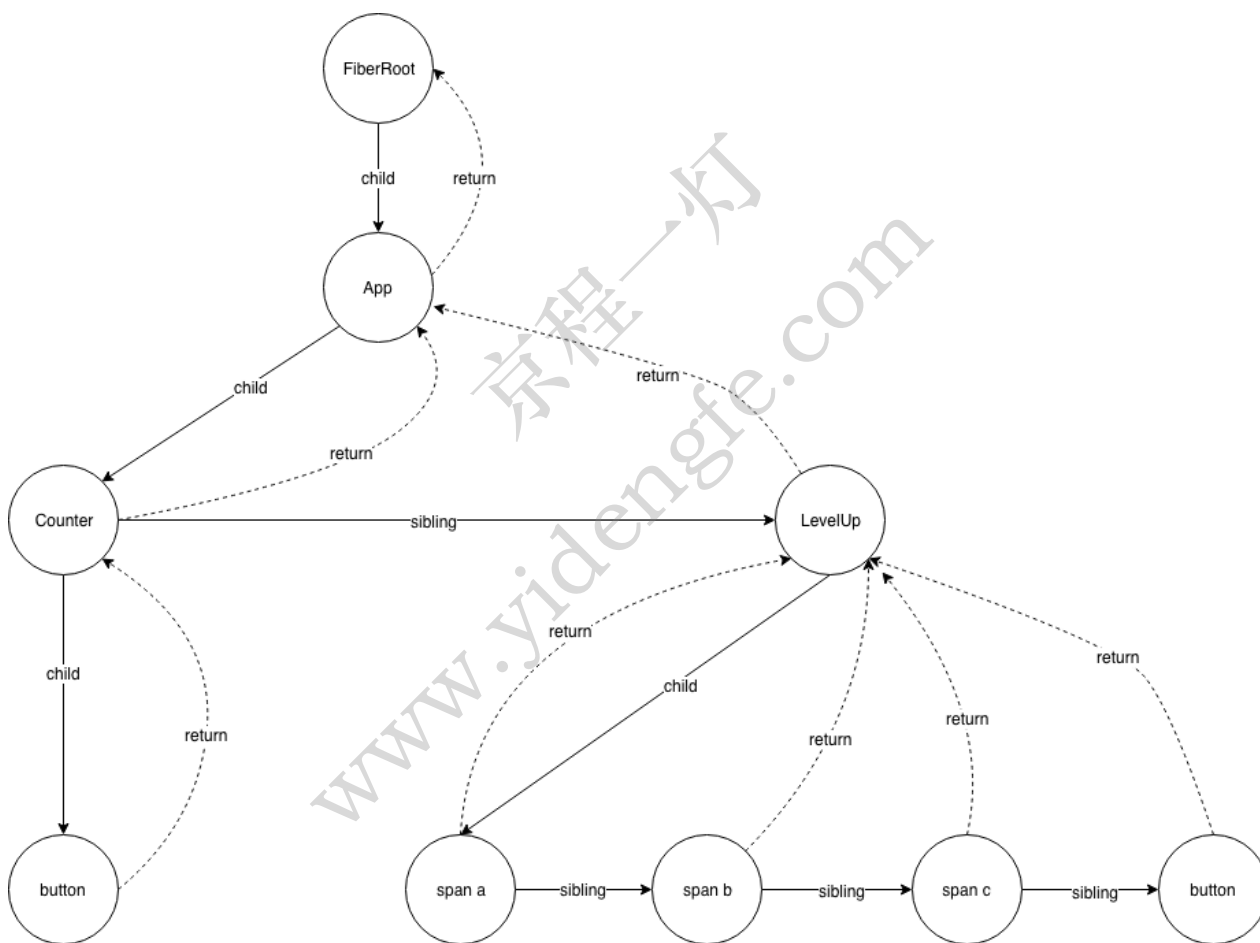


## 🌲 2.React创建Fiber Root

### 2-1.ReactDOM.render

React16新增了hydrate使用服务端渲染和本地DOM进行调和。

legacyRenderSubtreeIntoContainer具体的完成了render任务，通过\_reactRootContainer标识了react节点并完成了fiberRoot创建，ReactSyncRoot来源于,初始化更新的时候采用unbatchedUpdates，需要尽快的完成渲染。



