基于 NodeJS 和 WebSocket 的在线航海

棋游戏的设计与实现

查俊豪、徐明璨、喻启林
（南京市金陵中学）

目录

查俊豪 0903（组长）：程序设计与实现、玩法设计、论文撰写、幻灯片制作
徐明璨 0933：主玩法设计
喻启林 0937：摸鱼
（南京市金陵中学高二 9 班）

摘要：NodeJS 把 JavaScript 推动为互联网发展核心驱动力，给前端界带来的更大的发展潜力，而 WebSocket 技术为客户端与服务端通讯提供了便捷的解决方案，本文就一个基于 NodeJS 的在线多人游戏的实例来展示 WEB 技术在在线游戏方面的潜力与光明前景。

关键词：NodeJS；JavaScript；HTML5；WebSocket；在线多人游戏；PWA；WEB

1 概述

1.1 课题背景

自 2014 年 10 月 29 日，W3C 官网正式宣布 HTML5 的标准制定完毕，越来越多的开发者把目光聚向了 HTML5。而关于 HTML5 在游戏中的应用也受到了国内外越来越多的个人、社区、公司的关注，相关后端技术的出现也使得多人在线游戏成了可能。

在如今互联网飞速发展的时代，无需下载、即点即玩、跨平台等 HTML 技术的独特优势适应了如今快节奏的发展需求。经过调查，对于小游戏来说，依赖 HTML 作为载体是如今较好的一种解决方案，受开发者和用户青睐。

1.2 系统开发的技术简介

1.2.1 HTML5 相关技术

HTML5 是 HTML 最新的修订版本，2014 年 10 月由万维网联盟（W3C）完成标准制定。其设计目的是在移动设备上支持多媒体，提供了用于绘画的 canvas 元素，用于媒介回放的 video 和 audio 元素，和对本地离线存储的更好的支持。另外，用于控制样式的 CSS3 也带来了众多新特性。

1.2.2 NodeJS

Node.js 是一个基于 Chrome JavaScript 运行时建立的一个平台，为 JavaScript 语言提供了编写后端应用的能力，且因为 JavaScript 天生异步、单线程的特性，尤其适合高并发，IO 密集型应用。

1.2.3 WebSocket

WebSocket 是 HTML5 开始提供的一种在单个 TCP 连接上进行全双工通讯的协议，取

代了原来的 Ajax 轮询等解决方案，它基于 HTTP 协议，通过 header："Upgrade: WebSocket" 建立连接。

1.2.4 MySQL

MySQL 是最流行的关系型数据库管理系统，本系统使用 MySQL 数据库进行数据存储。

1.2.5 Progressive Web APP

PWA（渐进式网络应用）提供了将网页安装到用户设备本地的能力，开发者可以“渐进地”向网站中添加功能，如后台推送，离线使用等功能，是微信小程序等的原型。

2 需求分析

2.1 功能分析

客户端的主要功能：游戏资源预加载、查看公告、获取正在进行的游戏列表、创建新游戏、加入已存在的游戏、游戏内聊天、回放以前的游戏录像；

服务端的主要功能：监听用户请求、推送公告、保存并推送游戏列表、提供多人游戏支持、保存并推送游戏存档。

3 概要分析

3.1 系统总体分析

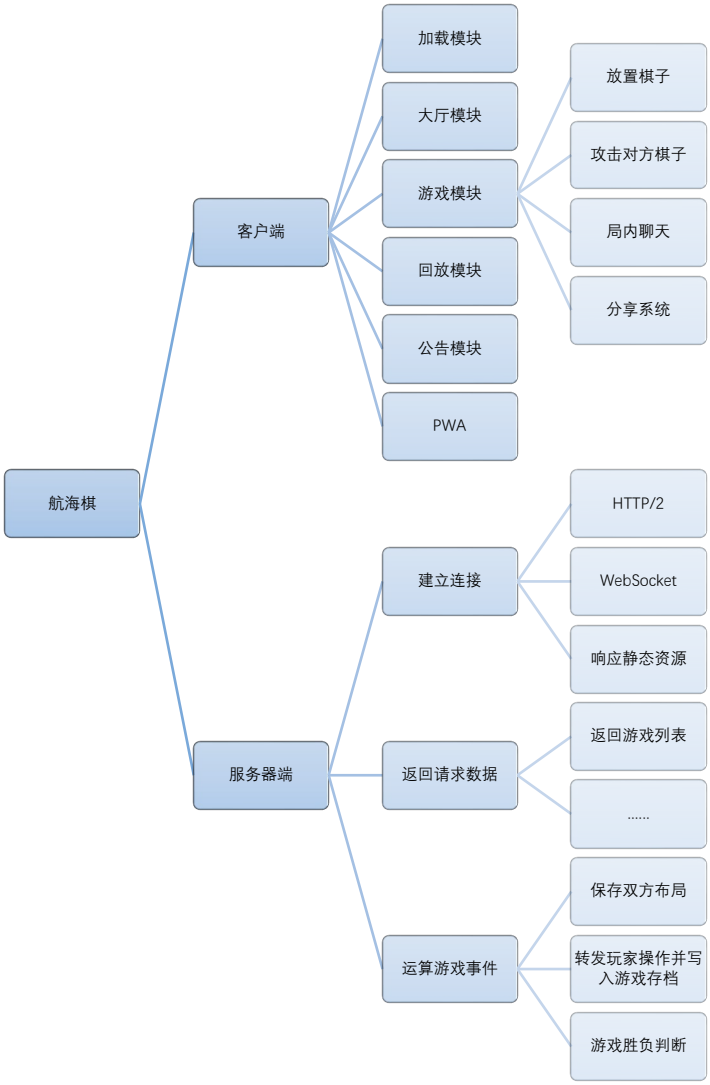
该游戏系统的主要模块有：

1) 加载模块，包括功能：调整游戏基础设置、下载游戏资源并保存到本地、与服务器建立 WebSocket 连接。
2) 大厅模块，包括功能：链接至其他模块、创建新游戏、进入已存在的游戏、查看游戏信息。
3) 游戏模块，包括功能：布局棋子、攻击对方棋子、局内聊天。
4) 回放模块，包括功能：获取、选择、查看、回放、下载过去的游戏存档。
5) 公告模块，包括功能：获取、选择、查看公告。

游戏玩法简介：

玩家初始拥有固定数量的“航母”、“巡洋舰”、“潜艇”，玩家需对己方船只进行合理布局，并合理猜测对手船只位置达到抵御对方的攻击以及击沉对手所有船只的目标。此外，相对于传统航海棋玩法，新增了“合约”系统，“合约”即额外的游戏规则，如扩大地图、增加闪避率，通过不同的合约组合可以使游戏变得更具趣味与挑战。

