

DOCUMENT_HEADING

10 files

multiplayer.css
style.css
styleWelcome.css
Battle.htm
index.htm
Multiplayer.htm
main.js
multiplayer.js
oppShipSetup.js
welcome.js

multiplayer.css

```
*,*::before,*::after{
    margin: 0;
    user-select: none;
    font-family: Arial, Helvetica, sans-serif;
}
body{
    box-sizing: border-box;
    overflow: hidden;
    background-color:  #fff;
    color:  #333;
}
/*Battle.htm*/
#container{
    display: flex;
    height: 100vh;
    width: 100vw;
    flex-direction: row;
    position: absolute;
}
#header{
    display: flex;
    width: 100vw;
    align-items: center;
    justify-content: center;
    background-color:  #222;
    color:  #fff;
    border-bottom: 3px solid rgb(10, 10, 10);
}
#header>p{
    font-size: 2rem;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    font-weight: bold;
}
#left, #right{
```

```
    display: flex;
    position: relative;
    width: 50vw;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    gap: 2rem;
}
#left-grid, #right-grid, #setup-grid{
    display: grid;
    border: 7px solid black;
    border-radius: 3%;
    grid-template-columns: repeat(10, minmax(1.85rem, 29px));
    grid-template-rows: repeat(10, minmax(1.85rem, 29px));
    background-color: black
}
.tile{
    border: 1px solid black;
    background-color: #4DA6FF;
    font-size: 0px;
    display: flex;
    justify-content: center;
    align-items: center;
}
.ship{
    background-color: rgb(80, 80, 80);
}
.shipClicked{
    background-color: black
}
.clicked{
    background-color: #ff9114;
}
.placed{
    text-decoration: line-through;
    color: rgb(200, 10, 10);
    background-color: #ff8881;
}
.ready{
    background-color: rgb(10, 200, 10);
}
#middle{
    position: fixed;
    left: 0;
    right: 0;
    top: 0;
    bottom: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    z-index: 10;
}
.background{
    background-color: rgba(20, 20, 20, 0.7);
}
```


```
}
#popup{
  position: absolute;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 40vh;
  width: 25vw;
  background-color: rgb(23, 19, 19);
  border-radius: 5rem;
  transition: scale 1s;
}
#popup > p{
  font-size: 2rem;
  color: rgb(221, 210, 210);
  position: absolute;
  top: 20%;
}
#popup > a{
  display: flex;
  justify-content: center;
  align-items: center;
  text-decoration: none;
  height: 3rem;
  width: 7rem;
  position: absolute;
  top: 60%;
  color: ■ black
  background-color: rgb(167, 167, 167);
  border-radius: 0.5rem;
  border: none;
}
#popup > a:hover{
  background-color: ■ black
  color: rgb(167, 167, 167);
}
/*setupboat.htm*/
#setup-container-player-1, #setup-container-player-2{
  display: grid;
  height: 100vh;
  width: 100vw;
  justify-content: center;
  align-items: center;
  grid-template-columns: repeat(3, 1fr);
  position: absolute;
}
#setup-grid{
  scale: 1.15;
  top: 20%;
  justify-content: center;
  align-self: center;
  height: 1fr;
  width: 1fr;
}
```

```
}
#setup-btns{
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
  align-items: center;
  justify-content: center;
  height: 1fr;
  width: 1fr;
}
#btn-container{
  display: flex;
  flex-direction: column;
  height: 20rem;
  width: 15rem;
  gap: 0.5rem;
  justify-content: center;
  align-items: center;
  padding: 5%;
  box-shadow: 0px 10px 10px rgb(100, 100, 100);
  border-radius: 10%;
  border: 1px solid #ccc;
}
#btn-container > hr{
  border: 1px solid black;
  width: 20%;
}
#btn-container p{
  font-size: 1.6rem;
  font-weight: 550;
}
#setup-right{
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  gap: 4rem;
}
#setup-middle{
  display: flex;
  height: 100vh;
  flex-direction: column;
  gap: 5rem;
  align-items: center;
  justify-content: center;
  position: relative;
  top: 10%;
}
#btn-container > button{
  border: 1px solid black;
  border-radius: 0.25rem;
  font-size: medium;
  cursor: pointer;
  width: 6rem;
  height: 2rem;
}
```

```
}
.default-ship-btn{
  background-color:  #5258ff;
}
#start-game-btn{
  border: 1px solid black;
  position: relative;
  text-decoration: none;
  font-size: 1.2rem;
  padding: 10px 5px;
  border: 1px solid black;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.5s ease-out;
}
.default-start-btn{
  background-color:  #ff3b0a;
}
.default-btns{
  background-color:  #b2d2f5;
}
.invisible{
  scale: 0;
}

.invisible-font{
  font-size: 0;
}

#left > p, #right > p{
  font-size: 2rem;
}
```

```
.statistics {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  background-color:  #f8f8f8;
  border: 1px solid #ccc;
  padding: 30px;
  border-radius: 10%;
  box-shadow: 0px 10px 10px rgb(100, 100, 100);
}
```

```
.statistics h3 {
  font-size: 18px;
  margin-bottom: 20px;
  text-align: center;
}
```

```
.statistics p {
```

```
margin-bottom: 10px;
}

.statistics button {
  background-color: #e74c3c;
  color: #fff;
  border: none;
  font-size: 16px;
  border-radius: 4px;
  cursor: pointer;
  margin-top: 30px;
}

header, #header{
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #222;
  color: #fff;
  position: fixed;
  top: 0;
  left: 0;
  width: 100vw;
  padding: 20px;
  font-size: 1.5rem;
  font-weight: 800;
  text-align: center;
  z-index: 20;
}

/*Tile sink bomb classes*/
.tile.bomb::before, .tile.bomb::after{
  content: '';
  border-radius: 50%;
  background-color: red
}

.tile.bomb::before{
  height: 0.5rem;
  width: 0.5rem;
}

.tileSink.tile.tileSink::before{
  transform: rotate(45deg);
}

.tile.tileSink::after{
  transform: rotate(-45deg);
}

.tile.tileSink::before, .tile.tileSink::after{
  content: '';
  position: absolute;
  width: 0.2rem;
  height: 1rem;
  background-color: red
}

.shipSink{
```

```
    background-color: ■ red
  }
/*player turn classes*/
body.player1 #left-grid > button:hover, body.player1 #left-grid:hover{
  cursor: not-allowed;
}
body.player1 #right-grid > button:hover, body.player1 #right-grid:hover{
  cursor: crosshair;
}
body.player2 #left-grid > button:hover, body.player2 #left-grid:hover{
  cursor: crosshair;
}
body.player2 #right-grid > button:hover, body.player2 #right-grid:hover{
  cursor: not-allowed;
}
body.player1 .ship1{
  background-color: ■ black
}
body.player2 .ship2{
  background-color: ■ black
}

body.player1 .ship2.tileSink {
  background-color: ■ black
}
body.player2 .ship1.tileSink {
  background-color: ■ black
}
body.player2 .ship2.shipSink{
  background-color: ■ red
}
body.player1 .ship1.shipSink{
  background-color: ■ red
}

}

.waitingScreen{
  display: flex;
  align-items: center;
  justify-content: center;
  height: 100vh;
  width: 100vw;
  position: absolute;
  background-color: ■ black
}

.waitingScreen #header{
  position: absolute;
  top: 0;
  right: 0;
  height: 10vh;
}
.continue{
  padding: 20px 30px;
}
```

```
    position: absolute;
    background-color: #237fdc;
    border-radius: 10px;
    z-index: 4;
    border: none;
}

.waitingScreen a{
    border-radius: 10%;
    background-color: #237fdc;
    position: absolute;
    opacity: 0.2;
    scale: 0;
}

.hover-1{
    height: 10px;
    width: 10px;
    padding: 30px 55px;
}

.hover-2{
    height: 10px;
    width: 10px;
    padding: 35px 60px;
    transition-delay: 0.4s;
}

.hover-3{
    height: 10px;
    width: 10px;
    padding: 40px 65px;
    transition-delay: 0.7s;
}

.continue.hover ~ a{
    animation: animate 1.2s;
}

@keyframes animate{
    0%{
        scale: 0;
    }
    60%{
        scale: 1;
    }
    100%{
        scale: 0;
    }
}

.continue.animate__animated.animate__bounceOutUp
{
    animation-delay: 100ms;
}


.continue-container{
    display: flex;
    align-items: center;
    justify-content: center;
}
```



```
    position: absolute;
    height: 200px;
    width: 200px;
}
.wave{
    position: absolute;
    width: 100%;
    height: 100px;
    bottom: 0;
    left: 0;
    background: url(wave.png);
    background-size: 1000px 100px;
}
.wave1{
    animation: animate1 30s linear infinite;
    animation-delay: 0s;
    z-index: 1000;
    bottom: 0;
    opacity: 1;
}
@keyframes animate1 {
    0%{
        background-position-x: 0;
    }
    100%{
        background-position-x: 1000px;
    }
}
.wave2{
    animation: animate2 15s linear infinite;
    z-index: 999;
    opacity: 0.5;
    animation-delay: -5s;
    bottom: 10px;
}
@keyframes animate2 {
    0%{
        background-position-x: 0;
    }
    100%{
        background-position-x: -1000px;
    }
}
.wave3{
    animation: animate1 30s linear infinite;
    z-index: 998;
    opacity: 0.2;
    animation-delay: -2s;
    bottom: 15px;
}
.wave4{
    animation: animate2 5s linear infinite;
    animation-delay: -5s;
    z-index: 997;
```

```
    opacity: 0.7;
    bottom: 20px;
  }
  .svg{
    scale: 0.6;
    width: 100px;
  }
  .home{
    position: absolute;
    left: 0;
    scale: 0.3;
    z-index: 15;
  }
  .home:hover{
    opacity: 0.6;
  }
}
```


