link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=73 sentences=132, words=1196

1. react

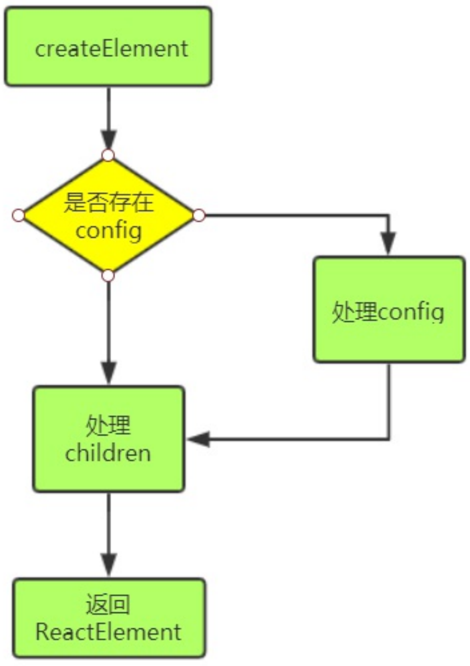
1.1 index.js

src\index.js

```
import React, { Component } from 'react';
import ReactDOM from 'react-dom';
import { mapChildren } from './react/ReactChildren';

class Child extends Component {
    render() {
        console.log('this.props.children', this.props.children);
        const mappedChildren = mapChildren(this.props.children, (c, index) => [{c}, {c}])
        console.log('mappedChildren', mappedChildren)
        return {mappedChildren};
    }
}
class App extends Component {
    render() {
        return (

                    child1

                child2
                child3
                {[
                    child4,
                    child5,
                    child6,
                ]}

        )
    }
}

ReactDOM.render(

    , document.getElementById('root'));
```

1.2 react\ReactElement.js

react\ReactElement.js



```
import ReactCurrentOwner from './ReactCurrentOwner';

const RESERVED_PROPS = {
    key: true,
    ref: true,
    __self: true
};
function hasValidRef(config) {
    return config.ref !== undefined;
}

function hasValidKey(config) {
    return config.key !== undefined;
}
export function createElement(type, config, children) {
    let propName;

    const props = {};
    let key = null;
    let ref = null;
    let self = null;
    let source = null;

    if (config != null) {

        if (hasValidRef(config)) {
            ref = config.ref;
        }
        if (hasValidKey(config)) {
            key = '' + config.key;
        }

        self = config.__self === undefined ? null : config.__self;
        source = config.__source === undefined ? null : config.__source;

        for (propName in config) {
            if (!RESERVED_PROPS.hasOwnProperty(propName)) {
                props[propName] = config[propName];
            }
        }
    }

    const childrenLength = arguments.length - 2;
    if (childrenLength === 1) {
        props.children = children;
    } else if (childrenLength > 1) {
        const childArray = Array(childrenLength);
        for (let i = 0; i < childrenLength; i++) {
            childArray[i] = arguments[i + 2];
        }
        props.children = childArray;
    }

    if (type && type.defaultProps) {
        const defaultProps = type.defaultProps;
        for (propName in defaultProps) {
            if (props[propName] === undefined) {
                props[propName] = defaultProps[propName];
            }
        }
    }
    return ReactElement(
        type,
        key,
        ref,
        self,
        source,
        ReactCurrentOwner.current,
        props,
    );
}

const ReactElement = function (type, key, ref, self, source, owner, props) {
    const element = {

        $typeof: REACT_ELEMENT_TYPE,

        type: type,
        key: key,
        ref: ref,
        props: props,

        _owner: owner,
    };
    return element;
};

export function isValidElement(object) {
    return (
        typeof object === 'object' &&
        object !== null &&
        object.$typeof === REACT_ELEMENT_TYPE
    );
}

export function cloneAndReplaceKey(oldElement, newKey) {
    const newElement = ReactElement(
        oldElement.type,
        newKey,
        oldElement.ref,
        oldElement._self,
        oldElement._source,
        oldElement._owner,
        oldElement.props,
    );

    return newElement;
}
```