3.2 功能模块图



4 详细设计

4.1 前端详细实现

4.1.1 加载模块详细实现

使用了 Promise API 包装了 XHR 请求，以达到依次下载资源并显示总进度和单个文件进度，用于下载音频资源和表情资源，避免网络问题导致不流畅的游戏体验。下载完成后对于音频文件会立即创建 audio 对象供未来使用，对于压缩包会调用 JSzip 库进行解压并将所有表情添加到表情选择框。

```
function preload(url, index) {
```

```
    return new Promise(function (resolve, reject) {
        localforage.getItem(url)
            .then(file => {
                let fileOK = function () {
                    if (file.type.indexOf("audio") != -1) {
                        let musicObj_ = new musicObj(URL.createObjectURL(file))
                        audioRes.push(musicObj_)
                    }
                    else if (file.type.indexOf("zip") != -1) {
                        let zip = new JSZip()
                        zip.loadAsync(file).then(function (zip) {
                            zip.forEach(function (relativePath, file) {
                                file.async("blob").then(function (content) {
                                    emoji[relativePath] = content
                                    let emojiE = document.createElement("img")
                                    emojiE.src = URL.createObjectURL(content)
                                    emojiE.className = "emojiListItem"
                                    emojiE.addEventListener("click", () => {
                                        document.getElementById("chatInputText").value +=
`#{${relativePath}}`
                                    })
                                    document.getElementById("emojiList").appendChild(emojiE)
                                })

                            })
                        })
                    }
                    setProgressBar(100, 100, "progressSingle")
                    localforage.setItem(url, file).then(() => resolve())
                }
                if (!file) {
                    let xhr = new XMLHttpRequest()
                    xhr.responseType = "blob"
                    xhr.onprogress = oEvent => {
                        setProgressBar(oEvent.loaded, oEvent.total, "progressSingle")
                        document.querySelector("#loadBytes").innerHTML = `正在加载资源 共
${index}/${resToGet.length}个 当前资源${url} ${Math.ceil(oEvent.loaded / 1024)}KB /
${Math.ceil(oEvent.total / 1024)}KB`
                    }
                    xhr.onreadystatechange = function () {
                        if (xhr.readyState == 4 && xhr.status == 200) {
                            file = xhr.response
                            fileOK()
                        }
                    }
                    xhr.open("GET", "https://feiesource.oss-cn-hangzhou.aliyuncs.com" + url,
true);
                    xhr.send();
                } else { fileOK() }
            })
    })
}
```

此外，针对音频定义了新类，便于控制由两部分组成且后半部分重复的背景音乐。

```
class musicObj {
    constructor(src) {
        this.src = src
    }
    play(option, value) {
        let ado = document.createElement("audio")
        ado.src = this.src
        ado.volume = sysVolume
        if (option == "switch") {
            ado.onended = () => audioRes[resToGet.indexOf(value)].play("loop")
        }
        if (option == "loop") {
            ado.loop = "loop"
        }
        musicPlaying.push(ado)
```

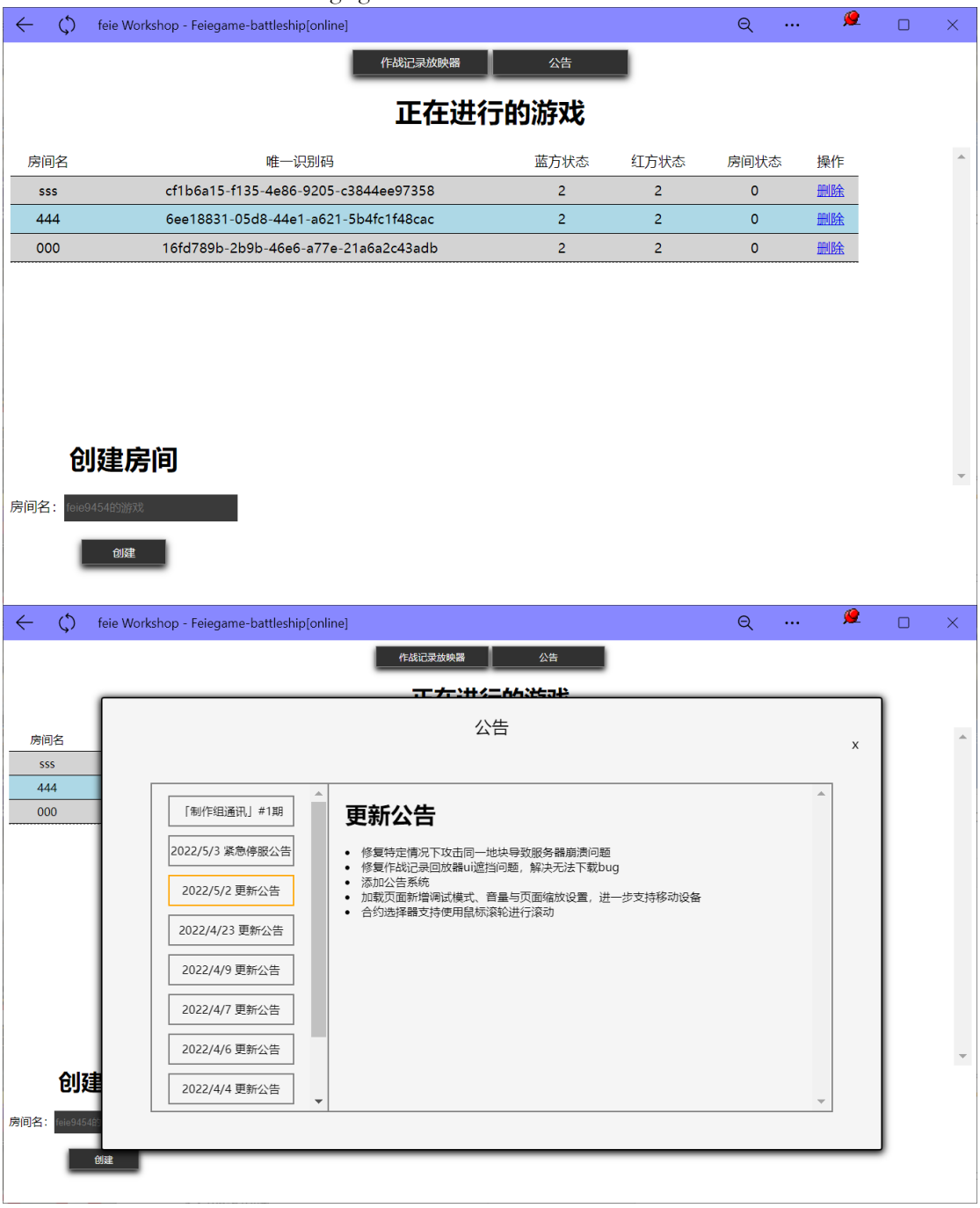
```
            ado.play()
        }
    }
```

