controlado por el router en un componente BrowserRouter

Enrutado

- Englobaremos todo lo que queramos que esté controlado por el router en un componente BrowserRouter
- Lo renombraremos a Router

Enrutado

```
1 // Al principio del archivo:
2 import { BrowserRouter as Router } from "react-router-dom";
3
4 // En el JSX:
5 <Router>
6   <RestoDeComponentes>...</RestoDeComponentes>
7 </Router>
```

Enrutado

Enrutado

- Usaremos los componentes Switch y Route para indicar qué componente se tiene que renderizar con cada ruta

Enrutado

- Usaremos los componentes Switch y Route para indicar qué componente se tiene que renderizar con cada ruta
- También podemos establecer redirecciones con el componente Redirect

Enrutado

- Usaremos los componentes Switch y Route para indicar qué componente se tiene que renderizar con cada ruta
- También podemos establecer redirecciones con el componente Redirect
- Las coincidencias se buscan por orden. Si se llega hasta el final es que no se ha encontrado ninguna coincidencia, por eso el 404 lo ponemos al final

Enrutado

```
<Switch>
  <Route path="/portada" exact>
    <Portada />
  </Route>
  <Route path="/listado" exact>
    <Listado />
  </Route>
  <Route path="/quienes-somos" exact>
    <QuienesSomos />
  </Route>
  <Route path="/" exact>
    <Redirect to="/portada" />
  </Route>
  <Route path="*" exact>
    <NotFound />
  </Route>
</Switch>
```

Enrutado

Enrutado

- Podemos establecer rutas con parámetros:
  /factura/4
  /alumno/2737
  /bus/32

Enrutado

- Podemos establecer rutas con parámetros:
  /factura/4
  /alumno/2737
  /bus/32
- Para recoger los parámetros desde el componente usamos el hook useParams

Enrutado

```
<Switch>
  ...
  <Route path="/factura/:id" exact>
    <Factura />
  </Route>
  <Route path="/editar/:tipo/:id" exact>
    <FormFactura />
  </Route>
  ...
</Switch>
```

Enrutado

```
<Switch>
  ...
  const Factura = () => {
    const { id } = useParams();
    return (
          ...
    )
  }

</Switch>
```

Enrutado

```
<Switch>
  ...
 const FormFactura = () => {
   const { tipo, id } = useParams();
   return (
         ...
   )
 }

</Switch>
```

Enrutado

Enrutado

- Podemos recoger los parámetros query con el hook useLocation

Enrutado

- Podemos recoger los parámetros query con el hook useLocation
- Devuelve un objeto con una propiedad search

Enrutado

- Podemos recoger los parámetros query con el hook useLocation
- Devuelve un objeto con una propiedad search
- Se utiliza la clase URLSearchParams de JavaScript para parsear el resultado

Enrutado

```
const FormFactura = () => {
  const { tipo, id } = useParams();
  const query = new URLSearchParams(useLocation().search);
  // query.get("dato") para obtener el parámetro query que queramos
  return (
        ...
  )
}
```

mario@mariogl.com

@marioglweb