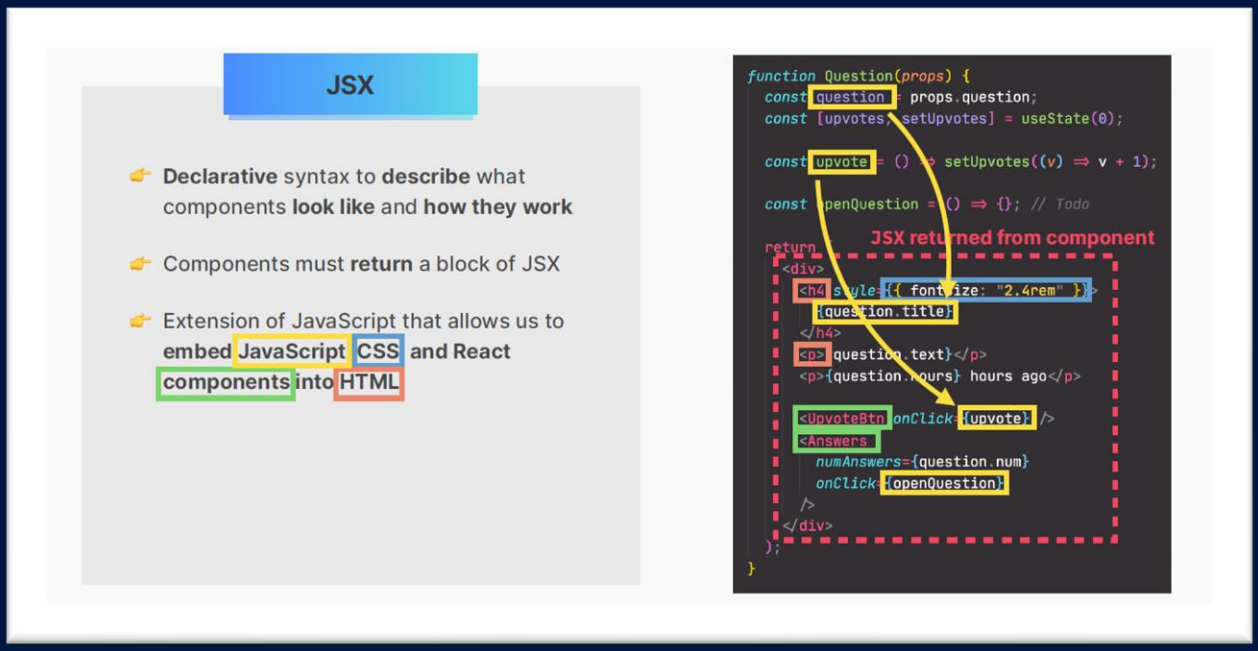
React

JSX

JSX stands for JavaScript XML



React

JSX

- **Declarative** syntax to **describe** what components **look like** and **how they work**
- Components must **return** a block of JSX
- Extension of JavaScript that allows us to **embed JavaScript, CSS, and React components into HTML**
- Each JSX element is **converted** to a `React.createElement` function call
- We could use React **without JSX**

```
<header>
  <h1 style="color: red">
    Hello React!
  </h1>
</header>
```

BABEL

```
React.createElement(
  'header',
  null,
  React.createElement(
    'h1',
    { style: { color: 'red' } },
    'Hello React!'
  )
);
```

Hello React!

IMPERATIVE

"How to do things"

JS

- Manual DOM element selections and DOM traversing
- Step-by-step DOM mutations until we reach the desired UI

```
const title = document.querySelector("title")
const upvoteBtn = document.querySelector("btn")
title.textContent = `[0] ${question.title}`;
let upvotes = 0;
upvoteBtn.addEventListener("click", function(){
  upvotes++;
  title.textContent =
    `[${upvotes}] ${question.title}`;
  title.classList.add("upvoted");
});
```

DECLARATIVE

"What we want"

- Describe what UI should look like using JSX, **based on current data**
- React is an **abstraction** away from DOM: **we never touch the DOM**
- Instead, we think of the UI as a **reflection of the current data**

```
function Question(props) {
  const question = props.question;
  const [upvotes, setUpvotes] = useState(0);
  const upvote = () => setUpvotes((v) => v + 1);

  return (
    <div>
      <h4>{question.title}</h4>
      <p>{question.text}</p>
      <UpvoteBtn
        onClick={upvote}
        upvotes={upvotes}
      />
    </div>
  );
}
```

React

GENERAL JSX RULES

- JSX works essentially like HTML, but we can enter "**JavaScript mode**" by using `{}` (for text or attributes)
- We can place **JavaScript expressions** inside `{}`. Examples: reference variables, create arrays or objects, `[].map()`, ternary operator
- Statements are **not allowed** (`if/else`, `for`, `switch`)
- JSX produces a **JavaScript expression**

==

```
const el = <h1>Hello React!</h1>;
const el = React.createElement("h1", null, "Hello React!");
```

1. We can place **other pieces of JSX** inside `{}`
2. We can write JSX **anywhere** inside a component (in `if/else`, assign to variables, pass it into functions)

- A piece of JSX can only have **one root element**. If you need more, use `<React.Fragment>` (or the short `<>`)

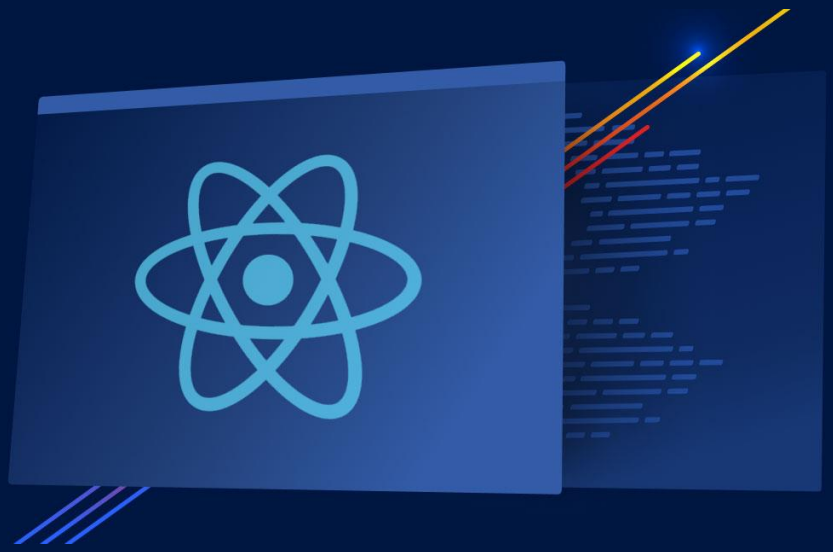
DIFFERENCES BETWEEN JSX AND HTML

- `className` instead of HTML's `class`
- `htmlFor` instead of HTML's `for`
- Every tag needs to be **closed**. Examples: `<img />` or `<br />`
- All event handlers and other properties need to be **camelCased**. Examples: `onClick` or `onMouseOver`
- **Exception**: `aria-*` and `data-*` are written with dashes like in HTML
- CSS inline styles are written like this: `{{<style>}}` (to reference a variable, and then an object)
- CSS property names are also **camelCased**
- Comments need to be in `{}` (because they are JS)

React

Do you find it helpful?

Let me know down in the comments.



Click To Follow For More On LinkedIN