基于 Promise 尝试与服务器建立 WebSocket 连接，在断线后自动重连，并提供连接完成后执行.then 的能力，方便其他系统开发。

```
function cws() {
    return new Promise(function (resolve, reject) {
        document.getElementById("WSLoadDiv").style.display = "block"
        try {
            if (!WebSocket) {
                alert("您的浏览器不支持 Websocket，请升级您的浏览器。")
            }
            ws = new WebSocket(`wss://${window.location.hostname}:9454`)

            ws.onmessage = function (evt) {
                ...
            }

            ws.onclose = function () {
                document.getElementsByTagName("title")[0].innerHTML = "Feiegame-
battleship[offline]"
                wsOK = false
                reject()
                tryWS()
                notice("神经网络连接失败，正在重试...", "error")
            };

        } catch (error) {
            wsOK = false
            reject()
            setTimeout(() => {
                tryWS()

            }, 2000);
            notice("神经网络连接失败，正在重试...", "error")
        }
    })
}
```

4.1.2 大厅模块详细实现

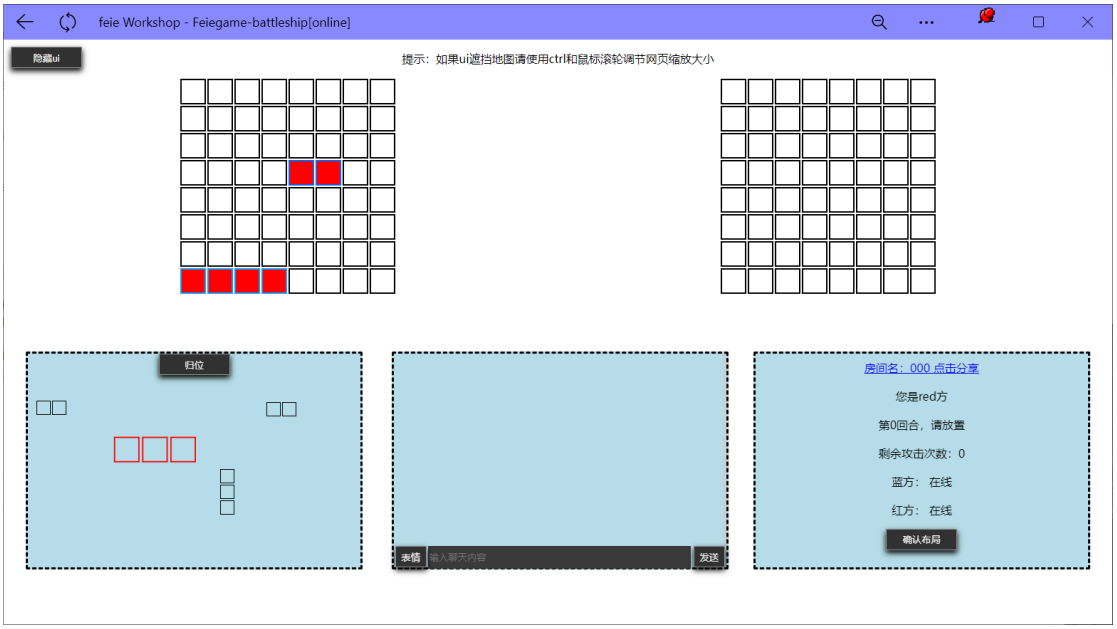
进入大厅后向服务器发送 getgames 请求，接收到响应后使用 dom 方法绘制表格。



4.1.3 游戏模块详细实现

4.1.3.1 放置棋子实现

游戏开始后，服务器根据所选“合约”情况通知客户端创建地图，地图采用<table>元素进行生成，并发放船只，同时对船只添加 pointerdown 事件监听器，响应拖动操作，并在拖动终止后判断所放位置并更改地图样式。玩家确认布局后，将会将棋盘数据使用 JSON 序列化并发送给服务器。



```
function addItemListener(element) {
    ...
    element.onpointerdown = function (event) {
        let offx = (window.event.clientX -
getElementLeft(element.parentElement.parentElement.parentElement.parentElement))
        let offy = (window.event.clientY -
getElementTop(element.parentElement.parentElement.parentElement.parentElement))
        element.parentElement.parentElement.parentElement.parentElement.style.transformOrigin =
`${offx}px ${offy}px`
        element.parentElement.parentElement.parentElement.parentElement.style.transform =
"scale(1.8,1.8)"
        let mouseoverE = document.elementsFromPoint(event.clientX, event.clientY)[3]
        if (mouseoverE) {
            if (mouseoverE.id.slice(0, 1) == "b" & mouseoverE.className == "mapBlock") {
                mouseover = mouseoverE.id.slice(1)
            } else {
                mouseover = undefined
            }
        }
        var event = event || window.event;
        var diffX = event.clientX
        var diffY = event.clientY
        let Y0 =
parseInt(element.parentElement.parentElement.parentElement.parentElement.style.top)
        let X0 =
parseInt(element.parentElement.parentElement.parentElement.parentElement.style.left)
        let element_ = element
        let objx
        let objy
        let refresh = function (event) {
            element_.parentElement.parentElement.parentElement.parentElement.style.top = objy
            element_.parentElement.parentElement.parentElement.parentElement.style.left = objx
        }
        document.onpointermove = function (event) {
            var event = event || window.event;
            objx = X0 + event.clientX - diffX + "px"
            objy = Y0 + event.clientY - diffY + "px"
            requestAnimationFrame(refresh)
            mouseoverE = document.elementsFromPoint(event.clientX, event.clientY)[3]
            if (mouseoverE) {
                if (mouseoverE.id.slice(0, 1) == "b" & mouseoverE.className == "mapBlock") {
                    mouseover = mouseoverE.id.slice(1)
                } else {
                    mouseover = undefined
                }
            }
        }
        let releaseFun = function () {
            document.onpointermove = null
            element.parentElement.parentElement.parentElement.parentElement.style.transform =
"scale(1,1)"
            document.onpointerup = null
            if (mouseover) {
                let success = false
                let eW =
element.parentElement.parentElement.parentElement.parentElement.getAttribute("width")
                let eH =
element.parentElement.parentElement.parentElement.parentElement.getAttribute("height")
                let isConflict = false
                if (eH >= eW) {
                    let pos = getPosH(element)
                    let mouseoverLine = id2pos(mouseover)[0]
                    if (mouseoverLine >= pos & mouseoverLine + (eH - pos - 1) <= height - 1) {
                        for (let index = 0; index < eH; index++) {
                            if (document.getElementById("b" + (mouseover - width * pos + index *
width)).style.backgroundColor == factionNow) { isConflict = true }
                        }
                        if (!isConflict) {
                            let posToPut = []
```

```
                            for (let index = 0; index < eH; index++) {
                                posToPut.push(mouseover - width * pos + index * width)
                            }
                            let
thisShip={1:b1ship,2:b2shipH,3:b3shipH,4:b4shipH}[Number(eH)](posToPut)
                            for (let index = 0; index < eH; index++) {
                                document.getElementById("b" + (mouseover - width * pos + index *
width)).style.backgroundColor = factionNow
                                document.getElementById("b" + (mouseover - width * pos + index *
width)).classList.add("canRemove")
                                mapArr[Math.floor((mouseover - width * pos + index * width) /
width)][(mouseover - width * pos + index * width) % width] = thisShip
                            }
                            element.onpointerdown = null
                            success = true
                        }
                    }
                } else {
                    let pos = getPosW(element)
                    if (pos <= mouseover % width & (mouseover % width) - pos <= (width - eW)) {
                        for (let index = 0; index < eW; index++) {
                            if (document.getElementById("b" + (mouseover - pos +
index)).style.backgroundColor == factionNow) { isConflict = true }
                        }
                        if (!isConflict) {
                            let posToPut = []
                            for (let index = 0; index < eW; index++) {
                                posToPut.push(mouseover - pos + index)
                            }
                            let
thisShip={1:b1ship,2:b2shipW,3:b3shipW,4:b4shipW}[Number(eW)](posToPut)
                            let ramColor = createRamColor()
                            for (let index = 0; index < eW; index++) {
                                document.getElementById("b" + (mouseover - pos +
index)).style.backgroundColor = factionNow
                                document.getElementById("b" + (mouseover - pos +
index)).style.borderColor = ramColor

                                document.getElementById("b" + (mouseover - pos +
index)).classList.add("canRemove")
                                mapArr[Math.floor((mouseover - pos + index) / width)][(mouseover
- pos + index) % width] = thisShip
                            }
                            success = true
                        }
                    }
                }
                if (success) {
                    element_.parentElement.parentElement.parentElement.parentElement.setAttribut
e("data-used", "true")
                    setTimeout(() => {
                        element_.parentElement.parentElement.parentElement.parentElement.style.o
pacity = 0
                        setTimeout(() => {
                            element_.parentElement.parentElement.parentElement.parentElement.rem
ove()
                        }, 100);
                    }, 10);
                }
            }
        }
        document.onpointerup = releaseFun
    }
}
```

