UniqueWeb 2020 夏令营 前端第2期任务

Intro

本期任务主要学习两个重要的内容，一个是Promise/A+规范，一个是前端框架中的重要编程概念Virtual DOM（以下简称vdom）。

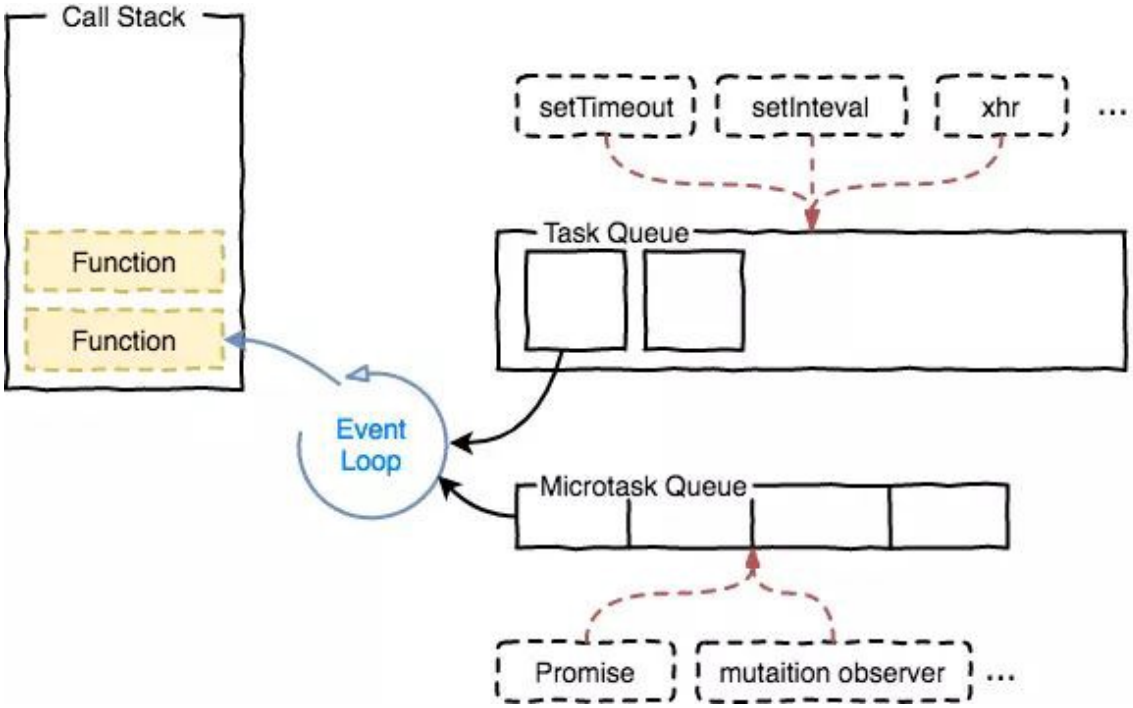
对于前者的学习，能让你更好地理解JavaScript的异步机制，以及相对现代化的异步处理方式；对于后者的学习，能让你在后续对框架的学习中体会更加深刻，同时避免写出一些性能不友好的代码。

开始任务前，请先将本任务说明全部阅读，注意两项任务的时间分配问题。

内容

任务一

1. 学习event loop机制，也即事件循环与任务队列，你的目标是完全看懂下图（网图...）：



2. 在理解上述机制的前提下，阅读Promise/A+规范
3. 手写一个Promise，并通过测试
4. 必须使用class或原型链完成，如果你不理解class的底层实现，建议认真学习继承

任务二

Intro

随着前端逐渐发展，传统的查询DOM、修改DOM使得应用变得很难维护，尤其是面临大量的需要维持引用的变量。同时，使用这种方式会使性能变得不可控。

所以这一次我们安排一期关于vdom的任务，总的来说，是让你们实现一个简单的vdom类库。

使用vdom的原因也很清晰：为了规范化前端开发过程中大量重复的 DOM 操作，使性能能变得可控的同时，尽可能提高语义化和表达能力。

我们以一段简单的html为例。

```
<div class="chat-box">
    <p style="position: relative;" >
        xxxxxx
    </p>
    <ul>
        <li>1</li>
        <li>2</li>
    </ul>
</div>
```