### 2-2.Fiber Root

render->legacyRenderSubtreeIntoContainer-> ReactSyncRoot->FiberRoot 里有一个关键的类型，react-reconciler负责具体Fiberroot的构建,他承载了整个React更新调度全部的数据结构.如上图的深入描述

```
function FiberRootNode(containerInfo, tag, hydrate) {
  // 标记不同的组件类型
  this.tag = tag;
  // 当前应用对应的Fiber对象
  this.current = null;
}
```

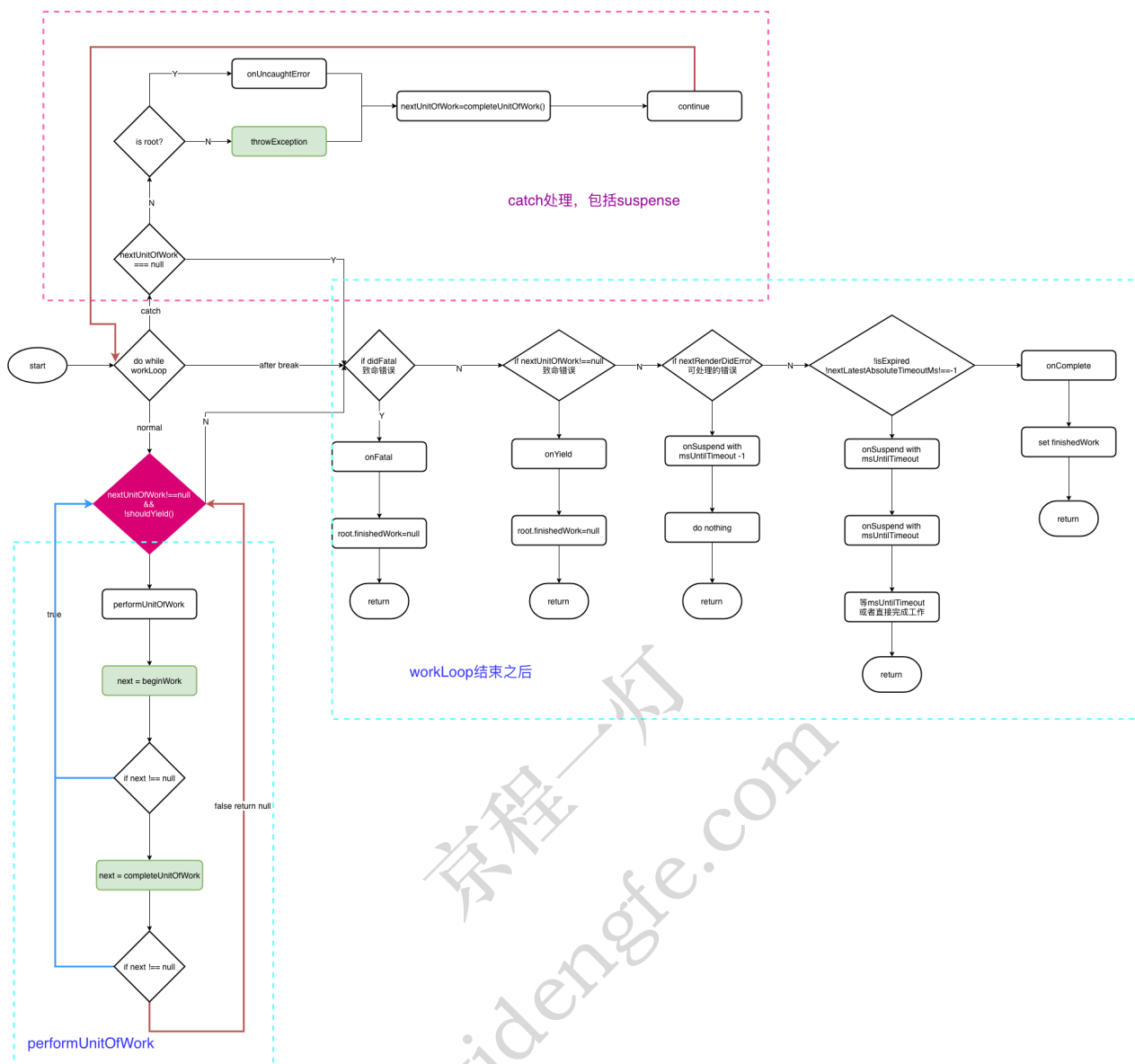
```

// root节点
this.containerInfo = containerInfo;
// 只有在持久更新中会用到
this.pendingChildren = null;
this.pingCache = null;
this.finishedExpirationTime = NoWork;
// 在commit阶段只会处理这个值对应的任务
this.finishedwork = null;
// 在任务被挂起的时候通过setTimeout设置的返回内容，用来下一次如果有新的任务挂起时清理还没触发的timeout
this.timeoutHandle = noTimeout;
}

