

# React JSX

Let's learn about Javascript XML role in React.

## We'll cover the following ^

- Exercises:

Recall that I mentioned the returned output of the App component resembles HTML. This output is called JSX, which mixes HTML and JavaScript. Let's see how this works for displaying the variable:

```
.App {
  text-align: center;
}


.App-logo {
  animation: App-logo-spin infinite 20s linear;
  height: 80px;
}

.App-header {
  background-color: #222;
  height: 150px;
  padding: 20px;
  color: white;
}


.App-title {
  font-size: 1.5em;
}

.App-intro {
  font-size: large;
}

@keyframes App-logo-spin {
  from { transform: rotate(0deg); }
  to { transform: rotate(360deg); }
}
```

 **Note:** In the app onwards we can now see the output in the browser since we are starting the app using `npm start`. This will start the live-server, which will then begin to listen for any changes made in our code files.

See the output in action, both in the **output tab** and the **host link** provided below. In case of any changes you make to your code, the live-server will detect these, just press **Run**.

 **Note:** The terminal remains alive for 30 minutes for each session, after which the server is killed;

The application is started with `npm start`, and now you can look for the rendered variable in the browser, which should read: “Hello React”.

Let's focus on the HTML, which is expressed almost the same in JSX. An input field with a label can be defined as follows:



























```






















0 0 0 0 0   ä F    9 5 @@    ° n   PNG
IHDR      (-S   äPLTE""""""2PX=r)7;*:>H-BGE8do5Xb6[eK®K~1MU9gs3S
IHDR      x@ÎÊ ePLTE""""""2RZN¢¹J«3R[J(-)59YÁpØKS4W`Q«ÄL?%+-ØJR
?^q÷ñíÜï.,ïisæŸ_TttÔ% 1#□□/(ì□-[□□□è`□è`î□ÚíÂðZd5□□□□?ÏebZ¿p□i.Üæ□□□iqÎ□+1°□}Â□5ù ìçd
IHDR      □□ DæÆ APLTE """"""2RZVöÖ_ôðU·Ñ=r□$( )'25]ÍíC□□θLS<o>}X
IHDR @ @□□ □·□ì □:PLTE """"""
ßBqÇ8Ü0´mKĚ±mÆŋmÜü·yilè□îäYiüë Äî_Äî?i÷□ý+ð□□ÄA□|□ù{□□´¿□_En□).□JĒDæ<□
Θ-φZTsΘR*(□ ´□□J□□□□u□X/□4J□9□j5·DEµ4kÇ4□&i¥V4U□i®D□□´□vsf:àg,□çèBC»i$ŋ□□íûi□□á□@□ð□I:
-e>Ù□°«φXð¢î}ß``ëÜÑ;□Äön´□øvÄý□î.ÿ1 □ëxÄO@&v/Äp_□ð\ð□Ç\i.□□%+θ□□;□□□!□fÊ□|´0%Ä JY·0□Ä□'/Ä_]_

```

We specified three HTML attributes here: `htmlFor`, `id`, and `type`. Where `id` and `type` should be familiar from native HTML, `htmlFor` might be new. The `htmlFor` reflects the `for` attribute in HTML. JSX replaces a handful of internal HTML attributes, but you can find all the [supported HTML attributes](#) in React's documentation, which follow the camelCase naming convention. Expect to come across more JSX-specific attributes like `className` and `onClick` instead of `class` and `onclick`, as you learn more about React.

We will revisit the HTML input field for implementation details later; for now, let's return to JavaScript in JSX. We have defined a string primitive to be displayed in the App component, and the same can be done with a JavaScript object:

```

IHDR  (S äPLTE"2PX=r);*:>H-BGE8do5Xb6[eK^K~1MU9gs3S
IHDR  x@ÍÊ ePLTE"2RZN¢¹J«3R[J~)59YÁp0KS4W`Q«ÄL²%+-0JR
?^q÷ñiÛi.},isæÝ_TttÔ% 1#/(i-[è`è`ÏÚiÅðZd5?ÏebZ¿p.i.ÛæiqÎ+1°}Â5ù içd
IHDR  DÆ APLTE "2RZVºÖ_ÔôU.Ñ=r$()'25]ÎiC00LS<o>X
IHDR  @ @ .i :PLTE
øβqÇ8Û ´mKË±mÆmÜü·yi!èîâYÏuë ÄÏ_Äi?i÷ý+ðÄÄ|û{´¿;_En).JËDæ<
@~¢Z\Ts0R*(  @JuuX/4J9;5·DEµ4kÇ4&i¥V4Ú;®vsvf:àg,¢èBC»i$¶ºÍûiá@ôI_
-è>Ûº«¢XÔ¢i}ß"ëÜÑ;ÄöN´øvÅýî.ÿ1 èxÄ0@&v/Äp_ö\ðÇ\i.º%+0;!!fÊ|´Ó%Â JY·OÂ'/'Ä]_

```

Remember, everything in curly braces in JSX can be used for JavaScript expressions (e.g. function execution):

```

  ä F  )  9 5 @@ ° n PNG
IHDR  (S äPLTE"2PX=r);*:>H-BGE8do5Xb6[eK^K~1MU9gs3S
IHDR  x@ÍÊ ePLTE"2RZN¢¹J«3R[J~)59YÁp0KS4W`Q«ÄL²%+-0JR
?^q÷ñiÛi.},isæÝ_TttÔ% 1#/(i-[è`è`ÏÚiÅðZd5?ÏebZ¿p.i.ÛæiqÎ+1°}Â5ù içd
IHDR  DÆ APLTE "2RZVºÖ_ÔôU.Ñ=r$()'25]ÎiC00LS<o>X
IHDR  @ @ .i :PLTE
øβqÇ8Û ´mKË±mÆmÜü·yi!èîâYÏuë ÄÏ_Äi?i÷ý+ðÄÄ|û{´¿;_En).JËDæ<
@~¢Z\Ts0R*(  @JuuX/4J9;5·DEµ4kÇ4&i¥V4Ú;®vsvf:àg,¢èBC»i$¶ºÍûiá@ôI_
-è>Ûº«¢XÔ¢i}ß"ëÜÑ;ÄöN´øvÅýî.ÿ1 èxÄ0@&v/Äp_ö\ðÇ\i.º%+0;!!fÊ|´Ó%Â JY·OÂ'/'Ä]_

```

JSX was initially invented for React, but it became useful for other modern libraries and frameworks after it gained popularity. It is one of my favorite things about React. Without any extra templating syntax (except for the curly braces), we are now able to use JavaScript in HTML.

## Exercises: #

- Confirm the [changes from the last section](#). This specifies the lines in which the changes were made in code. This makes it to easy to understand how it differs from the previous section.
- Read more about [React's JSX](#).
- Define more primitive and complex JavaScript data types and render them in JSX.
- Try to render a JavaScript array in JSX. If it's too complicated, don't worry, because you will learn more about this in the next lesson.



Which of the following statements is NOT true about JSX?

Retake Quiz