4.1.3.2 攻击对方棋子实现

轮到玩家回合后服务器发送 canAttack 指令，玩家此时可以点击敌方棋盘发起攻击，攻击后会向服务器报告攻击坐标，服务器判断是否击中并向双方广播进攻结果，击中将会提示为“o”，未击中提示为“x”，被闪避提示为“!”。

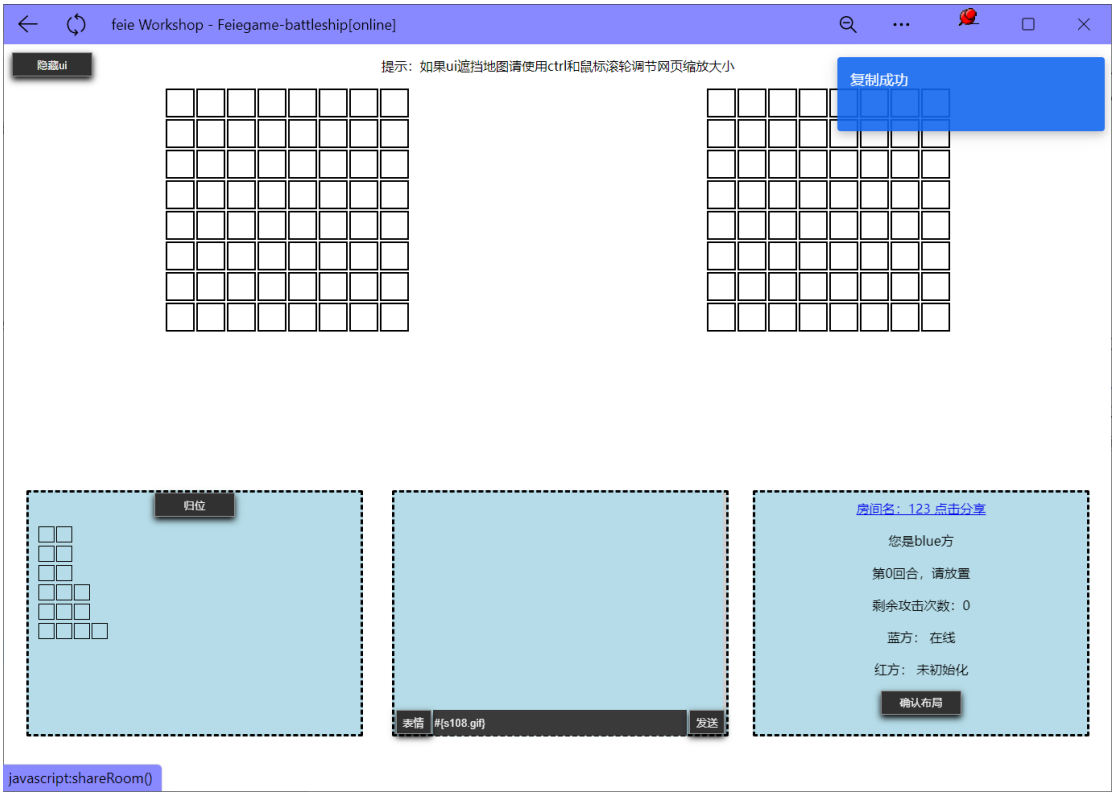
4.1.3.3 局内聊天实现

玩家发送信息时会向服务发送 chat 指令并附带聊天内容，服务器使用 naughty-words 库进行敏感词屏蔽后向玩家双方广播，玩家收到信息后替换特殊字符以防止 XSS 攻击后将信息添加到聊天框。表情则使用特殊代码实现，玩家选择表情后会添加例如“#{s108.gif}”的代码，收到信息时使用正则表达式进行检测，检测到此类代码后将会替换成对应的<img>标签。

```
case "chat":
    let msg = htmlspecialchars(received_msg[1].msg)
    let replaceEmoji = () => {
        msg = msg.replace(/#\{(.*?)\}/g, (match, p1) => {
            let img = document.createElement("img")
            img.src = URL.createObjectURL(emoji[p1])
            img.className = "chatEmoji"
            return img.outerHTML
        })
        if (msg.indexOf("#{") != -1) {
            replaceEmoji()
        }
    }
    if (msg.indexOf("#{") != -1) {
        replaceEmoji()
    }
    document.getElementById("chatContentDiv").innerHTML += `<div
class="chatContent"><strong>${received_msg[1].faction}:</strong> ${msg}</div>`
    document.querySelectorAll(".chatContent")[document.querySelectorAll(".chatContent").length
- 1].scrollIntoView({ behavior: "smooth" })
    break;
```

