9-5 render阶段-beginWork

`beginWork`

ReactFiberBeginWork.ts

TypeScript
```typescript
import { isNum, isStr } from "shared/utils";
import { Fiber } from "./ReactInternalTypes";
import { HostComponent, HostRoot } from "./ReactWorkTags";
import { mountChildFibers, reconcileChildFibers } from "./ReactChildFi

// 1. 处理当前fiber，因为不同组件对应的fiber处理方式不同，
// 2. 返回第一个子节点child
export function beginWork(
  current: Fiber | null,
  workInProgress: Fiber
): Fiber | null {
  switch (workInProgress.tag) {
    case HostRoot:
      // 根fiber
      return updateHostRoot(current, workInProgress);
    case HostComponent:
      // 原生标签
      return updateHostComponent(current, workInProgress);
    // todo 其他类型
  }

  throw new Error(
    `Unknown unit of work tag (${workInProgress.tag}). This error is l
      "React. Please file an issue."
  );
}

function updateHostRoot(current: Fiber | null, workInProgress: Fiber)
  const nextChildren = workInProgress.memoizedState.element;
  reconcileChildren(current, workInProgress, nextChildren);
  return workInProgress.child;
}
// 原生标签，div等
function updateHostComponent(current: Fiber | null, workInProgress: Fi
  const { type } = workInProgress;
  let nextChildren = workInProgress.pendingProps.children;
  const isDirectTextChild = shouldSetTextContent(
    type,
    workInProgress.pendingProps
  );

  if (isDirectTextChild) {
    // 文本属性
    nextChildren = null;
    return null;
  }
  reconcileChildren(current, workInProgress, nextChildren);
  return workInProgress.child;
}

// todo
// 页面初次渲染，单个原生标签节点
// 协调
function reconcileChildren(
  current: Fiber | null,
  workInProgress: Fiber,
  nextChildren: any
) {
  console.log(
    "%c [  ]-60",
    "font-size:13px; background:pink; color:#bf2c9f;",
    current
  );
  if (current === null) {
    // 初次渲染
    workInProgress.child = mountChildFibers(workInProgress, null, next
  } else {
    // 更新
    workInProgress.child = reconcileChildFibers(
      workInProgress,
      current.child,
      nextChildren
    );
  }
}

function shouldSetTextContent(type: string, props: any): boolean {
  return (
    type === "textarea" ||
    type === "noscript" ||
    isStr(props.children) ||
    isNum(props.children) ||
    (typeof props.dangerouslySetInnerHTML === "object" &&
      props.dangerouslySetInnerHTML !== null &&
      props.dangerouslySetInnerHTML.__html != null)
  );
}
```

协调

ReactChildFiber.ts

```TypeScript
import { createFiberFromElement } from "./ReactFiber";
import { Fiber } from "./ReactInternalTypes";
import { REACT_ELEMENT_TYPE } from "../../shared/ReactSymbols";
import { Placement } from "./ReactFiberFlags";
import { ReactElement } from "shared/ReactTypes";
```

```
type ChildReconciler = (
  returnFiber: Fiber,
  currentFirstChild: Fiber | null,
  newChild: any
) => Fiber | null;

export const reconcileChildFibers: ChildReconciler =
  createChildReconciler(true);
export const mountChildFibers: ChildReconciler = createChildReconciler

// wrapper function
function createChildReconciler(shouldTrackSideEffects: boolean) {
  function placeSingleChild(newFiber: Fiber) {
    if (shouldTrackSideEffects && newFiber.alternate === null) {
      newFiber.flags |= Placement;
    }
    return newFiber;
  }

  // 协调单个子节点
  function reconcileSingleElement(
    returnFiber: Fiber,
    currentFirstChild: Fiber | null,
    newChild: ReactElement
  ) {
    // 初次渲染
    const created = createFiberFromElement(newChild);
    created.return = returnFiber;
    return created;
    // 新老 vdom diff
  }
  function reconcileChildFibers(
    returnFiber: Fiber,
    currentFirstChild: Fiber | null,
    newChild: any
  ) {
    // 单个节点、数组、文本
    if (typeof newChild === "object" && newChild !== null) {
      switch (newChild.$$typeof) {
        case REACT_ELEMENT_TYPE:
          return placeSingleChild(
            reconcileSingleElement(returnFiber, currentFirstChild, new
          );
        }
      }

      return null;
    }

    // todo 数组、文本
    return reconcileChildFibers;
  }
```