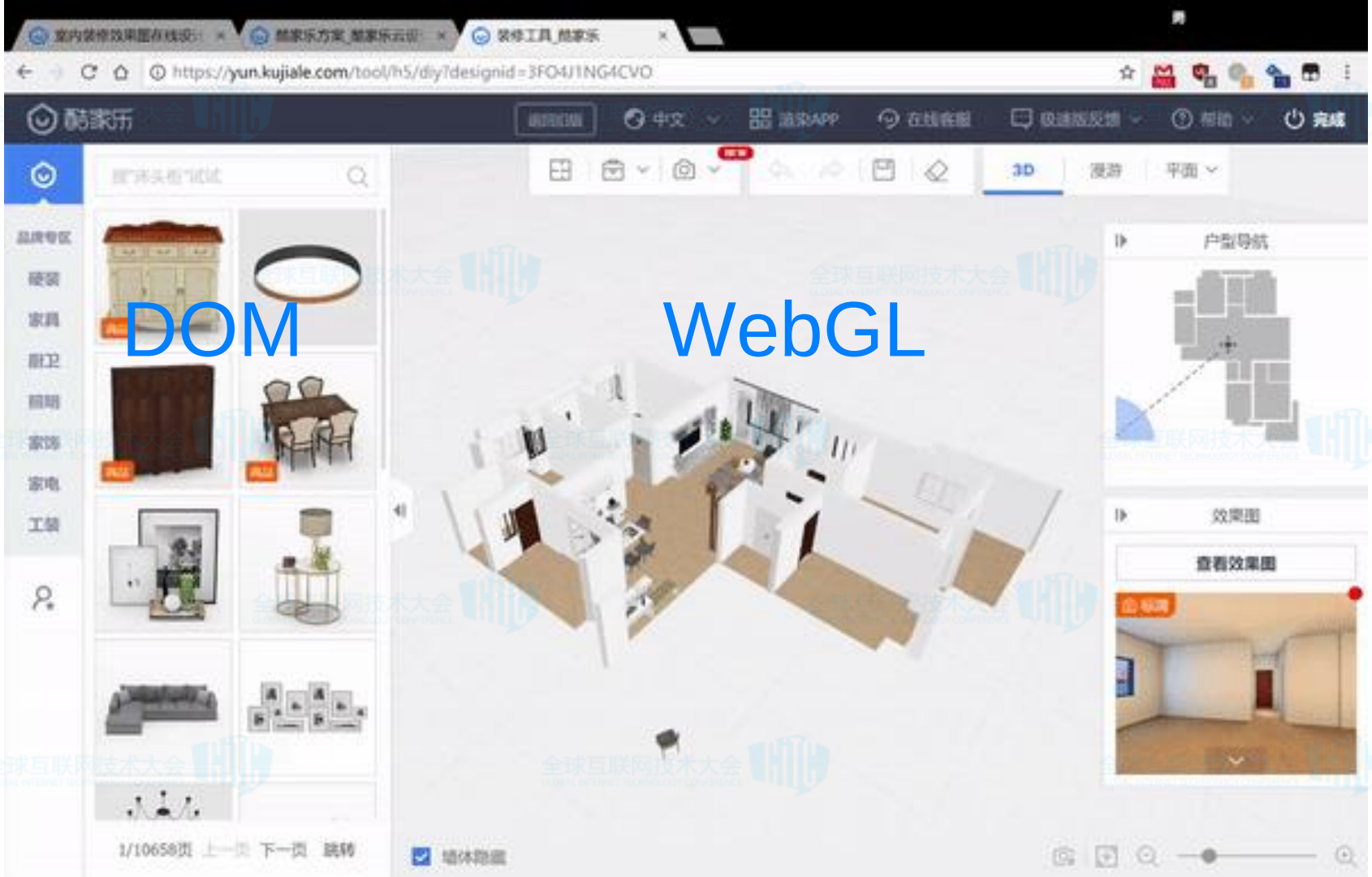
Virtual DOM在3D渲染中的应用

类ReactJS库的实现及3D应用

黄勇


https://yun.kujiale.com/tool/h5/diy?designid=3FO4J1NG4CVO
酷家乐
中文
3D
漫游
平面
户型导航
效果图
查看效果图
DOM
WebGL
1/10658页 上一页 下一页 跳转
墙体隐藏

黄勇

2006~2012: 武汉大学

5年前端开发经验

2016年8月加入酷家乐，花名佚名

HTML5工具前端负责人

设计并实现HTML5版本的家装设计工具

普通的3D程序

```
var geometry = new THREE.BoxGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( { color:

var cube = new THREE.Mesh( geometry, material ); //

scene.add( cube ); // add

camera.position.z = 5;

var animate = function () {
        requestAnimationFrame( animate );

        cube.rotation.x += 0.1; // update
        cube.rotation.y += 0.1;

        renderer.render(scene, camera);
};
```

