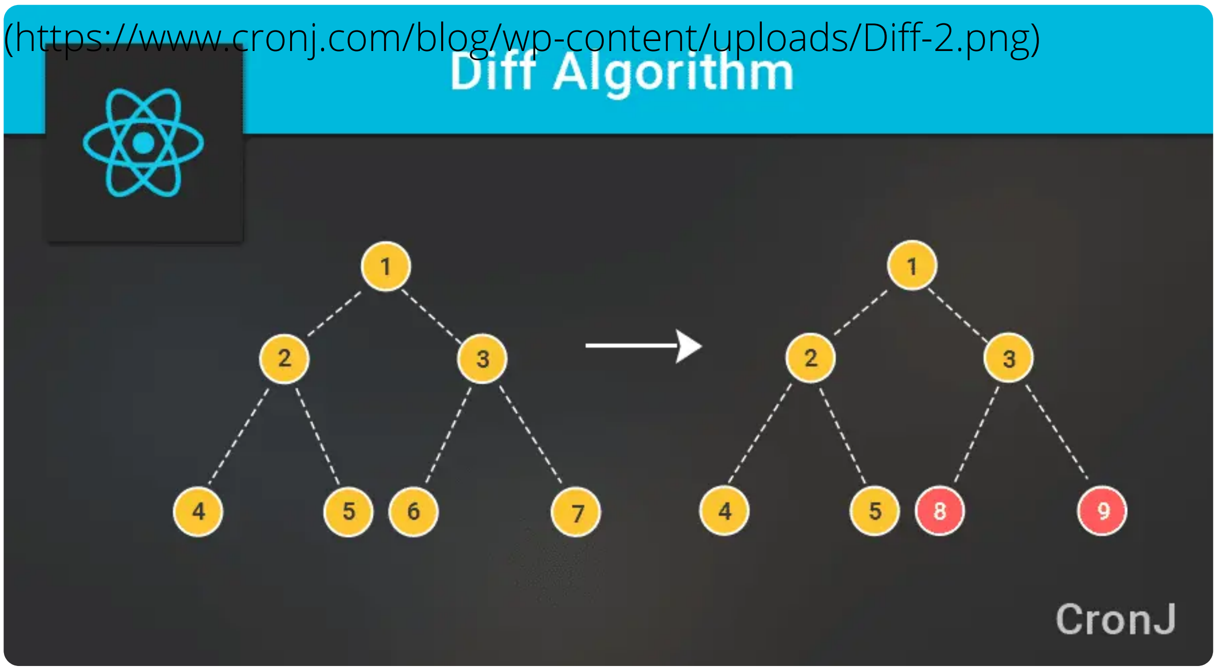
Blog Post

(https://www.cronj.com/blog/wp-content/uploads/Diff-2.png)



HTML FRONTEND (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/HTML/), REACT.JS (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/REACT-JS/), WEB DESIGN AND UI DESIGN (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/WEB-DESIGN/)

How Diff Algorithm is implemented in Reactjs?

(https://www.cronj.com/blog/author/cronj/) CronJ (https://www.cronj.com/blog/author/cronj/), 3 years ago (https://www.cronj.com/blog/diff-algorithm-implemented-reactjs/)

4 min 11363

Just Now:

- FASTag: A Complete Guide 24 Mar (https://www.cronj.com/blog/fastag-digital-toll-

Listen to Post **What is the diff algorithm?** As evident from its name, the main work of a diff algorithm is to find a heuristic to change anything from a state to another. Let's say there is a text A and with the minimal number of steps, it has to be changed to text B. The basic idea is to find a 'modification script' that will turn Text A into Text B. The scripts includes insertion and deletion. Usually, the minimal number of changes is required, but some application might also have to weigh the operations depending on how much text each one affects, or other factors. **Why is it needed to React?** The first time we render a react component, a tree of all the elements is made(point A). On the next update of any state or prop element, the render() function is called again to update a different set of react elements(point B), what react needs is to identify the fattest/ minimal utilization of resources to react from point A to point B. The general solution to this problem has a complexity in the order of O(n3), where n is the number of elements in the tree. **How React approached this problem?** If we used the above approach in React, displaying 1000 elements would require in the order of one billion comparisons. This is far too exhausting and time taking. Instead, React implements a heuristic O(n) algorithm based on two assumptions:

1. Two elements of different types will produce different trees.
2. The developer can hint at which child elements may be stable across different renders with a key prop.