4.1.3.4 分享功能实现

玩家点击分享按钮后会尝试向用户剪切板中写入例如“我正在玩 feiegame-battleship，房间名：123，合约等级：0。点击链接加入房间 => https://sg.kenc.top/feiegame/battleship/#share66aa8348-ef1e-45cd-b157-87861024ea0a”的数据，玩家使用此链接进入网站则会使地址带有特殊 hash 值。而玩家进入网站时会检测是否带有此类 hash 值，若有此类 hash 则自动进行加载等操作并直接进入房间。

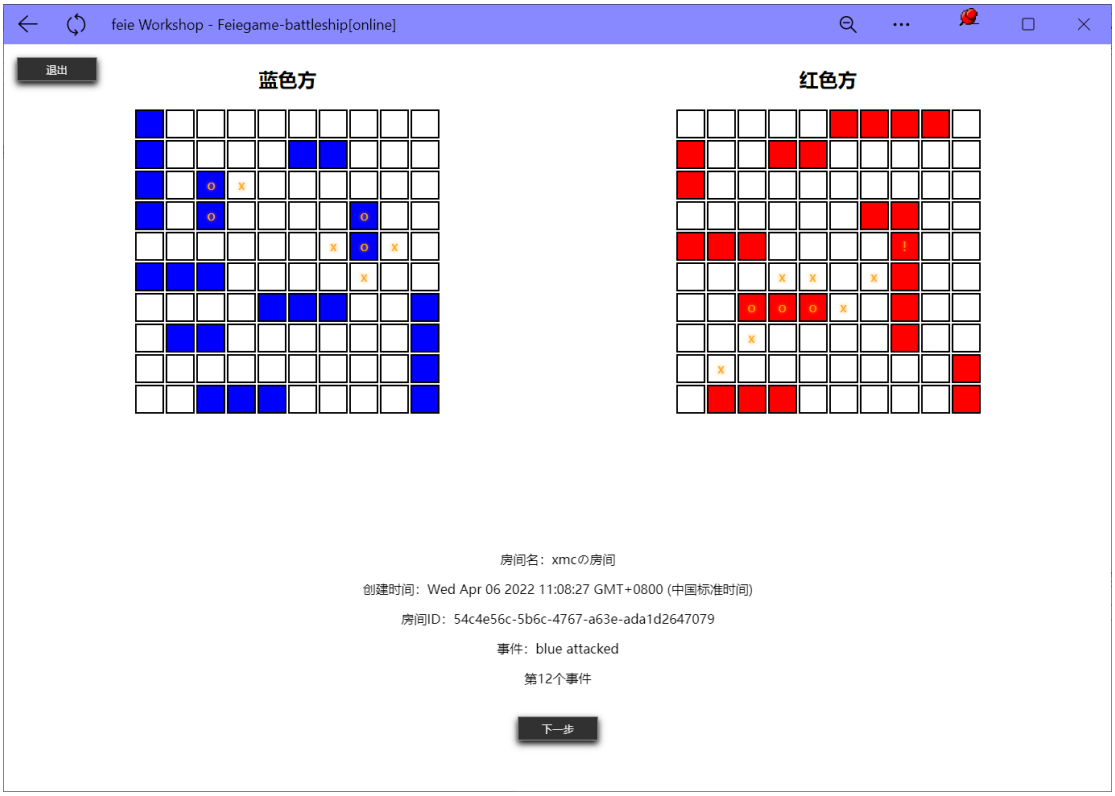


4.1.3.5 其他游戏内功能实现

点击隐藏 ui 按钮后会在棋盘上添加“`filter: blur(30px);`”的 css 样式。
点击归位按钮会让所有待部署棋子回到原位，以防止用户不慎将棋子拖出屏幕。

4.1.4 回放模块详细实现

每一局游戏结束后，服务器将会把存档移至归档区。玩家进入“作战记录回放器”后会向服务器发送 getSaveList 指令，服务器返回所有归档的存档的元数据。玩家点击放映或下载后，服务器会将完整存档发送至用户，前端再根据存档数据进行演算。



4.1.5 公告模块详细实现

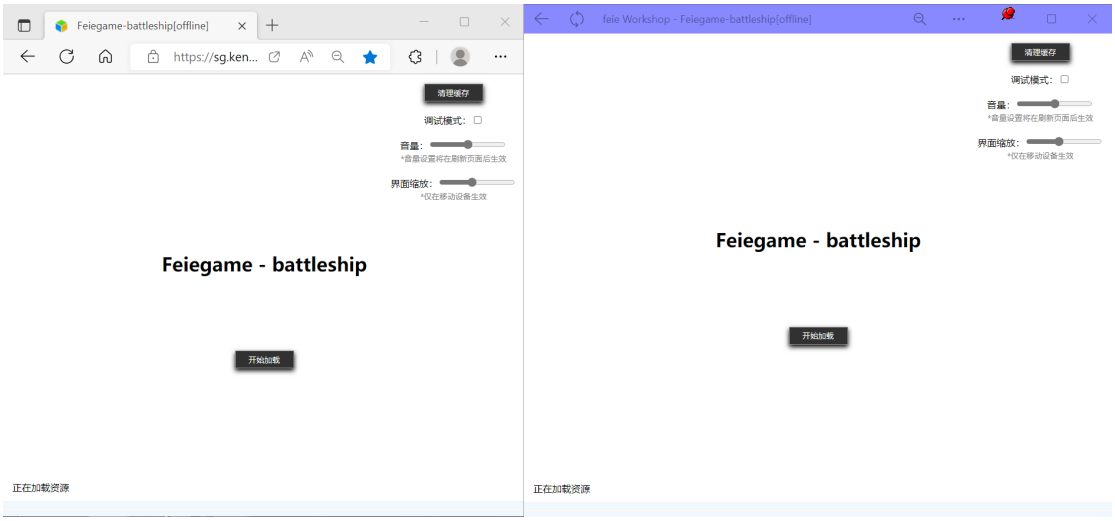
用户点击公告按钮后会向服务器发送 getNotification 请求，服务器读取本地的 publicNotification 文件并发送至用户。此文件格式基于 html，并且用“==========”作为不同公告之间分隔符，用“~~~~~~~~~~”作为标题和正文分隔符。此设计使公告可以拥有复杂的样式和交互性。

```
case "notification":
   noticeList=[]
   let asideContent =document.getElementById("publicNotificationContentAside")
   asideContent.innerHTML=""
   received_msg[1].split("==========").reverse().forEach(el=>{
      el=el.split("~~~~~~~~~~")
      noticeList.push({title:el[0],content:el[1]})
      let line = document.createElement("li")
      line.innerHTML=el[0]
      line.className="list-group-item"
      line.onclick=()=>{
         document.getElementById("publicNotificationContentMain").innerHTML=el[1]
         asideContent.childNodes.forEach(e=>e.className="list-group-item")
         line.classList.add("active")
      }
      asideContent.appendChild(line)
   })
   asideContent.childNodes[0].click()
break
```