不便之处

- 命令式：手动的创建、添加和更新
- 组合性差

ReactJS

- 声明式
- 良好的可组合性

DOM

```
render() {
    const { t } = this.props;

    return (
        <div className="tool-box">
            <div className="tool-link" data-km
                <Icon symbol="quanwuyingzhuang
                <span className="tag new">New<
                <span className="title">{t!('t
            </div>
            <div className="tool-link" data-km
                <Icon symbol="quanwudingzhi"/>
                <span className="tag new">New<
                <span className="title">{t!('t
            </div>
            <div className="tool-link" data-km
                <Icon symbol="dimian"/>
                <span className="tag beta">Beta
                <span className="title">{t!('t
            </div>
```

WebGL

```
render() {
    const { props: { model, isSelected, isH

    return (
        <Group>
            <Mesh
                key={model.id}
                model={model}
                holes={holes}
                geomVersion={model.geomVers
                transparent={model.transpa
                visible={visible} />
            {(isSelected || isHint) && <Wi
        </Group>
    );
}
```

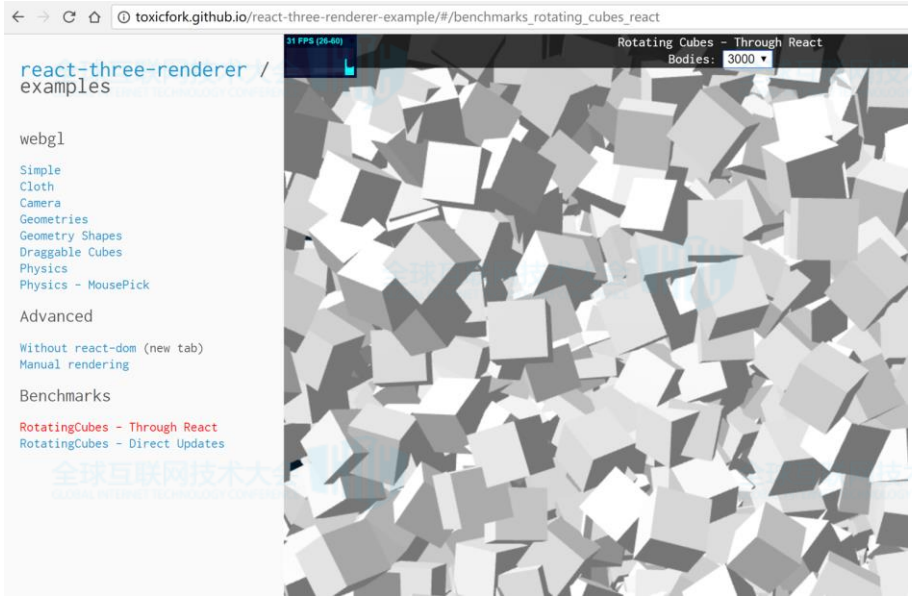
ReactJS？

```
render() {
    const { props } = this;
    if (!props.level) {
        return null;
    }
    return (
        <Group>
            <Dimension />
            <Walls />
            <Holes />
            <Pillars />
            <Rooms />
            <Furnitures />
            <SurfaceBounds />
            <Moldings />
            <Customs />
            <Decorations />
            <Beams />
            <Carriage />
            <Lights />
        </Group>
    );
}
```

- render()只能返回一个节点
- 多renderer不能共存
- 用 来包裹自定义组件？

ReactJS ?