style.css

```
*,*::before,*::after{
  margin: 0;
  user-select: none;
  font-family: Arial, Helvetica, sans-serif;
}
body{
  box-sizing: border-box;
  overflow: hidden;
  background-color:  #fff;
  color:  #333;
}
/*Battle.htm*/
#container{
  display: flex;
  height: 100vh;
  width: 100vw;
  flex-direction: row;
  position: absolute;
}
#header{
  display: flex;
  width: 100vw;
  align-items: center;
  justify-content: center;
  position: absolute;
  background-color:  #222;
  color:  #fff;
  border-bottom: 3px solid rgb(10, 10, 10);
}
#header>p{
  font-size: 2rem;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}
```


```
    font-weight: bold;
}


#left, #right{
    display: flex;
    position: relative;
    width: 50vw;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    gap: 2rem;
}


#left-grid, #right-grid, #setup-grid{
    display: grid;
    border: 7px solid black;
    border-radius: 3%;
    grid-template-columns: repeat(10, minmax(1.85rem, 29px));
    grid-template-rows: repeat(10, minmax(1.85rem, 29px));
}

.tile{
    border: 1px solid black;
    background-color:  #4DA6FF;
    font-size: 0px;
    display: flex;
    justify-content: center;
    align-items: center;
}


.ship{
    background-color: rgb(80, 80, 80);
}

.shipClicked{
    background-color:  black
}

.clicked{
    background-color:  #ff9114;
}

.placed{
    text-decoration: line-through;
    color: rgb(200, 10, 10);
    background-color:  #ff8881;
}

.ready{
    background-color: rgb(10, 200, 10);
}

.playerShip, .tileSink{
    background-color:  black
}

.shipSink{
    background-color: rgb(230, 50, 50);
}

#left-grid > button:hover, #left-grid:hover{
    cursor: not-allowed;
}
```


```
}
#right-grid >button:hover, #right-grid:hover{
  cursor: crosshair;
}
#middle{
  position: fixed;
  left: 0;
  right: 0;
  top: 0;
  bottom: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  z-index: 10;
}
.background{
  background-color: rgba(20, 20, 20, 0.7);
}
#popup{
  position: absolute;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 40vh;
  width: 25vw;
  background-color: rgb(23, 19, 19);
  border-radius: 5rem;
  transition: scale 1s;
}
#popup > p{
  font-size: 4rem;
  color: rgb(221, 210, 210);
  position: absolute;
  top: 20%;
}
#popup > a{
  display: flex;
  justify-content: center;
  align-items: center;
  text-decoration: none;
  height: 3rem;
  width: 7rem;
  position: absolute;
  top: 60%;
  color: ■ black
  background-color: rgb(167, 167, 167);
  border-radius: 0.5rem;
  border: none;
}
#popup > a:hover{
  background-color: ■ black
  color: rgb(167, 167, 167);
}
```

```
/*setupboat.htm*/
#setup-container{
  display: grid;
  height: 100vh;
  width: 100vw;
  justify-content: center;
  align-items: center;
  grid-template-columns: repeat(3, 1fr);
  position: absolute;
}
#setup-grid{
  scale: 1.15;
  top: 20%;
  justify-content: center;
  align-self: center;
  height: 1fr;
  width: 1fr;
}
#setup-btns{
  display: flex;
  flex-direction: column;
  gap: 0.5rem;
  align-items: center;
  justify-content: center;
  height: 1fr;
  width: 1fr;
}
#btn-container{
  display: flex;
  flex-direction: column;
  height: 20rem;
  width: 15rem;
  gap: 0.5rem;
  justify-content: center;
  align-items: center;
  padding: 5%;
  box-shadow: 0px 10px 10px rgb(100, 100, 100);
  border-radius: 10%;
  border: 1px solid #ccc;
}
#btn-container > hr{
  border: 1px solid black;
  width: 20%;
}
#btn-container p{
  font-size: 1.6rem;
  font-weight: 550;
}
#setup-right{
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  gap: 4rem;
}
```


```
#setup-middle{
  display: flex;
  height: 100vh;
  flex-direction: column;
  gap: 5rem;
  align-items: center;
  justify-content: center;
  position: relative;
  top: 10%;
}
#btn-container > button{
  border: 1px solid black;
  border-radius: 0.25rem;
  font-size: medium;
  cursor: pointer;
  width: 6rem;
  height: 2rem;
}
.default-ship-btn{
  background-color:  #5258ff;
}
#start-game-btn{
  border: 1px solid black;
  position: relative;
  text-decoration: none;
  font-size: 1.2rem;
  padding: 10px 5px;
  border: 1px solid black;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.5s ease-out;
}
.default-start-btn{
  background-color:  #ff3b0a;
}
.default-btns{
  background-color:  #b2d2f5;
}
.invisible{
  scale: 0;
}
.invisible-font{
  font-size: 0;
}
.tile.bomb::before, .tile.bomb::after{
  content: '';
  border-radius: 50%;
  background-color:  red
}
.tile.bomb::before{
  height: 0.5rem;
}
```

```
    width: 0.5rem;
  }

#left > p, #right > p{
  font-size: 2rem;
}



.tile.tileSink::before, .tile.tileSink::after{
  content: '';
  position: absolute;
  width: 0.2rem;
  height: 1rem;
  background-color:  red
}

.tileSink.tile.tileSink::before{
  transform: rotate(45deg);
}
.tile.tileSink::after{
  transform: rotate(-45deg);
}

.statistics {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  background-color:  #f8f8f8;
  border: 1px solid #ccc;
  padding: 30px;
  border-radius: 10%;
  box-shadow: 0px 10px 10px rgb(100, 100, 100);
}

.statistics h3 {
  font-size: 18px;
  margin-bottom: 20px;
  text-align: center;
}

.statistics p {
  margin-bottom: 10px;
}

.statistics button {
  background-color:  #e74c3c;
  color:  #fff;
  border: none;
  font-size: 16px;
  border-radius: 4px;
  cursor: pointer;
  margin-top: 30px;
}
```

```
header{
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #222;
  color: #fff;
  position: fixed;
  top: 0;
  left: 0;
  width: 100vw;
  padding: 20px;
  font-size: 1.5rem;
  font-weight: 800;
}
.home{
  position: absolute;
  left: 0;
  scale: 0.3;
  z-index: 15;
}
.home:hover{
  opacity: 0.6;
}

#waiting{
  height: 100vh;
  width: 100vw;
  z-index: 999;
  position: absolute;
}
```

