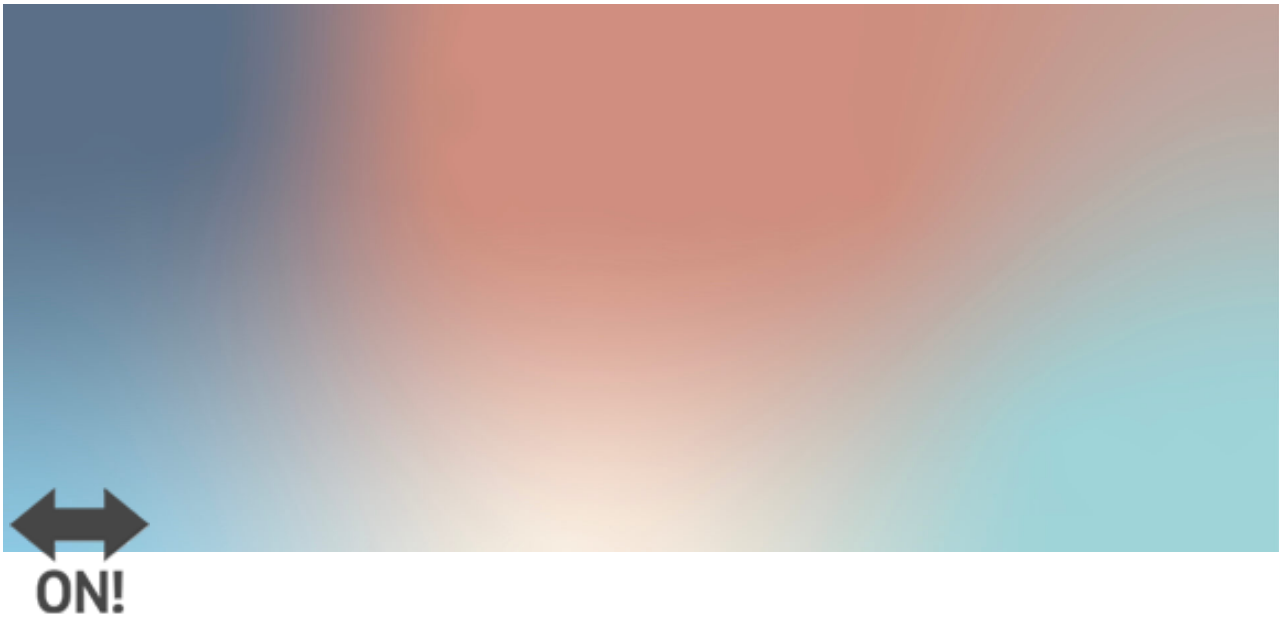

9-3 scheduleUpdateOnFiber 调度更新

页面初次渲染、类组件 setState/forceUpdate、函数组件 setState 都会走到更新，都会调用 `scheduleUpdateOnFiber` 函数。

DebugReact > src > react > packages > react-reconciler > src > JS ReactFiberWorkLoop.js

```
744  export function scheduleUpdateOnFiber(
745    root: FiberRoot,
746    fiber: Fiber,
747    lane: Lane,
748  ) {
```

从根节点开始更新。

scheduleUpdateOnFiber(root, current);

`scheduleUpdateOnFiber`

packages/react-reconciler/src/ReactFiberWorkLoop.ts

TypeScript

```
let workInProgress: Fiber | null = null;
let workInProgressRoot: FiberRoot | null = null;
```

```
// !
export function scheduleUpdateOnFiber(root: FiberRoot, fiber: Fiber) {
  workInProgressRoot = root;
  workInProgress = fiber;

  ensureRootIsScheduled(root);
}

export function performConcurrentWorkOnRoot(root: FiberRoot) {
  // ! 1. render
  renderRootSync(root);
  executionContext;

  const finishedWork: Fiber = root.current.alternate as Fiber;

  // !3. commit
  root.finishedWork = finishedWork;

  commitRoot(root);
}
```

ensureRootIsScheduled

packages/react-reconciler/src/ReactFiberRootScheduler.ts

TypeScript

```
import { performConcurrentWorkOnRoot } from "./ReactFiberWorkLoop";
import { FiberRoot } from "./ReactInternalTypes";
import { NormalPriority } from "scheduler/src/SchedulerPriorities";
import { Scheduler } from "scheduler";

export function ensureRootIsScheduled(root: FiberRoot) {
  queueMicrotask(() => {
    scheduleTaskForRootDuringMicrotask(root);
  });
}

export function scheduleTaskForRootDuringMicrotask(root: FiberRoot) {
  const schedulerPriorityLevel = NormalPriority;
```

```
  Scheduler.scheduleCallback(
    schedulerPriorityLevel,
    performConcurrentWorkOnRoot.bind(null, root)
  );
}
```