4.1.6 PWA

当一个网站注册为 PWA 应用后，用户可以将此网站作为独立的应用安装，并且在独立

的窗口中打开。（左：作为常规网站打开；右：作为 PWA 应用打开）



网站注册为 PWA 应用需要提供 manifest 文件和注册 Service Worker 并监听 fetch 事件。

```
    //注册 ServiceWorker
    if ('serviceWorker' in navigator) {
        navigator.serviceWorker.register('/sw.js').then(function () {
            console.log('Service Worker Registered');
        }).catch(function (err) {
            console.log(err);
        });
    }

// 监听 fetch 事件
self.addEventListener('fetch', function (event) {
    if (event.request.url.includes('/src')) {
        event.respondWith(
            caches.match(event.request)
                .then(function (response) {
                    // 如果缓存中有，则直接返回缓存数据
                    if (response) {
                        return response
                    }
                    // 如果缓存中没有，则从网络中获取数据，并缓存
                    return fetch(event.request)
                        .then(function (response) {
                            return caches.open(cacheStorageKey)
                                .then(function (cache) {
                                    cache.put(event.request, response.clone())
                                    return response
                                })
                        })
                })
        )
    } else {
        event.respondWith(
            fetch(event.request).then(function (response) {
                return caches.open(cacheStorageKey).then(function (cache) {
                    cache.put(event.request, response.clone())
                    return response
                })
            }).catch(function () {
                return caches.match(event.request).then(function (response) {
                    return response
                })
            })
        );
    }
});
```

4.2 后端详细实现

4.2.1 建立连接

4.2.1.1 HTTP/2

使用 NodeJS 的“http”库创建 HTTP/1.1 服务器监听 80 端口，使用“http2”库监听 443 端口，80 端口监听到请求后发送 301 状态码将用户重定向至 443 端口的 HTTP/2 服务器。HTTP/2 带来了头部压缩，服务器推送等先进特性，可有效提高服务器网络效率。

```
const server = createSecureServer({
    //创建 https 服务器
    cert: readFileSync(process.argv[1].slice(0, process.argv[1].length - 10) + 'server.crt'),
    key: readFileSync(process.argv[1].slice(0, process.argv[1].length - 10) + 'server.key'),
    port: 443,
    allowHTTP1: true
});

const server80 = createServer((req, res) => {
    let d = new Date()
    let url = new URL(req.url, `http://${req.headers.host}`)
    console.log(d, url.pathname, 301, req.socket.remoteAddress, url.searchParams.toString(),
req.headers['user-agent'])
    res.writeHead(301, {
        'Location': 'https://' + req.headers.host + req.url,
        'X-Content-Type-Options': 'nosniff',
        'Content-Type': 'text/html ; charset=utf-8',
    });
    res.end("301 Moved Permanently");
});
```

4.2.1.2 WebSocket

使用 NodeJS 第三方库“ws”创建 WebSocket 服务器。

```
const wss = new WebSocketServer({ server });
wss.on('connection', function connection(ws, req) {
...
})
```

4.2.1.3 响应静态资源

当用户请求的不是 api 资源时返回本地文件。先判断请求是否为文件夹，若是文件夹默认返回 index.html，若成功找到文件判断浏览器是否支持 Gzip 压缩，若支持则将文件流输入 Gzip，再输入 res，若不支持则直接输入 res。使用流可以加快响应速度并减少服务端内存占用。

```
    let enableCompress = false
    let encoding = req.headers["accept-encoding"];
    if(!encoding){encoding=[]}
    let physicURL = `${process.argv[1].slice(0, process.argv[1].length - 11)}${url.pathname}`
    let requestType = physicURL.split('.').pop()
    let httpCode
```

```
    stat(physicURL).then((file) => {
        if (file.isFile()) {
            if (['js', 'css', 'html', 'json', 'txt', 'xml'].includes(requestType)) {
                if (encoding.includes('gzip')) {
                    enableCompress = true
                }
            }
            if (enableCompress) {
                httpCode = 200
                console.log(d, url.pathname, httpCode, req.socket.remoteAddress,
url.searchParams.toString(), req.headers['user-agent']);
                res.writeHead(200, {
                    'X-Content-Type-Options': 'nosniff',
                    'Content-Type': getType(requestType) + '; charset=utf-8',
                    'Content-Encoding': 'gzip',
                    'Cache-Control': 'no-cache',
                });
                createReadStream(physicURL).pipe(createGzip()).pipe(res);
            } else {
                if (getType(requestType)) {
                    httpCode = 200
                    console.log(d, url.pathname, httpCode, req.socket.remoteAddress,
url.searchParams.toString(), req.headers['user-agent']);
                    res.writeHead(200, {
                        'X-Content-Type-Options': 'nosniff',
                        'Content-Type': getType(requestType),
                        'Content-Length': file.size,
                    });
                } else {
                    httpCode = 200
                    console.log(d, url.pathname, httpCode, req.socket.remoteAddress,
url.searchParams.toString(), req.headers['user-agent']);
                    res.writeHead(200, {
                        'X-Content-Type-Options': 'nosniff',
                        'Content-Type': 'text/plain; charset=utf-8',
                        'Content-Length': file.size,
                        'Cache-Control': 'no-cache',
                    });
                }
                createReadStream(physicURL).pipe(res);
            }
        } else {
            if (physicURL.endsWith('/')) {
                physicURL += 'index.html'
            } else {
                httpCode = 302
                console.log(d, url.pathname, httpCode, req.socket.remoteAddress,
url.searchParams.toString(), req.headers['user-agent']);
                res.writeHead(302, {
                    'Location': `${url.pathname}/`,
                    'X-Content-Type-Options': 'nosniff',
                    'Content-Type': 'text/plain; charset=utf-8',
                });
                res.end("302 Moved Permanently");
                return
            }
            requestType = physicURL.split('.').pop()
            console.log(physicURL);
            stat(physicURL).then(file_ => {
                if (encoding.includes('gzip')) {
                    enableCompress = true
                }
                if (enableCompress) {
                    httpCode = 200
                    console.log(d, url.pathname, httpCode, req.socket.remoteAddress,
url.searchParams.toString(), req.headers['user-agent']);

                    res.writeHead(200, {
                        'X-Content-Type-Options': 'nosniff',
```

```
                    'Content-Type': getType(requestType) + '; charset=utf-8',
                    'Content-Encoding': 'gzip',
                    'cache-control': 'no-cache',
                });
                createReadStream(physicURL).pipe(createGzip()).pipe(res);
            }
            else {
                httpCode = 200
                console.log(d, url.pathname, httpCode, req.socket.remoteAddress,
url.searchParams.toString(), req.headers['user-agent']);
                res.writeHead(200, {
                    'X-Content-Type-Options': 'nosniff',
                    'Content-Type': getType(requestType) + '; charset=utf-8',
                    'Content-Length': file_.size
                });
                createReadStream(physicURL).pipe(res);
            }
        })
            .catch(err => {
                console.log('i',err);
                httpCode = 404
                console.log(d, url.pathname, httpCode, req.socket.remoteAddress,
url.searchParams.toString(), req.headers['user-agent']);
                res.writeHead(404, {
                    'X-Content-Type-Options': 'nosniff',
                    'Content-Type': 'text/plain; charset=utf-8',
                });
                res.end("404 Not Found");
            })
    }
})
    .catch(err => {
        console.log('o',err);
        httpCode = 404
        console.log(d, url.pathname, httpCode, req.socket.remoteAddress,
url.searchParams.toString(), req.headers['user-agent']);
        res.writeHead(404, {
            'X-Content-Type-Options': 'nosniff',
            'Content-Type': 'text/plain; charset=utf-8',
        });
        res.end("404 Not Found");
    })
```