styleWelcome.css

```
/* reset default styles */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  overflow-x: hidden;
  user-select: none;
}

/* global styles */
body {
  font-family: Arial, sans-serif;
  font-size: 16px;
  line-height: 1.5;
  background-color: #fff;
  color: #333;
}

.container {
```



```
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 20px;
  }

/* header styles */
header {
  background-color: #222;
  color: #fff;
  padding: 30px;
}

/* main styles */
main {
  padding: 50px 0;
}

.hero {
  text-align: center;
}

.hero h1 {
  font-size: 4rem;
  margin-bottom: 20px;
}

.hero p {
  font-size: 1.5rem;
  margin-bottom: 40px;
}

.instructions {
  margin: 50px auto;
  width: 80%;
  max-width: 800px;
  padding: 20px;
  border-radius: 5px;
}

.instructions h2 {
  font-size: 28px;
  font-weight: bold;
  margin-bottom: 20px;
  text-align: center;
}

.instructions ol {
  font-size: 18px;
  line-height: 1.5;
  margin-left: 20px;
}

.instructions li {
  margin-bottom: 10px;
}

/* Style for the "Play Now" button */
```

```
.play-btn {
  display: inline-block;
  background-color: #4CAF50;
  color: #fff;
  padding: 10px 20px;
  border-radius: 5px;
  text-decoration: none;
  transition: all 0.3s ease;
}

.play-btn:hover {
  background-color: #3e8e41;
}

.svg{
  z-index: -1;
  position: relative;
  height: 25vh;
  width: 100vw;
}
```