1. **Different root element**

If the root element is different , it completely tears down the old tree and begins from the root of the new tree Example:

Point A	Point B
<p><MyComponent></p>	<span><MyComponent></span>

In the above case, it will unmount(componentWillUnmount) point A and mount (componentWillMount followed by componentDidMount) point B.

2. **Same root element**

If the root element is the same, then the algorithm compares each and every difference in attributes keeps the same valued attributes and changes only the new/differed attributes.

Point A	Point B
<div className="component-1" style="color:red;background-color:black;font-weight:bold;">test</div>	<div className="component-2" style="color:black;background-color:blue;font-weight:bold;">test</div>

1. By comparing these two elements, React knows to only modify the className on Point 2.
2. When updating style, React also knows to update only the properties that changed.

3. **All the Elements are Of The Same Type**

When a component updates, the instance stays the same so that state is maintained across renders. React updates the props of the underlying component instance to match the new element, and calls componentWillReceiveProps() and componentWillUpdate() on the underlying instance. Next, the render() method is called and the diff algorithm recurses on the previous result and the new result.

4. **Recursion on Child elements**

React generates a mutation comparing each and every element of the previous and the new list.

Point A	Point B
<ul> <li>a</li> <li>b</li> </ul>	<ul> <li>a</li> <li>b</li> <li>c</li> </ul>

React will match the two <li>a</li> trees, match the two <li>b</li>trees, and then insert the <li>c</li> tree.On the other hand, inserting an element at the top performs badly. For example, converting between these two trees works badly:

Point A	Point B
<ul> <li>a</li> <li>b</li> </ul>	<ul> <li>c</li> <li>a</li> <li>b</li> </ul>

React will mutate every child instead of realizing it can keep the Point A subtrees intact. This inefficiency can be a problem.

5. **Use of key:**

To solve the above problem, React came out with the 'key' attribute.

Point A	Point B
<ul> <li key='1'>b</li> <li key='2'>c</li> </ul>	<ul> <li key='3'>c</li> <li key='1'>a</li> <li key='2'>b</li> </ul>

In the above scenario React compare the two list and does not change the attributes having the same key value.(The key needs to be unique) Looking for ReactJS Development Company (https://www.cronj.com/reactjs.html), Hire our dedicated developers!

- Color space Image Processing (Explained with RGB, CMY, HSI, Color Models) 2020 (https://www.cronj.com/blog/color-space-image-processing/)
- ReactJs Pagination: How to Page Your Data With ReactJs Pagination? (https://www.cronj.com/blog/reactjs-pagination/)
- HTML to PDF Javascript: Export HTML/CSS To PDF Using Javascript (https://www.cronj.com/blog/export-htmlcss-pdf-using-javascript/)
- How to upload, save Image to Node.js and Express.js using javascript? (https://www.cronj.com/blog/upload-image-nodejs-expressjs-using-javascript/)
- How to Develop Chat System Design like Facebook Messenger | Whatsapp (https://www.cronj.com/blog/how-to-develop-chat-system-design-like-facebook-

messenger/)

- Browser push notifications using JavaScript (https://www.cronj.com/blog/browser-push-notifications-using-javascript/)
- What is hospital management system? features, modules | uses -blog (https://www.cronj.com/blog/what-is-hospital-management-system-features-modules-uses-blog/)
- How Diff Algorithm is implemented in Reactjs? (https://www.cronj.com/blog/diff-algorithm-implemented-reactjs/)