一个vdom至少需要含有三个属性`tag`，`props`，`children`，比如上述的html转为js对象就为：

```
{
    tag: "div",
    props: {
        class: "chat-box"
    },
    children: [
        {
            tag: "p",
            props: {
                style: "position: relative;"
            },
            children: ["xxxxxx"]
        },
        {
            tag: "ul",
            props: {},
            children: [
                {
                    tag: "li",
                    props: {},
                    children: ["1"]
                },
                {
                    tag: "li",
                    props: {},
                    children: ["2"]
                }
            ]
        }
    ]
}
```

任务要求

- 构造，使用大家普遍使用的语法构造 VDOM:

```
function createElement(tagName: string, props: object, ...children: Child[]);
```

上面代码的意思就是，第一个参数是 tagName 的字符串，第二个是标签的 attrs 是一个 JS 对象，第三个是子元素们。
注：对于子元素，为了方便diff，不需要是完全摊平了的数组，例如：

```
createElement(
  "div",
  { className: "big" },
  createElement("span", { className: "big-inner" }, "hehe"),
  [1, 2, 3, 4, 5].map(n =>
    createElement("span", { className: "big-inner-number", key: n }, n)
  ),
  createElement("span", { className: "big-inner" }, "hehe")
); // 无论数组如何变化，都只占用了参数表中一个位置，因此该元素的必然有相同数目的子元素
```

如上，由于没必要在编译时得知数组的长度，直接将整个数组作为一个子元素，而不是展开之后的数组的每个元素作为子元素。

- 渲染

  我们的最终目的是将dom渲染出来，编写一个render函数，将vdom渲染为真实dom节点。

- diff

  对于dom的变更将首先反映在vdom的变化上。diff的目标是为了获取新旧两棵vdom树的差异，使用O(n)的时间复杂度对新旧两棵vdom树进行diff。尤其注意对列表的diff，需要一个特殊的key属性用于标识（也即React和Vue中的key），先使用listdiff来进行列表成员顺序的diff，对每个成员再递归diff。（PS. 请务必理解key在列表diff中的作用，为什么它很重要，以及如果使用数组下标作为key可能会造成什么问题）

- 获取补丁

  补丁是两棵vdom树diff后所得的结果。例如 diff两棵 vdom 树的结果是某个子节点props中的某个attr 发生了变化，补丁就应该能体现该变化，而不包括其他相同的部分而导致额外修改。

- 应用补丁

  将收集到的补丁集合运用到旧的真实dom上，才能使dom更新。请注意，之所以使用diff来获得差异，是为了让不需要变动的dom保持原样，减少dom操作的性能开销，补丁之所以称为补丁，就是在需要修改的地方修改。请不要将整棵vdom重新渲染一遍。

- 事件（选做）

  事件是特殊的attr，关于节点变动造成的一系列副作用，多数情况下可能需要重新绑定事件监听器，这部分可作为选做，可只作一两个事件比如onClick

Tips

1. 两个任务都可以使用class来编写，前提你完全了解js继承
2. 关于Promise和vdom网络上都能找到大量的资料，在参考的同时，切勿抄袭代码
3. vdom任务较为困难，建议分配较多的时间，你可能会陷入对递归debug的泥潭中，如果你能熟练使用浏览器调试，可能会省事很多

4. 部分参考：

   群内后续会放上组内洪志远学长关于vdom的ppt，可供参考

   react reconciliation

   一个相对详细的blog

如果你有时间

考虑到后续会进行为期一星期的项目开发，你可以抽点时间看看以下内容

- async/await等一系列es6内容
- yarn/npm，以及webpack/parcel打包构建工具
- 框架学习，推荐react/vue，团队内项目基本基于这两个框架
- RESTful api规范
- jwk鉴权，oauth三方授权
- sass/less/stylus，简化你的css书写