Battle.htm

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Battleships Game</title>
  <link rel="stylesheet" href="/style.css">
  <script src="main.js" defer type="module"></script>
</head>
<body>
  <div id="setup-container">
    <header> <a href="index.htm" class="home"></a>SET UP
YOUR SHIPS!</header>
    <div id="setup-btns">
      <div id="btn-container">
        <p>Select your ship!</p>
        <button data-ship-btn class="default-ship-btn">Destroyer</button>
        <button data-ship-btn class="default-ship-btn">Submarine</button>
        <button data-ship-btn class="default-ship-btn">Cruiser</button>
        <button data-ship-btn class="default-ship-btn">Battleship</button>
        <button data-ship-btn class="default-ship-btn">Carrier</button>
        <hr>
        <button data-remove-btn class="default-btns">Remove</button>
        <button data-clear-btn class="default-btns">Clear</button>
        <button data-rotate-btn class="default-btns">Rotate</button>
      </div>
    </div>
  </div>
```

```
<div id="setup-middle">
  <div id="setup-grid">
    <button class="tile" data-tile>A1</button>
    <button class="tile" data-tile>B1</button>
    <button class="tile" data-tile>C1</button>
    <button class="tile" data-tile>D1</button>
    <button class="tile" data-tile>E1</button>
    <button class="tile" data-tile>F1</button>
    <button class="tile" data-tile>G1</button>
    <button class="tile" data-tile>H1</button>
    <button class="tile" data-tile>I1</button>
    <button class="tile" data-tile>J1</button>
    <button class="tile" data-tile>A2</button>
    <button class="tile" data-tile>B2</button>
    <button class="tile" data-tile>C2</button>
    <button class="tile" data-tile>D2</button>
    <button class="tile" data-tile>E2</button>
    <button class="tile" data-tile>F2</button>
    <button class="tile" data-tile>G2</button>
    <button class="tile" data-tile>H2</button>
    <button class="tile" data-tile>I2</button>
    <button class="tile" data-tile>J2</button>
    <button class="tile" data-tile>A3</button>
    <button class="tile" data-tile>B3</button>
    <button class="tile" data-tile>C3</button>
    <button class="tile" data-tile>D3</button>
    <button class="tile" data-tile>E3</button>
    <button class="tile" data-tile>F3</button>
    <button class="tile" data-tile>G3</button>
    <button class="tile" data-tile>H3</button>
    <button class="tile" data-tile>I3</button>
    <button class="tile" data-tile>J3</button>
    <button class="tile" data-tile>A4</button>
    <button class="tile" data-tile>B4</button>
    <button class="tile" data-tile>C4</button>
    <button class="tile" data-tile>D4</button>
    <button class="tile" data-tile>E4</button>
    <button class="tile" data-tile>F4</button>
    <button class="tile" data-tile>G4</button>
    <button class="tile" data-tile>H4</button>
    <button class="tile" data-tile>I4</button>
    <button class="tile" data-tile>J4</button>
    <button class="tile" data-tile>A5</button>
    <button class="tile" data-tile>B5</button>
    <button class="tile" data-tile>C5</button>
    <button class="tile" data-tile>D5</button>
    <button class="tile" data-tile>E5</button>
    <button class="tile" data-tile>F5</button>
    <button class="tile" data-tile>G5</button>
    <button class="tile" data-tile>H5</button>
    <button class="tile" data-tile>I5</button>
    <button class="tile" data-tile>J5</button>
    <button class="tile" data-tile>A6</button>
    <button class="tile" data-tile>B6</button>
```

