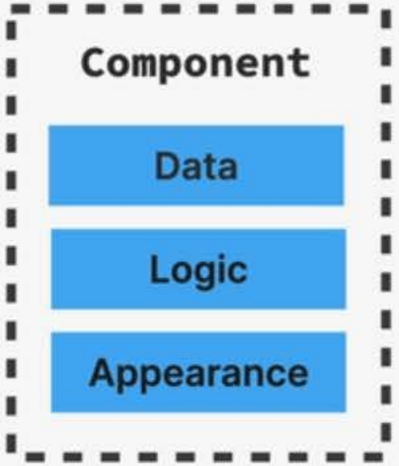
WHAT IS JSX?

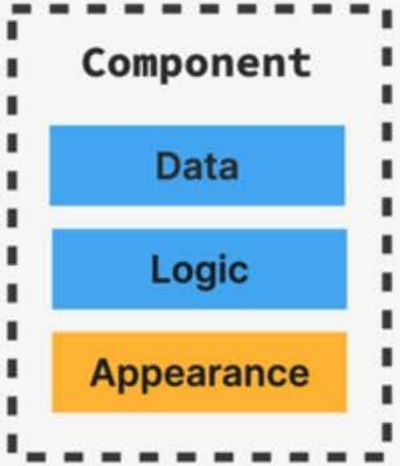
WHAT IS JSX?



WHAT IS JSX?

JSX

👉 **Declarative** syntax to **describe** what components **look like** and **how they work**



WHAT IS JSX?

JSX

- 👉 **Declarative** syntax to **describe** what components **look like** and **how they work**
- 👉 Components must **return** a block of JSX

```jsx
function Question(props) {
  const question = props.question;
  const [upvotes, setUpvotes] = useState(0);

  const upvote = () => setUpvotes((v) => v + 1);

  const openQuestion = () => {}; // Todo

  return (
    <div>
      <h4 style={{ fontSize: "2.4rem" }}>
        {question.title}
      </h4>
      <p>{question.text}</p>
      <p>{question.hours} hours ago</p>

      <UpvoteBtn onClick={upvote} />
      <Answers
        numAnswers={question.num}
        onClick={openQuestion}
      />
    </div>
  );
}
```

JSX returned from component

WHAT IS JSX?

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**

```jsx
function Question(props) {
  const question = props.question;
  const [upvotes, setUpvotes] = useState(0);

  const upvote = () => setUpvotes((v) => v + 1);

  const openQuestion = () => {}; // Todo

  return (
    <div>
      <h4 style={{ fontSize: "2.4rem" }}>
        {question.title}
      </h4>
      <p>{question.text}</p>
      <p>{question.hours} hours ago</p>

      <UpvoteBtn onClick={upvote} />
      <Answers
        numAnswers={question.num}
        onClick={openQuestion}
      />
    </div>
  );
}
```

JSX returned from component

WHAT IS JSX?

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**

```jsx
function Question(props) {
  const question = props.question;
  const [upvotes, setUpvotes] = useState(0);

  const upvote = () => setUpvotes((v) => v + 1);

  const openQuestion = () => {}; // Todo

  return (
    <div>
      <h4 style={{ fontSize: "2.4rem" }}>
        {question.title}
      </h4>
      <p>{question.text}</p>
      <p>{question.hours} hours ago</p>

      <UpvoteBtn onClick={upvote} />
      <Answers
        numAnswers={question.num}
        onClick={openQuestion}
      />
    </div>
  );
}
```

JSX returned from component

WHAT IS JSX?

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**

```jsx
function Question(props) {
  const question = props.question;
  const [upvotes, setUpvotes] = useState(0);

  const upvote = () => setUpvotes((v) => v + 1);

  const openQuestion = () => {}; // Todo

  return (
    <div>
      <h4 style={{ fontSize: "2.4rem" }}>
        {question.title}
      </h4>
      <p>{question.text}</p>
      <p>{question.hours} hours ago</p>

      <UpvoteBtn onClick={upvote} />
      <Answers
        numAnswers={question.num}
        onClick={openQuestion}
      />
    </div>
  );
}
```

JSX returned from component

WHAT IS JSX?