With React: 30FPS



Without React: 60FPS

我们的解决方案

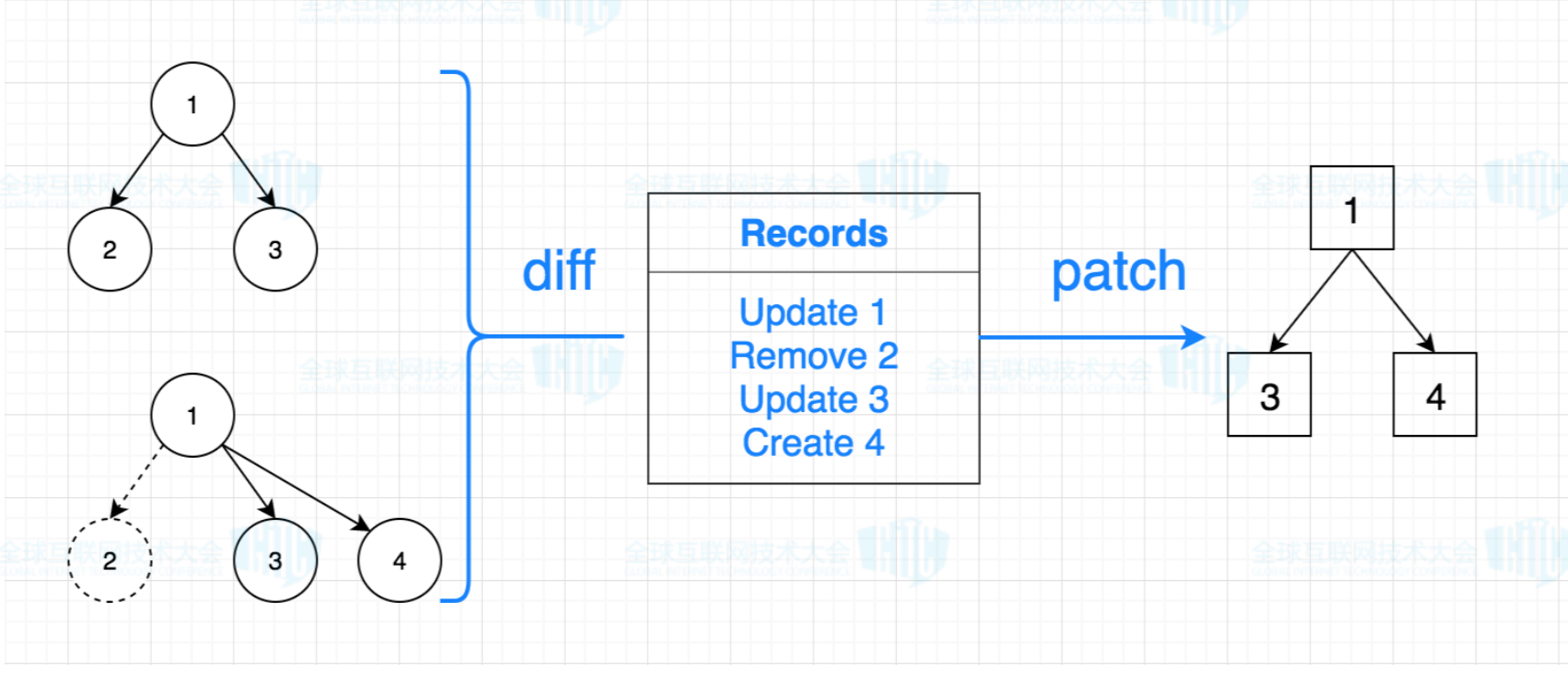
vdom + threejs

我们的解决方案

vdom + threejs

类ReactJS的实现——vdom

create, diff, patch



类ReactJS的实现——vdom

```
function doPatch(newNode?: VNode, oldNode?: VNode, context?: ComponentContext, forceUpdate?: boolean) {
    if (newNode === oldNode || (!newNode && !oldNode)) {
        return;
    }

    if (!oldNode) {
        doCreate(newNode!, context!);
    } else if (!newNode) {
        doRemove(oldNode);
    } else if (oldNode.type !== newNode.type || oldNode.props.key !== newNode.props.key) {
        doRemove(oldNode);
        doCreate(newNode, context!);
    } else {
        doUpdate(newNode, oldNode, context!, forceUpdate);
    }
}
```

类ReactJS的实现——vdom

```
function create(node: VNode, context: ComponentContext) {
    const componentInstance = new node.type(node.props, context);
    node.instance = componentInstance;
    componentInstance.vnode = node;
    componentInstance.context = context;
    const parent = node.parent;
    if (parent) {
        componentInstance.parent = parent.instance;
    }
    if (node.props.ref && node.owner) {
        node.owner.refs[node.props.ref] = componentInstance;
    }
    tryCall(() => componentInstance.componentWillMount());
    expandChildren(node, componentInstance);
    // mount children first, then parent
    patchChildren(node.children, [], getChildContext(componentInstance, context));
    // the children are ready
    tryCall(() => componentInstance.componentDidMount());
}
```

类ReactJS的实现——vdom

```
function update(newNode: VNode, oldNode: VNode, context: ComponentContext, forceUpdate?: boolean) {
    const newProps = newNode.props;
    const componentInstance = oldNode.instance;
    if (!componentInstance) {
        throw new Error(`cannot find the attached instance for node ${oldNode.type.name}`);
    }
    tryCall(() => componentInstance.componentWillReceiveProps(newProps));
    const shouldUpdate = !!forceUpdate ||
        tryCall(() => componentInstance.shouldComponentUpdate(newProps, componentInstance.state));
    const preProps = componentInstance.props;
    componentInstance.context = context;
    // copy the contexture properties from old node
    newNode.instance = componentInstance;
    newNode.owner = oldNode.owner;
    componentInstance.vnode = newNode;
    if (!shouldUpdate) {
        // if no need to recreate children, just use the old ones
        newNode.children = oldNode.children;
        componentInstance.props = newProps;
        setParent(newNode, newNode.children);
        return;
    }
    tryCall(() => componentInstance.componentWillUpdate(newProps));
    componentInstance.props = newProps;
    expandChildren(newNode, componentInstance);
    patchChildren(newNode.children, oldNode.children, getChildContext(componentInstance, context));
    tryCall(() => componentInstance.componentDidUpdate(preProps));
}
```