```
<button class="tile" data-tile>C6</button>
<button class="tile" data-tile>D6</button>
<button class="tile" data-tile>E6</button>
<button class="tile" data-tile>F6</button>
<button class="tile" data-tile>G6</button>
<button class="tile" data-tile>H6</button>
<button class="tile" data-tile>I6</button>
<button class="tile" data-tile>J6</button>
<button class="tile" data-tile>A7</button>
<button class="tile" data-tile>B7</button>
<button class="tile" data-tile>C7</button>
<button class="tile" data-tile>D7</button>
<button class="tile" data-tile>E7</button>
<button class="tile" data-tile>F7</button>
<button class="tile" data-tile>G7</button>
<button class="tile" data-tile>H7</button>
<button class="tile" data-tile>I7</button>
<button class="tile" data-tile>J7</button>
<button class="tile" data-tile>A8</button>
<button class="tile" data-tile>B8</button>
<button class="tile" data-tile>C8</button>
<button class="tile" data-tile>D8</button>
<button class="tile" data-tile>E8</button>
<button class="tile" data-tile>F8</button>
<button class="tile" data-tile>G8</button>
<button class="tile" data-tile>H8</button>
<button class="tile" data-tile>I8</button>
<button class="tile" data-tile>J8</button>
<button class="tile" data-tile>A9</button>
<button class="tile" data-tile>B9</button>
<button class="tile" data-tile>C9</button>
<button class="tile" data-tile>D9</button>
<button class="tile" data-tile>E9</button>
<button class="tile" data-tile>F9</button>
<button class="tile" data-tile>G9</button>
<button class="tile" data-tile>H9</button>
<button class="tile" data-tile>I9</button>
<button class="tile" data-tile>J9</button>
<button class="tile" data-tile>A10</button>
<button class="tile" data-tile>B10</button>
<button class="tile" data-tile>C10</button>
<button class="tile" data-tile>D10</button>
<button class="tile" data-tile>E10</button>
<button class="tile" data-tile>F10</button>
<button class="tile" data-tile>G10</button>
<button class="tile" data-tile>H10</button>
<button class="tile" data-tile>I10</button>
<button class="tile" data-tile>J10</button>
</div>
<button data-start-game-btn id='start-game-btn' class="default-start-btn">Start
Game</button>
</div>

<div id="setup-right">
  <div class="statistics">
```

```
<h3>Statistics</h3>
<p data-accuracy>Hit Accuracy: NA</p>
<p data-best-game>Best Game: NA</p>
<p data-win-percentage>Win Percentage: NA</p>
<p data-games-played>Games Played: NA</p>
</div>
</div>
</div>

<div id="container" class="invisible">
  <div id="header">
    <a href="index.htm" class="home"></a>
    <p>BATTLESHIPS</p>
  </div>
  <div id="left">
    <p>Player's ships</p>
    <div id="left-grid">
      <button class="tile" data-tile-left>A1</button>
      <button class="tile" data-tile-left>B1</button>
      <button class="tile" data-tile-left>C1</button>
      <button class="tile" data-tile-left>D1</button>
      <button class="tile" data-tile-left>E1</button>
      <button class="tile" data-tile-left>F1</button>
      <button class="tile" data-tile-left>G1</button>
      <button class="tile" data-tile-left>H1</button>
      <button class="tile" data-tile-left>I1</button>
      <button class="tile" data-tile-left>J1</button>
      <button class="tile" data-tile-left>A2</button>
      <button class="tile" data-tile-left>B2</button>
      <button class="tile" data-tile-left>C2</button>
      <button class="tile" data-tile-left>D2</button>
      <button class="tile" data-tile-left>E2</button>
      <button class="tile" data-tile-left>F2</button>
      <button class="tile" data-tile-left>G2</button>
      <button class="tile" data-tile-left>H2</button>
      <button class="tile" data-tile-left>I2</button>
      <button class="tile" data-tile-left>J2</button>
      <button class="tile" data-tile-left>A3</button>
      <button class="tile" data-tile-left>B3</button>
      <button class="tile" data-tile-left>C3</button>
      <button class="tile" data-tile-left>D3</button>
      <button class="tile" data-tile-left>E3</button>
      <button class="tile" data-tile-left>F3</button>
      <button class="tile" data-tile-left>G3</button>
      <button class="tile" data-tile-left>H3</button>
      <button class="tile" data-tile-left>I3</button>
      <button class="tile" data-tile-left>J3</button>
      <button class="tile" data-tile-left>A4</button>
      <button class="tile" data-tile-left>B4</button>
      <button class="tile" data-tile-left>C4</button>
      <button class="tile" data-tile-left>D4</button>
      <button class="tile" data-tile-left>E4</button>
      <button class="tile" data-tile-left>F4</button>
      <button class="tile" data-tile-left>G4</button>
      <button class="tile" data-tile-left>H4</button>
```

[illegible]