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**

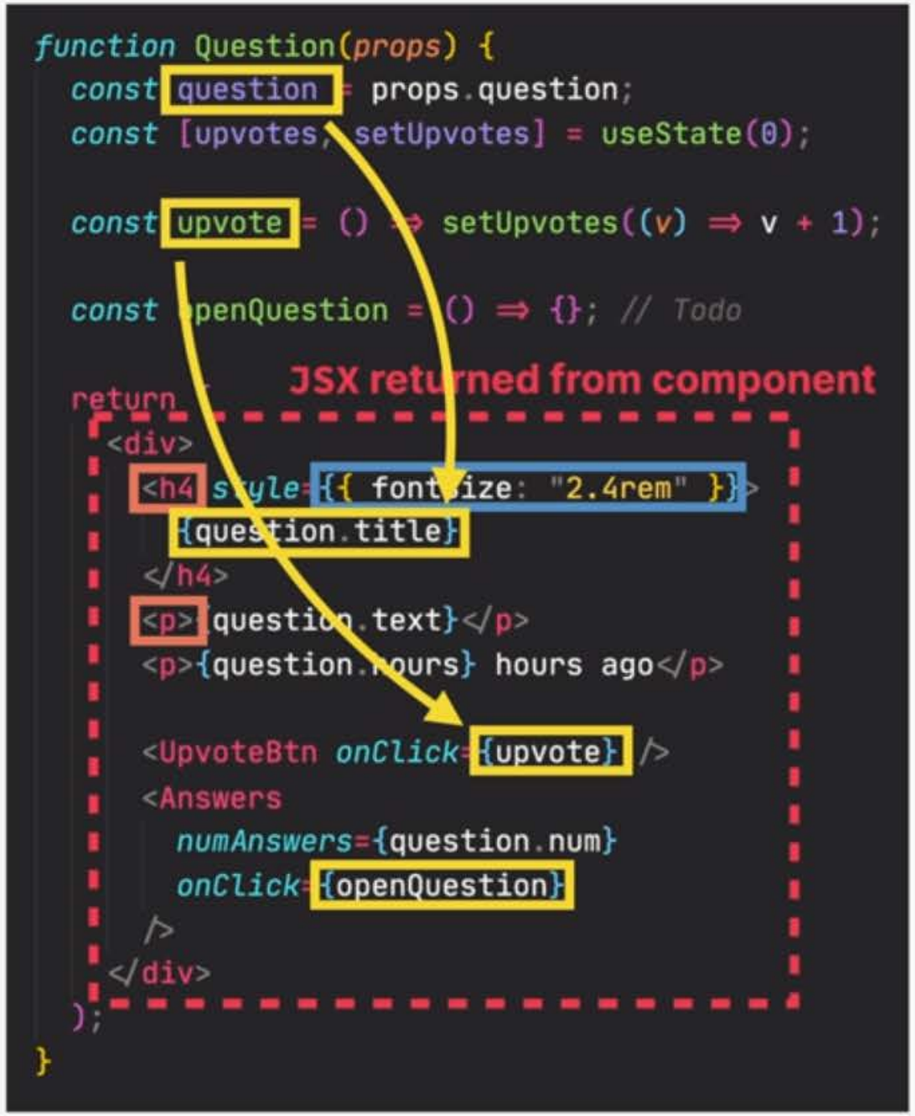
```jsx
function Question(props) {
  const question = props.question;
  const [upvotes, setUpvotes] = useState(0);

  const upvote = () => setUpvotes((v) => v + 1);

  const openQuestion = () => {}; // Todo

  return (
    <div>
      <h4 style={{ fontSize: "2.4rem" }}>
        {question.title}
      </h4>
      <p>{question.text}</p>
      <p>{question.hours} hours ago</p>

      <UpvoteBtn onClick={upvote} />
      <Answers
        numAnswers={question.num}
        onClick={openQuestion}
      />
    </div>
  );
}
```

JSX returned from component

WHAT IS JSX?