4.2.2 返回请求数据

ws 服务器监听到请求后使用 switch 语句分析用户指令并返回对应数据。

4.2.2.1 返回游戏列表

先用 Promise 定义 getGames 方法，方便游戏判断等其他逻辑使用。

```
function getGames() {
    return new Promise(function (resolve, reject) {
        let returnData = []
        sql.query('SELECT * FROM savesmap', function (err, result) {
            if (err) {
                console.log('[SELECT ERROR] - ', err.message);
            }
            for (let index = 0; index < result.length; index++) {
                const element = result[index];
                let line = { name: element.name, id: element.uuid, blueState: element.blueState,
redState: element.redState, gameState: element.gameState, tags: JSON.parse(element.tags) }
                returnData.push(line)
```

```
            }
            resolve(returnData)
        })
    })
}
```

在 switch 语句中响应“getgames”请求。

```
case "getgames":
    getGames().then(r => {
        gamesArr = r
        let dataToSend = ["gamelist", gamesArr]
        dataToSend = JSON.stringify(dataToSend)
        ws.send(dataToSend)
    })
    break;
```

其他类型响应大同小异，不再赘述。

4.2.3 运算游戏数据

4.2.3.1 保存双方布局

保存玩家存档使用的是写入本地文件方式。

```
case "gameframe":
    let dataPacket = dataArr[1]
    switch (dataPacket.type) {
        case "confirm":
            appendFileSync("/home/lighthouse/www/WServer/saves/running/" + dataPacket.id +
".save", `${dataPacket.faction} confirm:\n`)
            appendFileSync("/home/lighthouse/www/WServer/saves/running/" + dataPacket.id +
".save", "JSON: " + dataPacket.data + "\n\n")
            broadcast(ws.uuid, ["gameframe", { type: "playerconfirm", data:
dataPacket.faction }])
            getNowTurn(dataPacket.id).then(turn => broadcast(ws.uuid, ["gameframe", { type:
"next", data: { turn: turn[0], factionNow: turn[1], canAttackTimes: turn[2] } }]))
            break;
            ...
```