```
<button class="tile" data-tile-left>D10</button>
<button class="tile" data-tile-left>E10</button>
<button class="tile" data-tile-left>F10</button>
<button class="tile" data-tile-left>G10</button>
<button class="tile" data-tile-left>H10</button>
<button class="tile" data-tile-left>I10</button>
<button class="tile" data-tile-left>J10</button>
</div>
</div>
<div id="middle" class="invisible">
  <p id="message"></p>
  <div id='popup' class="invisible">
    <p data-win-text></p>
    <a href="Battle.htm" data-new-game>New Game</a>
  </div>
</div>
<div id="waiting" class="invisible"></div>
<div id="right">
  <p>Opponent's ships</p>
  <div id="right-grid">
    <button class="tile" data-tile-right>A1</button>
    <button class="tile" data-tile-right>B1</button>
    <button class="tile" data-tile-right>C1</button>
    <button class="tile" data-tile-right>D1</button>
    <button class="tile" data-tile-right>E1</button>
    <button class="tile" data-tile-right>F1</button>
    <button class="tile" data-tile-right>G1</button>
    <button class="tile" data-tile-right>H1</button>
    <button class="tile" data-tile-right>I1</button>
    <button class="tile" data-tile-right>J1</button>
    <button class="tile" data-tile-right>A2</button>
    <button class="tile" data-tile-right>B2</button>
    <button class="tile" data-tile-right>C2</button>
    <button class="tile" data-tile-right>D2</button>
    <button class="tile" data-tile-right>E2</button>
    <button class="tile" data-tile-right>F2</button>
    <button class="tile" data-tile-right>G2</button>
    <button class="tile" data-tile-right>H2</button>
    <button class="tile" data-tile-right>I2</button>
    <button class="tile" data-tile-right>J2</button>
    <button class="tile" data-tile-right>A3</button>
    <button class="tile" data-tile-right>B3</button>
    <button class="tile" data-tile-right>C3</button>
    <button class="tile" data-tile-right>D3</button>
    <button class="tile" data-tile-right>E3</button>
    <button class="tile" data-tile-right>F3</button>
    <button class="tile" data-tile-right>G3</button>
    <button class="tile" data-tile-right>H3</button>
    <button class="tile" data-tile-right>I3</button>
    <button class="tile" data-tile-right>J3</button>
    <button class="tile" data-tile-right>A4</button>
    <button class="tile" data-tile-right>B4</button>
    <button class="tile" data-tile-right>C4</button>
    <button class="tile" data-tile-right>D4</button>
    <button class="tile" data-tile-right>E4</button>
```


[illegible]


```

        <button class="tile" data-tile-right>A10</button>
        <button class="tile" data-tile-right>B10</button>
        <button class="tile" data-tile-right>C10</button>
        <button class="tile" data-tile-right>D10</button>
        <button class="tile" data-tile-right>E10</button>
        <button class="tile" data-tile-right>F10</button>
        <button class="tile" data-tile-right>G10</button>
        <button class="tile" data-tile-right>H10</button>
        <button class="tile" data-tile-right>I10</button>
        <button class="tile" data-tile-right>J10</button>
    </div>
</div>
</div>
</body>
</html>

```

index.htm

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Battleship Game - Play Now</title>
    <link rel="stylesheet" href="styleWelcome.css">
    <script src="welcome.js" defer></script>
</head>
<body>
    <header>
    </header>
    <main>
        <section class="hero">
            <h1>Battleship Game</h1>
            <p>Command your fleet and sink your opponent's ships in this classic game of strategy
and skill.</p>
            <a class="play-btn" href="Battle.htm">Play Against Robot</a>
            <a class="play-btn" href="Multiplayer.htm">Play Against Friend</a>
        </section>
        
        <section class="instructions">
            <h2>How to Play Battleships</h2>
            <ol>
                <li>Each player places their ships on the board by clicking on the cells. You can
rotate the ship by clicking the rotate button.</li>
                <li>Once both players have placed their ships, the game begins. The players take
turns guessing the location of the other player's ships by clicking on the cells on the board.
</li>
                <li>If a player's guess hits a ship, it will be marked as a hit. If the guess
misses, it will be marked as a miss.</li>
                <li>The game continues until one player has sunk all of the other player's ships.
</li>
            </ol>
        </section>
    </main>
</body>

```

```
</html>
```

Multiplayer.htm

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="multiplayer.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"/>
  <script src="multiplayer.js" defer></script>
</head>
<body>
  <div id="setup-container-player-1" class="">
    <header><a href="index.htm" class="home"></a>SET UP
YOUR SHIPS - Player 1</header>
    <div id="setup-btns">
      <div id="btn-container">
        <p>Select your ship!</p>
        <button data-ship-btn-player-1 data-ship-btn class="default-ship-
btn">Destroyer</button>
        <button data-ship-btn-player-1 data-ship-btn class="default-ship-
btn">Submarine</button>
        <button data-ship-btn-player-1 data-ship-btn class="default-ship-
btn">Cruiser</button>
        <button data-ship-btn-player-1 data-ship-btn class="default-ship-
btn">Battleship</button>
        <button data-ship-btn-player-1 data-ship-btn class="default-ship-
btn">Carrier</button>
        <hr>
        <button data-remove-btn-player-1 class="default-btns">Remove</button>
        <button data-clear-btn-player-1 class="default-btns">Clear</button>
        <button data-rotate-btn-player-1 class="default-btns">Rotate</button>
      </div>
    </div>

    <div id="setup-middle">
      <div id="setup-grid">
        <button class="tile" data-tile-player-1>A1</button>
        <button class="tile" data-tile-player-1>B1</button>
        <button class="tile" data-tile-player-1>C1</button>
        <button class="tile" data-tile-player-1>D1</button>
        <button class="tile" data-tile-player-1>E1</button>
        <button class="tile" data-tile-player-1>F1</button>
        <button class="tile" data-tile-player-1>G1</button>
        <button class="tile" data-tile-player-1>H1</button>
        <button class="tile" data-tile-player-1>I1</button>
        <button class="tile" data-tile-player-1>J1</button>
        <button class="tile" data-tile-player-1>A2</button>
```

[illegible]