```

创建完Fiber Root在unbatchedUpdates中执行updateContainer对容器内容进行更新，更新前会先通过expirationTime对节点结算超时时间，具体是通过在ReactFiberWorkLoop中computeExpirationForFiber进行计算，那么我们root循环调用了ReactFiberWorkLoop。任务循环的源码比较简单，是一个while循环，其中workInProgress对象就是一个Fiber Node，循环每渲染完成一个Fiber Node就利用shouldYield来判断是否有优先级更高的任务存在，是则跳出循环先执行优先级更高的任务，否则继续渲染下一个Fiber Node。

实际上渲染过程是需要对比过去已经渲染好的节点树（current）和即将要渲染的新节点树（workInProgress）的差异，利用类似diff的算法来最小化界面需要重新渲染的部分。但是React并没有实现两棵节点树，而是利用FiberNode.alternate属性来存储上一次渲染过的结果。



这个超时时间实现的非常精妙，我们拿computeAsyncExpiration举例子，在computeExpirationBucket中接收的就是currentTime、5000和250最终的公式就是酱紫的：  

$$(((currentTime - 2 + 5000 / 10) / 25) \mid 0) + 1) * 25$$

```
ceiling(
  MAGIC_NUMBER_OFFSET - currentTime + expirationInMs / UNIT_SIZE,
  bucketSizeMs / UNIT_SIZE)

function ceiling(num: number, precision: number): number {
  return (((num / precision) | 0) + 1) * precision;
}
```

翻译一下就是：最终结果是以25为单位向上增加的，比如说我们输入10002 - 10026之间，最终得到的结果都是10525，但是到了10027的到的结果就是10550，这就是除以25取整的效果。

其实一句话  $100 / 25 \mid 0 = 4$ ;  $4 \mid 0 = 0100$  | 0000 => 0100

$101 / 25 \mid 0 = 4$ ;  $4 \mid 0 = 0100$  | 0000 => 0100

React 这么设计抹相当于抹平了25ms内计算过期时间的误差，那他为什么要这么做呢？看到 LOW\_PRIORITY\_BATCH\_SIZE 这个字样，batch，是不是就对应batchedUpdates？再细想了一下，这么做也许是为了让非常相近的两次更新得到相同的expirationTime，然后在一次更新中完成，相当于一个自动的batchedUpdates。神奇不神奇？

## 2-3.事件绑定

React 为了跨平台，对于事件体系的代码做出了一些妥协，采用动态注入的方式让不同的平台对事件核心模块进行插件注入。我们先来看 ReactDOM 的入口文件，里面有这么一句代码：`import './ReactDOMClientInjection'`；具体实现采用了plugin挂载的方式，

```
import ChangeEventPlugin from '../events/ChangeEventPlugin';
```

EventListener是具体的绑定判断，setInitialProperties对外进行了事件绑定。具体的事件绑定进了根节点的dom元素利用冒泡进行了控制。同时整个事件处理位于批处理状态中。

---

志佳老师@2019

京程一灯  
www.yidengle.com