TAGS #DIFF ALGORITHM (HTTPS://WWW.CRONJ.COM/BLOG/TAG/DIFF-ALGORITHM/)

#HIRE REACTJS DEVELOPER (HTTPS://WWW.CRONJ.COM/BLOG/TAG/HIRE-REACTJS-DEVELOPER/)

#REACTJS DEVELOPERS (HTTPS://WWW.CRONJ.COM/BLOG/TAG/REACTJS-DEVELOPMENT-COMPANY/)

CUSTOM SOFTWARE DEVELOPMENT COMPANY IN BANGALORE (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/CUSTOM-DEVELOPMENT/)

/www.cronj.com/blog/video-
lytics-object-extraction/)

Video Analytics - Object Extraction by background subtraction and noise filtering (https://www.cronj.com/blog/video-analytics-object-extraction/)

BANGALORE
(HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/CUSTOM-DEVELOPMENT/), HTML FRONTEND
(HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/HTML/),
REACT.JS
(HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/REACT-JS/)

(https://www.cronj.com/
immurability-is-so-imp

CronJ

(https://www.cronj.com/blog/author/cronj/)

CronJ is a product development company which solves problems in Video analytics, IoT, Industrial automation, Manufacturing 4.0, Smart Factory, Smart City, Blockchain, Artificial Intelligence, Mobility Solutions and supply chain consulting.

Related posts

ANGULAR.JS (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/ANGULAR-JS-EN/), ANGULAR.JS DEVELOPMENT SERVICES (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/ANGULAR-JS/), BUSINESS (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/BUSINESS/), CUSTOM SOFTWARE DEVELOPMENT COMPANY IN BANGALORE (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/CUSTOM-DEVELOPMENT/), DESIGN (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/DESIGN/), MOBILE (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/MOBILE/), MOBILE APPLICATIONS DEVELOPMENT (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/MOBILE-APPLICATIONS/), WEB APPLICATION (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/WEB-APPLICATION/), WEB DESIGN AND UI DESIGN (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/WEB-DESIGN/), WEB DESIGN AND UX DESIGN (HTTPS://WWW.CRONJ.COM/BLOG/CATEGORY/DESIGN/WEB-DESIGN-AND-UX-DESIGN/)

Ecommerce Web Development Agency – 8 Things to Note before Hiring One

(https://www.cronj.com/blog/ecommerce-web-development-agency-8-things-to-note-before-hiring/)

Nupur Pal (https://www.cronj.com/blog/author/nupur/), 5 months ago (https://www.cronj.com/blog/ecommerce-web-development-agency-8-things-to-note-before-hiring/)

5 min

ANGULAR.JS DEVELOPMENT SERVICES (HTTPS://WWW.CRONJ.COM/BLOG/CATEGO JS/), CUSTOM SOFTWARE DEVELOPMENT C BANGALORE (HTTPS://WWW.CRONJ.COM/BLOG/CATEGO DEVELOPMENT/), MOBILE APPLICATIONS D (HTTPS://WWW.CRONJ.COM/BLOG/CATEGO APPLICATIONS/), NODE JS - BACK END (HTTPS://WWW.CRONJ.COM/BLOG/CATEGO BACK-END/), NODE.JS (HTTPS://WWW.CRONJ.COM/BLOG/CATEGO WEB-DEVELOPMENT/), WEB DESIGN AND U (HTTPS://WWW.CRONJ.COM/BLOG/CATEGO DESIGN/)

json2CSV in POST data using Node.js Angular.js

(https://www.cronj.com/blog/json2cs in-post-data-using-node-js-and-angu js/)

CronJ (https://www.cronj.com/blog/author/cronj/), 6 years ago (https://www.cronj.com/blog/json2csv-in-post-data-using-node-js-and-angular-js/)

1 min