```
<button class="tile" data-tile-player-1>G7</button>
<button class="tile" data-tile-player-1>H7</button>
<button class="tile" data-tile-player-1>I7</button>
<button class="tile" data-tile-player-1>J7</button>
<button class="tile" data-tile-player-1>A8</button>
<button class="tile" data-tile-player-1>B8</button>
<button class="tile" data-tile-player-1>C8</button>
<button class="tile" data-tile-player-1>D8</button>
<button class="tile" data-tile-player-1>E8</button>
<button class="tile" data-tile-player-1>F8</button>
<button class="tile" data-tile-player-1>G8</button>
<button class="tile" data-tile-player-1>H8</button>
<button class="tile" data-tile-player-1>I8</button>
<button class="tile" data-tile-player-1>J8</button>
<button class="tile" data-tile-player-1>A9</button>
<button class="tile" data-tile-player-1>B9</button>
<button class="tile" data-tile-player-1>C9</button>
<button class="tile" data-tile-player-1>D9</button>
<button class="tile" data-tile-player-1>E9</button>
<button class="tile" data-tile-player-1>F9</button>
<button class="tile" data-tile-player-1>G9</button>
<button class="tile" data-tile-player-1>H9</button>
<button class="tile" data-tile-player-1>I9</button>
<button class="tile" data-tile-player-1>J9</button>
<button class="tile" data-tile-player-1>A10</button>
<button class="tile" data-tile-player-1>B10</button>
<button class="tile" data-tile-player-1>C10</button>
<button class="tile" data-tile-player-1>D10</button>
<button class="tile" data-tile-player-1>E10</button>
<button class="tile" data-tile-player-1>F10</button>
<button class="tile" data-tile-player-1>G10</button>
<button class="tile" data-tile-player-1>H10</button>
<button class="tile" data-tile-player-1>I10</button>
<button class="tile" data-tile-player-1>J10</button>
</div>
<button data-confirm-player-1 id='start-game-btn' class="default-start-btn">Start
Game</button>
</div>
</div>

<div id="setup-container-player-2" class="invisible">
  <header><a href="index.htm" class="home"></a>SET UP
  YOUR SHIPS - Player 2</header>
  <div id="setup-btns">
    <div id="btn-container">
      <p>Select your ship!</p>
      <button data-ship-btn-player-2 data-ship-btn class="default-ship-
      btn">Destroyer</button>
      <button data-ship-btn-player-2 data-ship-btn class="default-ship-
      btn">Submarine</button>
      <button data-ship-btn-player-2 data-ship-btn class="default-ship-
      btn">Cruiser</button>
      <button data-ship-btn-player-2 data-ship-btn class="default-ship-
      btn">Battleship</button>
      <button data-ship-btn-player-2 data-ship-btn class="default-ship-
      btn">Carrier</button>
```

```
        <hr>
        <button data-remove-btn-player-2 class="default-btns">Remove</button>
        <button data-clear-btn-player-2 class="default-btns">Clear</button>
        <button data-rotate-btn-player-2 class="default-btns">Rotate</button>
    </div>
</div>

<div id="setup-middle">
    <div id="setup-grid">
        <button class="tile" data-tile-player-2>A1</button>
        <button class="tile" data-tile-player-2>B1</button>
        <button class="tile" data-tile-player-2>C1</button>
        <button class="tile" data-tile-player-2>D1</button>
        <button class="tile" data-tile-player-2>E1</button>
        <button class="tile" data-tile-player-2>F1</button>
        <button class="tile" data-tile-player-2>G1</button>
        <button class="tile" data-tile-player-2>H1</button>
        <button class="tile" data-tile-player-2>I1</button>
        <button class="tile" data-tile-player-2>J1</button>
        <button class="tile" data-tile-player-2>A2</button>
        <button class="tile" data-tile-player-2>B2</button>
        <button class="tile" data-tile-player-2>C2</button>
        <button class="tile" data-tile-player-2>D2</button>
        <button class="tile" data-tile-player-2>E2</button>
        <button class="tile" data-tile-player-2>F2</button>
        <button class="tile" data-tile-player-2>G2</button>
        <button class="tile" data-tile-player-2>H2</button>
        <button class="tile" data-tile-player-2>I2</button>
        <button class="tile" data-tile-player-2>J2</button>
        <button class="tile" data-tile-player-2>A3</button>
        <button class="tile" data-tile-player-2>B3</button>
        <button class="tile" data-tile-player-2>C3</button>
        <button class="tile" data-tile-player-2>D3</button>
        <button class="tile" data-tile-player-2>E3</button>
        <button class="tile" data-tile-player-2>F3</button>
        <button class="tile" data-tile-player-2>G3</button>
        <button class="tile" data-tile-player-2>H3</button>
        <button class="tile" data-tile-player-2>I3</button>
        <button class="tile" data-tile-player-2>J3</button>
        <button class="tile" data-tile-player-2>A4</button>
        <button class="tile" data-tile-player-2>B4</button>
        <button class="tile" data-tile-player-2>C4</button>
        <button class="tile" data-tile-player-2>D4</button>
        <button class="tile" data-tile-player-2>E4</button>
        <button class="tile" data-tile-player-2>F4</button>
        <button class="tile" data-tile-player-2>G4</button>
        <button class="tile" data-tile-player-2>H4</button>
        <button class="tile" data-tile-player-2>I4</button>
        <button class="tile" data-tile-player-2>J4</button>
        <button class="tile" data-tile-player-2>A5</button>
        <button class="tile" data-tile-player-2>B5</button>
        <button class="tile" data-tile-player-2>C5</button>
        <button class="tile" data-tile-player-2>D5</button>
        <button class="tile" data-tile-player-2>E5</button>
        <button class="tile" data-tile-player-2>F5</button>
```

[illegible]


