✅ 第一问：点击方块发送事件

🧠 题目要求

点击 .square 方块时，调用 sendRequest()，发送事件名为 "click"，data 包含背景色的请求。

✅ 解法实现

eventLogger.js

js

CopyEdit

```
import { sendRequest } from "./utils";


export class EventLogger {
  logEvent(eventName, data) {
    const payload = {
      events: [
        {
          eventName,
          hostname: window.location.hostname,
          timestamp: new Date().toISOString(),
          data
        }
      ]
    };
    sendRequest(payload);
  }
}
```

```
const eventLogger = new EventLogger();
export { eventLogger };
```

index.js

```
js
CopyEdit
import { eventLogger } from "./eventLogger";

document.querySelectorAll(".square").forEach((square) => {
  square.addEventListener("click", () => {
    const color = getComputedStyle(square).backgroundColor;
    eventLogger.logEvent("click", { color });
  });
});
```

✅ **第二问：data 中包含方块的 index**

🧠 **题目要求**

在 data 中加入 "index"，表示被点击方块的索引。

✅ **解法修改（只需改 index.js）：**

```
js
CopyEdit
document.querySelectorAll(".square").forEach((square, index) => {
  square.addEventListener("click", () => {
    const color = getComputedStyle(square).backgroundColor;
    eventLogger.logEvent("click", { color, index });
  });
```

});

✅ **第三问：对相同 index 的点击事件进行合并**

🧠 **题目要求**

如果多次点击相同 index，只发送一个请求；也就是说 **只发送最后一次点击每个 index 的事件**。

✅ **解法：**

我们需要做两件事：

1. 缓存点击事件（按 index 去重）
2. 批量发送（例如用 debounce 或 delay 模拟批处理）

方法一：点击时更新缓存，延迟统一发送

eventLogger.js 修改如下：

js

CopyEdit

```
import { sendRequest } from "./utils";


export class EventLogger {
  constructor() {
    this.eventMap = new Map();  // key: index, value: { color, index }
    this.debounceTimer = null;
  }


  logEvent(eventName, data) {
    this.eventMap.set(data.index, data);
```

```
    clearTimeout(this.debounceTimer);
    this.debounceTimer = setTimeout(() => {
      const payload = {
        events: Array.from(this.eventMap.entries()).map(([index, d]) => ({
          eventName,
          hostname: window.location.hostname,
          timestamp: new Date().toISOString(),
          data: d
        }))
      };
      sendRequest(payload);
      this.eventMap.clear();  // 清空缓存
    }, 300); // debounce delay 可根据需要调整
  }
}

const eventLogger = new EventLogger();
export { eventLogger };
```

✅ **第四问：支持 abort 前一个未完成的请求**

🧠 **题目要求**

每次点击后，若有尚未完成的请求，应取消上一次的请求，只保留当前的。

sendRequest 返回一个带 .abort() 方法的 promise。

✅ **解法：**

我们需要记录当前的请求，并在新请求开始前中止上一个。

修改 eventLogger.js：

js

CopyEdit

```
import { sendRequest } from "./utils";

export class EventLogger {
  constructor() {
    this.eventMap = new Map();
    this.debounceTimer = null;
    this.pendingRequest = null;  // 新增字段记录当前请求
  }

  logEvent(eventName, data) {
    this.eventMap.set(data.index, data);

    clearTimeout(this.debounceTimer);
    this.debounceTimer = setTimeout(() => {
      const payload = {
        events: Array.from(this.eventMap.entries()).map(([index, d]) => ({
          eventName,
          hostname: window.location.hostname,
          timestamp: new Date().toISOString(),
          data: d
        }))
      };
```

```
    // 如果上一个请求未完成，先中止它
    if (this.pendingRequest && typeof this.pendingRequest.abort === "function") {
      this.pendingRequest.abort();
    }

    this.pendingRequest = sendRequest(payload);
    this.eventMap.clear();
   }, 300);
  }
}

const eventLogger = new EventLogger();
export { eventLogger };
```

✅ **总结**

问题	核心考察	实现点
第一问	基本事件记录	logEvent + .backgroundColor
第二问	data 增加 index	使用 .forEach((el, index) => ...)
第三问	去重 + 批量发送	Map 缓存、setTimeout 延迟
第四问	请求中止机制	保存并 .abort() 上一个请求

如需我打包成一个完整 HTML 单页 demo 供本地测试，我也可以帮你生成。