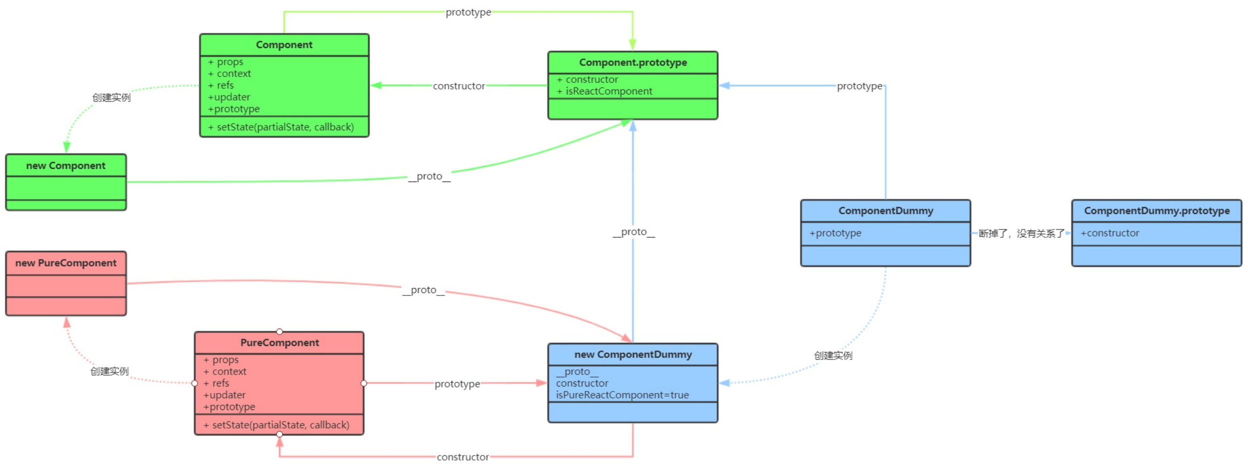
1.3 react\ReactCurrentOwner.js

react\ReactCurrentOwner.js

```
const ReactCurrentOwner = {
    current: null
};

export default ReactCurrentOwner;
```

1.4 react\ReactBaseClasses.js

src\react\ReactBaseClasses.js



```
const emptyObject = {};

function Component(props, context, updater) {
    this.props = props;
    this.context = context;

    this.refs = emptyObject;

    this.updater = updater;
}

Component.prototype.setState = function (partialState, callback) {
    this.updater.enqueueSetState(this, partialState, callback, 'setState');
};

Component.prototype.isReactComponent = {};

function ComponentDummy() { }
ComponentDummy.prototype = Component.prototype;

function PureComponent(props, context, updater) {
    this.props = props;
    this.context = context;
    this.refs = emptyObject;
    this.updater = updater;
}

const pureComponentPrototype = (PureComponent.prototype = new ComponentDummy());
pureComponentPrototype.constructor = PureComponent;

Object.assign(pureComponentPrototype, Component.prototype);
pureComponentPrototype.isPureReactComponent = true;

export { Component, PureComponent };
```

src\shared\ReactSymbols.js

```
const hasSymbol = typeof Symbol === 'function' && Symbol.for;
export const REACT_ELEMENT_TYPE = hasSymbol
    ? Symbol.for('react.element')
    : 0xeac7;
export const REACT_FORWARD_REF_TYPE = hasSymbol
    ? Symbol.for('react.forward_ref')
    : 0xead0;
```

