DOM Attributes in React 16

September 08, 2017 by Dan Abramov

In the past, React used to ignore unknown DOM attributes. If you wrote JSX with an attribute that React doesn't recognize, React would just skip it. For example, this:

```
// Your code:
<div mycustomattribute="something" />
```

would render an empty div to the DOM with React 15:

```
// React 15 output:
<div />
```

In React 16, we are making a change. Now, any unknown attributes will end up in the DOM:

```
// React 16 output:
<div mycustomattribute="something" />
```

Why Are We Changing This?

React has always provided a JavaScript-centric API to the DOM. Since React components often take both custom and DOM-related props, it makes sense for React to use the `camelCase` convention just like the DOM APIs:

```
<div tabIndex={-1} />
```

This has not changed. However, the way we enforced it in the past forced us to maintain an allowlist of all valid React DOM attributes in the bundle:

```
// ...
summary: 'summary',
tabIndex: 'tabindex'
target: 'target',
title: 'title',
// ...
```

This had two downsides:

- You could not pass a custom attribute. This is useful for supplying browser-specific non-standard attributes, trying new DOM APIs, and integrating with opinionated third-party libraries.
- The attribute list kept growing over time, but most React canonical attribute names are already valid in the DOM. Removing most of the allowlist helped us reduce the bundle size a little bit.

With the new approach, both of these problems are solved. With React 16, you can now pass custom attributes to all HTML and SVG elements, and React doesn't have to include the whole attribute allowlist in the production version.

Note that you should still use the canonical React naming for known attributes:

```
// Yes, please
<div tabIndex={-1} />

// Warning: Invalid DOM property `tabindex`. Did you mean `tabIndex`?
<div tabindex={-1} />
```

In other words, the way you use DOM components in React hasn't changed, but now you have some new capabilities.

Should I Keep Data in Custom Attributes?

No. We don't encourage you to keep data in DOM attributes. Even if you have to, `data-` attributes are probably a better approach, but in most cases data should be kept in React component state or external stores.

However, the new feature is handy if you need to use a non-standard or a new DOM attribute, or if you need to integrate with a third-party library that relies on such attributes.

Data and ARIA Attributes

Just like before, React lets you pass `data-` and `aria-` attributes freely:

```
<div data-foo="42" />
<button aria-label="Close" onClick={onClose} />
```

This has not changed.

Accessibility is very important, so even though React 16 passes any attributes through, it still validates that `aria-` props have correct names in development mode, just like React 15 did.

Migration Path

We have included a warning about unknown attributes since React 15.2.0 which came out more than a year ago. The vast majority of third-party libraries have already updated their code. If your app doesn't produce warnings with React 15.2.0 or higher, this change should not require modifications in your application code.

If you still accidentally forward non-DOM props to DOM components, with React 16 you will start seeing those attributes in the DOM, for example:

```
<div myData='[Object object]' />
```

This is somewhat safe (the browser will just ignore them) but we recommend to fix these cases when you see them. One potential hazard is if you pass an object that implements a custom `toString()` or `valueOf()` method that throws. Another possible issue is that legacy HTML attributes like `align` and `valign` will now be passed to the DOM. They used to be stripped out because React didn't support them.

To avoid these problems, we suggest to fix the warnings you see in React 15 before upgrading to React 16.

Changes in Detail

We've made a few other changes to make the behavior more predictable and help ensure you're not making mistakes. We don't anticipate that these changes are likely to break real-world applications.

These changes only affect DOM components like `<div>`, not your own components.

Below is a detailed list of them.

- **Unknown attributes with string, number, and object values:**

```
<div mycustomattribute="value" />
<div mycustomattribute={42} />
<div mycustomattribute={myObject} />
```

React 15: Warns and ignores them.
React 16: Converts values to strings and passes them through.
Note: attributes starting with on are not passed through as an exception because this could become a potential security hole.

- **Known attributes with a different canonical React name:**

```
<div tabindex={-1} />
<div class="hi" />
```

React 15: Warns and ignores them.
React 16: Warns but converts values to strings and passes them through.
Note: always use the canonical React naming for all supported attributes.

- **Non-boolean attributes with boolean values:**

```
<div className={false} />
```

React 15: Converts booleans to strings and passes them through.
React 16: Warns and ignores them.

- **Non-event attributes with function values:**

```
<div className={function() {}} />
```

React 15: Converts functions to strings and passes them through.
React 16: Warns and ignores them.

- **Attributes with Symbol values:**

```
<div className={Symbol('foo')} />
```

React 15: Crashes.
React 16: Warns and ignores them.

- **Attributes with `NaN` values:**

```
<div tabIndex={0 / 0} />
```

React 15: Converts `NaN`s to strings and passes them through.
React 16: Converts `NaN`s to strings and passes them through with a warning.

While testing this release, we have also created an automatically generated table for all known attributes to track potential regressions.

Try It!

You can try the change in this CodePen.
It uses React 16 RC, and you can help us by testing the RC in your project!

Thanks

This effort was largely driven by Nathan Hunzaker who has been a prolific outside contributor to React.

You can find his work on this issue in several PRs over the course of last year: #6459, #7311, #10229, #10397, #10385, and #10470.

Major changes in a popular project can take a lot of time and research. Nathan demonstrated perseverance and commitment to getting this change through, and we are very thankful to him for this and other efforts.

We would also like to thank Brandon Dail and Jason Quense for their invaluable help maintaining React this year.

Future Work

We are not changing how custom elements work in React 16, but there are existing discussions about setting properties instead of attributes, and we might revisit this in React 17. Feel free to chime in if you'd like to help!