类ReactJS的实现——vdom

```
function remove(node: VNode) {
    const componentInstance = node.instance;
    if (!componentInstance) {
        throw new Error(`the component instance for node ${node.type.name} has not been created and we're trying to remove it.`)
    }
    patchChildren([], node.children, {});
    tryCall(() => componentInstance.componentWillUnmount());
    if (node.props.ref && node.owner) {
        delete node.owner.refs[node.props.ref];
    }
}
```

我们的解决方案

vdom + threejs

副作用

ThreeComponent

```
insertMesh() {
    if (!this.mesh) {
        console.warn(`${this.constructor.name} is a primitive but doesn't build a mesh. Mayb
        return;
    }
    if (this.autoInsert) {
        this.mesh.name = this.constructor.name;
        this.getLatestParentMesh().add(this.mesh);
    }
}

componentWillMount() {
    const mesh = this.buildMesh();
    this.mesh = mesh as THREE.Object3D;
    this.insertMesh();
}

componentWillUnmount() {
    super.componentWillUnmount();
    const { mesh } = this;
    if (mesh) {
        dispose(mesh);
        if (mesh.parent) {
            mesh.parent.remove(mesh);
        }
    }
}

buildMesh(): THREE.Object3D | Promise<THREE.Object3D> {
    return new THREE.Group();
}
```

DOM

```
render() {
    const { t } = this.props;

    return (
        <div className="tool-box">
            <div className="tool-link" data-km
                <Icon symbol="quanwuyingzhuang
                <span className="tag new">New<
                <span className="title">{t!('t
            </div>
            <div className="tool-link" data-km
                <Icon symbol="quanwudingzhi"/>
                <span className="tag new">New<
                <span className="title">{t!('t
            </div>
            <div className="tool-link" data-km
                <Icon symbol="dimian"/>
                <span className="tag beta">Beta
                <span className="title">{t!('t
            </div>
```

WebGL

```
render() {
    const { props: { model, isSelected, isH

    return (
        <Group>
            <Mesh
                key={model.id}
                model={model}
                holes={holes}
                geomVersion={model.geomVers
                transparent={model.transpa
                visible={visible} />
            {(isSelected || isHint) && <Wi
        </Group>
    );
}
```

回顾

- 使用virtual dom来描述视图结构
- 利用组件的副作用修改threejs对象，驱动WebGL

2D！

- 使用virtual dom来描述视图结构
- 利用组件的副作用修改pixijs对象，驱动2D Canvas

PixiComponent

```
componentWillMount() {
    const pixiObject = this.buildView();
    this.view = pixiObject as PIXI.Container;
    this.insertView();
}

componentDidMount() {
    super.componentDidMount();
    this.isMounted = true;
}

componentWillUnmount() {
    super.componentWillUnmount();
    if (this.view) {
        if (this.pixiParent) {
            this.pixiParent.removeChildView(this);
        }
    }
    this.isMounted = false;
}

buildView(): PIXI.Container {
    return new PIXI.Container();
}
```

DOM

```
render() {
    const { t } = this.props;

    return (
        <div className="tool-box">
            <div className="tool-link" data-km
                <Icon symbol="quanwuyingzhuang
                <span className="tag new">New<
                <span className="title">{t!('t
            </div>
            <div className="tool-link" data-km
                <Icon symbol="quanwudingzhi"/>
                <span className="tag new">New<
                <span className="title">{t!('t
            </div>
            <div className="tool-link" data-km
                <Icon symbol="dimian"/>
                <span className="tag beta">Beta
                <span className="title">{t!('t
            </div>
```

WebGL

```
render() {
    const { props: { model, isSelected, isH

    return (
        <Group>
            <Mesh
                key={model.id}
                model={model}
                holes={holes}
                geomVersion={model.geomVers
                transparent={model.transpa
                visible={visible} />
            {(isSelected || isHint) && <Wi
        </Group>
    );
}
```

Canvas

```
render() {
    const { state: { sprite }, props: { mod
    const modelSize = model.modelSize;
    return (
        <Group >
            {sprite &&
                <Sprite
                    sprite={sprite}
                    width={modelSize.x}
                    height={modelSize.y}
                    cursor={isDragging ? 'm
                />
            }
            {(isSelected || isHovered) &&
                <Box width={modelSize.x} he
            }
        </Group >
```

总结

- Virtual DOM库的原理及实现
- vdom + threejs
- vdom + pixijs
- 全局统一的编程体验

Join Us

杭州 | 上海 | 成都 | 广州 | 北京