1.6 react\ReactChildren.js

src\react\ReactChildren.js

```
import { isValidElement, cloneAndReplaceKey } from './ReactElement';
import { REACT_ELEMENT_TYPE } from '../shared/ReactSymbols';
const POOL_SIZE = 10;
const traverseContextPool = [];
const SEPARATOR = '.';
const SUBSEPARATOR = ':';

const userProvidedKeyEscapeRegex = /\/+/g;
function escapeUserProvidedKey(text) {
    return ('' + text).replace(userProvidedKeyEscapeRegex, '{{content}}amp;/');
}

function escape(key) {
    const escapeRegex = /[=:]/g;
    const escaperLookup = {
        '=': '=0',
        ':': '=2',
    };
    const escapedString = ('' + key).replace(escapeRegex, function (match) {
        return escaperLookup[match];
    });

    return '{{content}}#x27; + escapedString;
}
function getPooledTraverseContext(
    mapResult,
    keyPrefix,
    mapFunction,
    mapContext,
) {
    if (traverseContextPool.length) {
        const traverseContext = traverseContextPool.pop();
        traverseContext.result = mapResult;
        traverseContext.keyPrefix = keyPrefix;
        traverseContext.func = mapFunction;
        traverseContext.context = mapContext;
        traverseContext.count = 0;
        return traverseContext;
    } else {
        return {
            result: mapResult,
            keyPrefix: keyPrefix,
            func: mapFunction,
            context: mapContext,
            count: 0,
        };
    }
}

function mapChildren(children, func, context) {
    debugger;
    if (children == null) {
        return children;
    }
    const result = [];
    mapIntoWithKeyPrefixInternal(children, result, null, func, context);
    return result;
}

function mapIntoWithKeyPrefixInternal(children, array, prefix, func, context) {
    let escapedPrefix = '';
    if (prefix != null) {
        escapedPrefix = escapeUserProvidedKey(prefix) + '/';
    }
    const traverseContext = getPooledTraverseContext(
        array,
        escapedPrefix,
        func,
        context,
    );
    traverseAllChildren(children, mapSingleChildIntoContext, traverseContext);
    releaseTraverseContext(traverseContext);
}

function mapSingleChildIntoContext(bookKeeping, child, childKey) {
    const { result, keyPrefix, func, context } = bookKeeping;

    let mappedChild = func.call(context, child, bookKeeping.count++);

    if (Array.isArray(mappedChild)) {
        mapIntoWithKeyPrefixInternal(mappedChild, result, childKey, c => c);
    } else if (mappedChild != null) {
        if (isValidElement(mappedChild)) {
            mappedChild = cloneAndReplaceKey(
                mappedChild,
                keyPrefix +
                (mappedChild.key && (!child || child.key !== mappedChild.key)
                    ? escapeUserProvidedKey(mappedChild.key) + '/'
                    : '') +
                childKey,
            );
        }
        result.push(mappedChild);
    }
}

function releaseTraverseContext(traverseContext) {
    traverseContext.result = null;
    traverseContext.keyPrefix = null;
    traverseContext.func = null;
    traverseContext.context = null;
    traverseContext.count = 0;
    if (traverseContextPool.length < POOL_SIZE) {
        traverseContextPool.push(traverseContext);
    }
}

function traverseAllChildren(children, callback, traverseContext) {
    if (children == null) {
        return 0;
    }
    return traverseAllChildrenImpl(children, '', callback, traverseContext);
}

function getComponentKey(component, index) {
    if (
        typeof component === 'object' &&
        component !== null &&
        component.key != null
    ) {
        return escape(component.key);
    }
    return index.toString(36);
}

function traverseAllChildrenImpl(
    children,
    nameSoFar,
    callback,
    traverseContext,
) {
    const type = typeof children;

    if (type === 'undefined' || type === 'boolean') {
        children = null;
    }

    let invokeCallback = false;

    if (children === null) {
        invokeCallback = true;
    } else {
        switch (type) {
            case 'string':
            case 'number':
                invokeCallback = true;
                break;
            case 'object':
                switch (children.$typeof) {
                    case REACT_ELEMENT_TYPE:
                        invokeCallback = true;
                }
        }
    }
    if (invokeCallback) {
        callback(
            traverseContext,
            children,
            nameSoFar === '' ? SEPARATOR + getComponentKey(children, 0) : nameSoFar,
        );
        return 1;
    }

    let child;
    let nextName;
    let subtreeCount = 0;
    const nextNamePrefix =
        nameSoFar === '' ? SEPARATOR : nameSoFar + SUBSEPARATOR;

    if (Array.isArray(children)) {
        for (let i = 0; i < children.length; i++) {
            child = children[i];
            nextName = nextNamePrefix + getComponentKey(child, i);
            subtreeCount += traverseAllChildrenImpl(
                child,
                nextName,
                callback,
                traverseContext,
            );
        }
    }
    return subtreeCount;
}

export {
    mapChildren
}
```