JSX

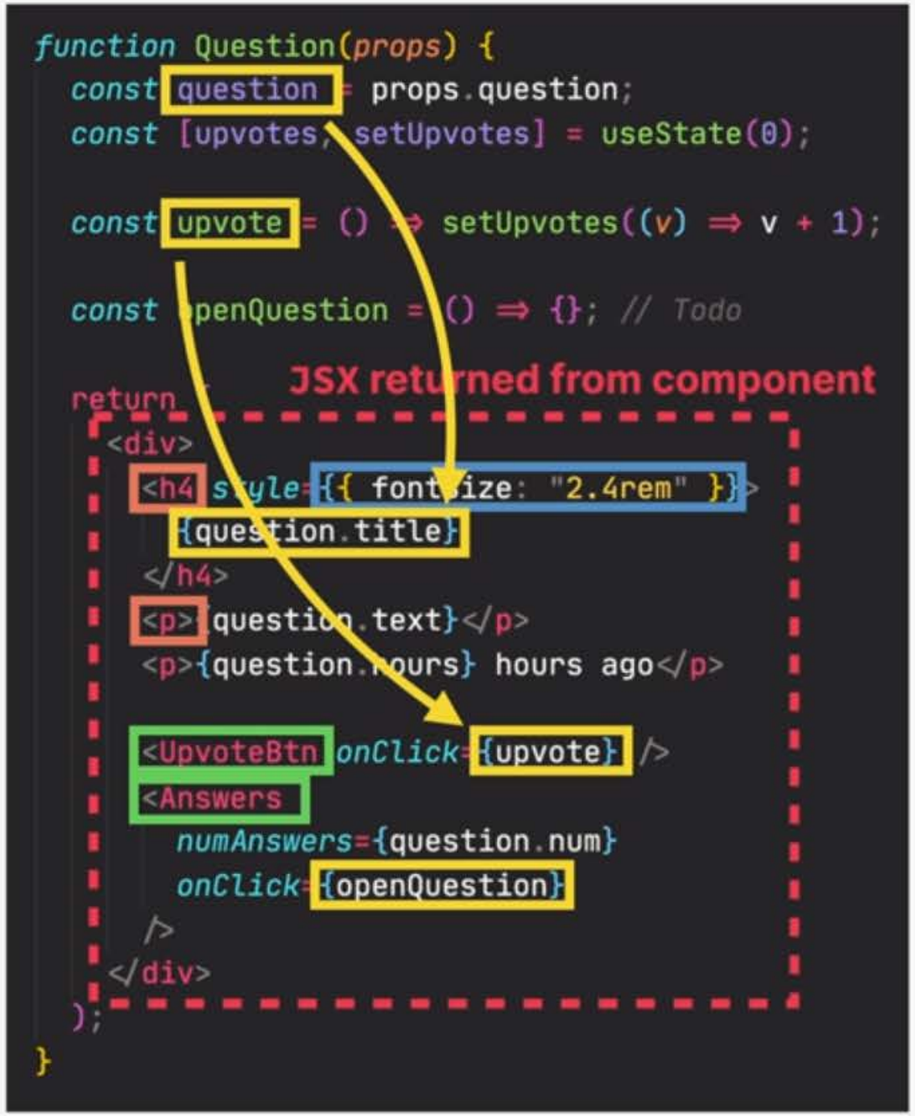
- 👉 **Declarative** syntax to **describe** what components **look like** and **how they work**
- 👉 Components must **return** a block of JSX
- 👉 Extension of JavaScript that allows us to **embed JavaScript** **CSS** **and React components into HTML**



WHAT IS JSX?

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**



WHAT IS JSX?

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**

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

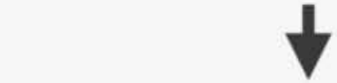
```jsx
React.createElement(
  'header',
  null,
  React.createElement(
    'h1',
    { style: { color: 'red' } },
    'Hello React!'
  )
);
```

WHAT IS JSX?

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**

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



BABEL

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

WHAT IS JSX?

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

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

BABEL

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

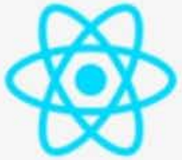
WHAT IS JSX?

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

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


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



Hello React!

WHAT IS JSX?

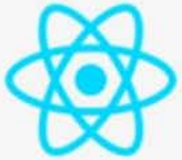
JSX

- **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**

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


BABEL

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

Hello React!

JSX IS **DECLARATIVE**

JSX IS DECLARATIVE

IMPERATIVE

JSX IS DECLARATIVE

IMPERATIVE

JS

```
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");
});
```

JSX IS DECLARATIVE

IMPERATIVE

👉 Manual DOM element selections and DOM traversing

JS

```
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");
});
```

JSX IS DECLARATIVE

IMPERATIVE

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

JS

```
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");
});
```

JSX IS DECLARATIVE

IMPERATIVE

"How to do things"

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

JS

```
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");
});
```

JSX IS DECLARATIVE

IMPERATIVE ← *"How to do things"*

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

JS

```
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");
});
```

```
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>
  );
}
```

JSX IS DECLARATIVE

IMPERATIVE

"How to do things"

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

JS

```
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

```
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>
  );
}
```

JSX IS DECLARATIVE

IMPERATIVE

"How to do things"

- 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

- Describe what UI should look like using JSX, **based on 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>
  );
}
```

JSX IS DECLARATIVE

IMPERATIVE

"How to do things"

- 👉 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

- 👉 Describe what UI should look like using JSX, **based on current data**
- 👉 React is an **abstraction** away from DOM: **we never touch the DOM**

```
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>
  );
}
```

JSX IS DECLARATIVE

IMPERATIVE

"*How* to do things"

- 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

- 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>
  );
}
```