4.2.3.2 转发玩家操作并写入存档

收到玩家攻击请求后，从本地读取存档并判断是否命中。

```
case "gameframe":
    let dataPacket = dataArr[1]
    switch (dataPacket.type) {
        case "confirm":
            appendFileSync("/home/lighthouse/www/WServer/saves/running/" + dataPacket.id +
".save", `${dataPacket.faction} confirm:\n`)
            appendFileSync("/home/lighthouse/www/WServer/saves/running/" + dataPacket.id +
".save", "JSON: " + dataPacket.data + "\n\n")
            broadcast(ws.uuid, ["gameframe", { type: "playerconfirm", data:
dataPacket.faction }])
            getNowTurn(dataPacket.id).then(turn => broadcast(ws.uuid, ["gameframe", { type:
"next", data: { turn: turn[0], factionNow: turn[1], canAttackTimes: turn[2] } }]))
            break;
        case "attack":
            let needSendNext = true
            getSaveTags(dataPacket.id).then(tags => {
                let height
```

```
                let width
                if (tags.includes("td11")) {
                    height = width = 9
                } else if (tags.includes("td21")) {
                    height = width = 10
                } else if (tags.includes("td31")) {
                    height = width = 12
                } else {
                    height = width = 8
                }
                let lookForMap
                if (dataPacket.faction == "blue") {
                    lookForMap = "red"
                } else {
                    lookForMap = "blue"
                }
                let save = getSave(dataPacket.id)
                let saveArr = save.split("\n")
                let map
                if (save.includes("red attack")) {
                    map = JSON.parse(saveArr[saveArr.lookForLastInclude(lookForMap) -
2].slice(6))
                } else {
                    map = JSON.parse(saveArr[saveArr.lookForFirstInclude(lookForMap) +
1].slice(6))
                }
                const normalAttack = async function (pos) {

                    let diedShip = []
                    let posToAtt = id2pos(pos, width)
                    if (map[posToAtt[0]][posToAtt[1]] == "attacked") {
                        return
                    }
                    if (map[posToAtt[0]][posToAtt[1]] == null) {
                        broadcast(ws.uuid, ["gameframe", { type: "attfail", data: { pos: pos,
faction: dataPacket.faction } }])
                        return { pos: pos, isMiss: "none" }
                    } else {
                        let chance = 1
                        let tags = await getSaveTags(dataPacket.id)
                        tags = JSON.stringify(tags)
                        if (tags.includes("tf11")) {
                            chance = 0.9
                        } else if (tags.includes("tf21")) {
                            chance = 0.8
                        } else if (tags.includes("tf31")) {
                            chance = 0.65
                        }
                        if (hit(chance)) {
                            broadcast(ws.uuid, ["gameframe", { type: "attsuccess", data: { pos:
pos, faction: dataPacket.faction } }])
                            let shipToBeAtt = map[posToAtt[0]][posToAtt[1]]
                            for (let index = 0; index < map.length; index++) {
                                const element = map[index];
                                for (let index_ = 0; index_ < element.length; index_++) {
                                    const element_ = element[index_];
                                    if (element_) {
                                        if (element_.id == shipToBeAtt.id) {
                                            map[index][index_].alive -= 1
                                            if (element_.alive == 0) {
                                                diedShip = diedShip.concat(element_.pos)
                                                map[index][index_] = "sinkShip"
                                            }
                                        }
                                    }
                                }
                            }
                            map[posToAtt[0]][posToAtt[1]] = "attacked"
                            if (diedShip.length > 0) {
```

```
                            shipDied(diedShip)
                        }
                        return { pos: pos, isMiss: "false" }
                    }
                    else {
                        broadcast(ws.uuid, ["gameframe", { type: "attmiss", data: { pos:
pos, faction: dataPacket.faction } }])
                        return { pos: pos, isMiss: "true" }
                    }
                }
            }
            const shipDied = (pos) => {
                broadcast(ws.uuid, ["gameframe", { type: "shipSink", data: { shipid: pos,
faction: dataPacket.faction } }])
            }
            const attEnd = (map, result) => {
                appendFileSync("/home/lighthouse/www/WServer/saves/running/" + dataPacket.id
+ ".save", `${dataPacket.faction} attacked:\n`)
                appendFileSync("/home/lighthouse/www/WServer/saves/running/" + dataPacket.id
+ ".save", "JSON: " + JSON.stringify(map) + "\n")
                appendFileSync("/home/lighthouse/www/WServer/saves/running/" + dataPacket.id
+ ".save", JSON.stringify({ type: dataPacket.data.type, result: result }) + "\n")
                if (!JSON.stringify(map).includes("aliveShip")) {
                    broadcast(ws.uuid, ["gameframe", { type: "playersuccess", faction:
dataPacket.faction }])
                    appendFileSync("/home/lighthouse/www/WServer/saves/running/" +
dataPacket.id + ".save", `${dataPacket.faction} successed!\n`)
                    let delSQL = 'DELETE FROM savesmap WHERE uuid="' + dataPacket.id + '"'
                    console.log(delSQL);
                    sql.query(delSQL, function (err, result) {
                        if (err) {
                            console.log('[SELECT ERROR] - ', err.message);
                        }
                    })
                    needSendNext = false
                    renameSync("/home/lighthouse/www/WServer/saves/running/" + dataPacket.id
+ ".save", "/home/lighthouse/www/WServer/saves/archive/" + dataPacket.id + ".save")
                }
                if (needSendNext) {
                    getNowTurn(dataPacket.id).then(turn => broadcast(ws.uuid, ["gameframe",
{ type: "next", data: { turn: turn[0], factionNow: turn[1], canAttackTimes: turn[2] } }]))
                }
            }
            switch (dataPacket.data.type) {
                case "normal":
                    let commands = []
                    dataPacket.data.pos.forEach(pos => {
                        commands.push(normalAttack(pos))
                    })
                    Promise.all(commands).then(result => {
                        attEnd(map, result)
                    })
                    break;
                default:
                    break;
            }
        })
    default:
        break;
}
break
default:
    break;
```

4.2.3.3 游戏胜负判断

每当玩家有船只被击沉便判断地图内是否剩余存活的船，若没有船存活便广播 playerSuccess 事件，代码已包含在上一部分。

4.2.3.4 其他细节

1）在玩家加入一局游戏后服务端将把对应 ws 链接的 id 属性设置为游戏 id，若需要广播信息，则遍历所有连接，找到 id 与游戏 id 一致的并发送信息。

```
function broadcast(id, data) {
    wss.clients.forEach(c => {
        if (c.uuid == id) {
            c.send(JSON.stringify(data))
        }
    })
}
```

2）闪避判定，玩家进行攻击时读取存档对应的“合约”计算出命中率，并由此函数判断是否击中。

```
const hit = (chance) => {
    return Math.random() <= chance
}
```

5 总结

本系统使用了大量时下流行技术如 WebSocket 和 NodeJS，NodeJS 在解决当今网络应用的问题时，提供了高性能、高可靠和低功耗的方法，依赖于 Google 的 v8 引擎，使 JavaScript 作为一个脚本语言，提供了高于同为脚本语言的 Python 数十倍的性能，且它的异步特性使得不需要进行复杂的多线程操作，降低了使用门槛也提高了程序效率。WebSocket 作为 http 的扩展不仅降低了 http 带来的额外开销，还为在线游戏提供了无限可能。此外还有 HTTP/2 和 PWA 等技术无一不展现给我们 WEB 技术的蓬勃发展前景。此外，Chrome 新支持的 File System Access API 更是给我们提供了在安全的上下文（https）中直接操作硬盘数据的能力，WEB 技术将变得越来越完善并安全。

微信小程序是基于 WEB 技术的，微信小程序的成功告诉我们网站已不是我们印象中的网站了，但是微信小程序依托于微信支持，而完全自由的 PWA 在国内常用 APP 中仅被“抖音”支持（2022/6）。

相信在不远的未来，将会有更多 PWA 应用被开发，带来跨平台的、统一的用户体验，单个应用的大小将会大幅减少，不再有相同 lib 重复占用空间的情况，也无需频繁的软件更新；对于开发者，WEB 的快迭代特性和不依赖与应用平台的特性将有利于开发者摆脱平台的束缚，使互联触手可及。

本项目大部分代码已在 Github 开源。前端开源地址：feie9455/feiegame-battleship: Html5 battleship game. (github.com)（附 commit 记录）；

后端开源地址：feie9455/feiegame-battleship-server (github.com)（附 commit 记录）。

在线游玩地址：https://sg.kenc.top/feiegame/battleship/

6 参考文献

[1] Nicholas C.Zakas.JavaScript 高级程序设计[M].北京:人民邮电出版社,2020.
[2] Jeremy Keith, Jeffrey Sambells.JavaScript DOM 编程艺术[M].北京:人民邮电出版社,2011.
[3] David Flanagan.JavaScript 权威手册[M].北京:机械工业出版社,2021.
[4] 霍春阳.Vue.js 设计与实现[M].北京:人民邮电出版社,2022.