# React.js Week 1 – Day 2: JSX Basics

This canvas explains **JSX** in depth with concrete examples you can copy-paste and run. Read the rules carefully — many beginners trip over tiny syntax differences between HTML and JSX.

## 1) What is JSX?

* **JSX** (JavaScript XML) is a syntax extension for JavaScript that looks like HTML and is used with React.
* It is **not** a string or HTML — it compiles (via Babel) into React.createElement() calls that produce React elements.
* Example: const el = <h1>Hello</h1> compiles roughly to React.createElement('h1', null, 'Hello').

**Why use JSX?** - Combines UI markup and JavaScript logic in the same place — clearer and more powerful than string templates. - Easier to visualize component structure and pass dynamic data.

## 2) Basic rules & important differences from HTML

* **JSX expressions**: put any JavaScript expression inside curly braces {}. Example: {1 + 2}, {user.name}, {items.map(x => x)}.
* **Single root element**: a component must return one root. Use a wrapper <div> or a fragment <>...</> for multiple siblings.
* **Use className instead of class** (because class is a reserved word in JS).
* **Use htmlFor instead of for** on <label>.
* **Style is an object**: style={{ backgroundColor: 'red', fontSize: 14 }} (camelCase keys).
* **Self-closing tags** are required for void elements: <img />, <input />, <br />.
* **Comments** in JSX use {/\* comment \*/} not <!-- -->.

## 3) Common patterns & examples

### Embedding variables and expressions

const name = 'Rehan';  
const greeting = <h1>Hello, {name}!</h1>;

### Using functions inside JSX

function formatName(user) {  
 return user.firstName + ' ' + user.lastName;  
}  
const user = { firstName: 'Rehan', lastName: 'Abbasi' };  
// inside JSX: <h1>Hello, {formatName(user)}</h1>

### Conditional rendering (short and readable)

const isLoggedIn = true;  
{ isLoggedIn ? <p>Welcome back</p> : <p>Please sign in</p> }

### Rendering arrays (briefly; keys covered later)

const nums = [1, 2, 3];  
<p>{ nums.map(n => <span key={n}>{n} </span>) }</p>

### Inline style object

const boxStyle = { padding: '12px', borderRadius: '6px', backgroundColor: '#f3f3f3' };  
<div style={boxStyle}>Styled box</div>

### Fragment for multiple top-level nodes

return (  
 <>  
 <h1>Title</h1>  
 <p>Paragraph</p>  
 </>  
)

### dangerouslySetInnerHTML (use rarely)

<div dangerouslySetInnerHTML={{ \_\_html: '<strong>Raw HTML</strong>' }} />

**Warning:** This bypasses React’s escaping — only use with trusted content.

## 4) A full example component you can paste into src/JSXBasics.js

// src/JSXBasics.js  
import React from 'react';  
import logo from './logo.svg'; // or use public assets  
import './JSXBasics.css';  
  
function JSXBasics() {  
 const name = 'Rehan';  
 const user = { firstName: 'Rehan', lastName: 'Abbasi' };  
 const now = new Date();  
 const numbers = [1, 2, 3, 4];  
  
 function formatName(u) {  
 return u.firstName + ' ' + u.lastName;  
 }  
  
 const boxStyle = { padding: '12px', borderRadius: '8px', backgroundColor: '#f9fafb' };  
  
 const exampleElement = <em>JSX can even be stored in variables.</em>;  
  
 return (  
 <>  
 <div className="card" style={boxStyle}>  
 <img src={logo} alt="logo" width="48" />  
 <h2>Hello, {formatName(user)}!</h2>  
 <p>{exampleElement}</p>  
 <p>Today is {now.toLocaleDateString()}</p>  
 <p>Math in JSX: 2 + 3 = {2 + 3}</p>  
  
 <p>  
 Numbers: {numbers.map(n => <span key={n}>{n} </span>)}  
 </p>  
  
 {/\* This is a JSX comment. \*/}  
 <label htmlFor="nameInput">Your name:</label>  
 <input id="nameInput" defaultValue={name} />  
 </div>  
  
 <div style={{ marginTop: '12px' }}>  
 {/\* small example of raw HTML insertion (only if necessary) \*/}  
 <div dangerouslySetInnerHTML={{ \_\_html: '<strong>Raw HTML example</strong>' }} />  
 </div>  
 </>  
 );  
}  
  
export default JSXBasics;

Create the optional src/JSXBasics.css with a minimal style so you can see className working:

/\* src/JSXBasics.css \*/  
.card {  
 border: 1px solid #e5e7eb;  
 padding: 12px;  
 border-radius: 8px;  
 max-width: 520px;  
}

## 5) Common pitfalls (read this carefully)

* class → use className.
* for → use htmlFor.
* style expects an **object**, not a CSS string. Use camelCase keys.
* Can’t put statements inside {} — only expressions (no if statements inside; use ternary or compute before return).
* Every element in a list should have a **key** (unique stable id preferred).
* Don’t forget to **import** your component in App.js and use <JSXBasics />.

## 6) Task (practice — 15–20 minutes)

Follow these exact steps and paste the final src/JSXBasics.js code here when done (I will review it):

1. Create a new file src/JSXBasics.js and paste the component example from section **4**.
2. Create src/JSXBasics.css with the CSS provided (optional but recommended).
3. Open src/App.js and **import** the new component:

* import JSXBasics from './JSXBasics';  
  // then inside App's return: <JSXBasics />

1. Run npm start and check the browser:
   * You should see your logo, the greeting with your full name, today’s date, the numbers mapped, and the input with a default value.
   * Verify the card CSS is applied (change the border color to test).
2. Extra challenge (optional): inside JSXBasics, add a small conditional that shows Weekend when today is Saturday/Sunday or Weekday otherwise (use now.getDay(); Sunday = 0, Saturday = 6).

**When done:** paste the contents of your src/JSXBasics.js here and I’ll review and point out any improvements or subtle bugs.

## 7) Quick reference cheatsheet

* Expression: { expression }
* Fragment: <>...</>
* Class: className
* For label: htmlFor
* Style object: style={{ camelCaseProp: value }}
* Comment: {/\* comment \*/}
* Danger: dangerouslySetInnerHTML={{ \_\_html: '...' }}

Good — this is the JSX foundation. Once you finish the task and paste your file, I’ll review for correctness and then we’ll move to **Day 3: Props**.