```
        <button data-confirm-player-2 id='start-game-btn' class="default-start-btn">Start
Game</button>
    </div>
</div>
```

```
<div id="container" class="invisible">
    <div id="header">
        <a href="index.htm" class="home"></a>
        <p>BATTLESHIPS</p>
    </div>
    <div id="left">
        <p>Player-1 ships</p>
        <div id="left-grid">
            <button class="tile" data-tile-left>A1</button>
            <button class="tile" data-tile-left>B1</button>
            <button class="tile" data-tile-left>C1</button>
            <button class="tile" data-tile-left>D1</button>
            <button class="tile" data-tile-left>E1</button>
            <button class="tile" data-tile-left>F1</button>
            <button class="tile" data-tile-left>G1</button>
            <button class="tile" data-tile-left>H1</button>
            <button class="tile" data-tile-left>I1</button>
            <button class="tile" data-tile-left>J1</button>
            <button class="tile" data-tile-left>A2</button>
            <button class="tile" data-tile-left>B2</button>
            <button class="tile" data-tile-left>C2</button>
            <button class="tile" data-tile-left>D2</button>
            <button class="tile" data-tile-left>E2</button>
            <button class="tile" data-tile-left>F2</button>
            <button class="tile" data-tile-left>G2</button>
            <button class="tile" data-tile-left>H2</button>
            <button class="tile" data-tile-left>I2</button>
            <button class="tile" data-tile-left>J2</button>
            <button class="tile" data-tile-left>A3</button>
            <button class="tile" data-tile-left>B3</button>
            <button class="tile" data-tile-left>C3</button>
            <button class="tile" data-tile-left>D3</button>
            <button class="tile" data-tile-left>E3</button>
            <button class="tile" data-tile-left>F3</button>
            <button class="tile" data-tile-left>G3</button>
            <button class="tile" data-tile-left>H3</button>
            <button class="tile" data-tile-left>I3</button>
            <button class="tile" data-tile-left>J3</button>
            <button class="tile" data-tile-left>A4</button>
            <button class="tile" data-tile-left>B4</button>
            <button class="tile" data-tile-left>C4</button>
            <button class="tile" data-tile-left>D4</button>
            <button class="tile" data-tile-left>E4</button>
            <button class="tile" data-tile-left>F4</button>
            <button class="tile" data-tile-left>G4</button>
            <button class="tile" data-tile-left>H4</button>
            <button class="tile" data-tile-left>I4</button>
            <button class="tile" data-tile-left>J4</button>
            <button class="tile" data-tile-left>A5</button>
```

[illegible]


```
        <button class="tile" data-tile-left>G10</button>
        <button class="tile" data-tile-left>H10</button>
        <button class="tile" data-tile-left>I10</button>
        <button class="tile" data-tile-left>J10</button>
    </div>
</div>
<div id="middle" class="invisible">
    <div id='popup'>
        <p data-win-text></p>
        <a href="Multiplayer.htm" data-new-game>New Game</a>
    </div>
</div>
<div id="right">
    <p>Player-2 ships</p>
    <div id="right-grid">
        <button class="tile" data-tile-right>A1</button>
        <button class="tile" data-tile-right>B1</button>
        <button class="tile" data-tile-right>C1</button>
        <button class="tile" data-tile-right>D1</button>
        <button class="tile" data-tile-right>E1</button>
        <button class="tile" data-tile-right>F1</button>
        <button class="tile" data-tile-right>G1</button>
        <button class="tile" data-tile-right>H1</button>
        <button class="tile" data-tile-right>I1</button>
        <button class="tile" data-tile-right>J1</button>
        <button class="tile" data-tile-right>A2</button>
        <button class="tile" data-tile-right>B2</button>
        <button class="tile" data-tile-right>C2</button>
        <button class="tile" data-tile-right>D2</button>
        <button class="tile" data-tile-right>E2</button>
        <button class="tile" data-tile-right>F2</button>
        <button class="tile" data-tile-right>G2</button>
        <button class="tile" data-tile-right>H2</button>
        <button class="tile" data-tile-right>I2</button>
        <button class="tile" data-tile-right>J2</button>
        <button class="tile" data-tile-right>A3</button>
        <button class="tile" data-tile-right>B3</button>
        <button class="tile" data-tile-right>C3</button>
        <button class="tile" data-tile-right>D3</button>
        <button class="tile" data-tile-right>E3</button>
        <button class="tile" data-tile-right>F3</button>
        <button class="tile" data-tile-right>G3</button>
        <button class="tile" data-tile-right>H3</button>
        <button class="tile" data-tile-right>I3</button>
        <button class="tile" data-tile-right>J3</button>
        <button class="tile" data-tile-right>A4</button>
        <button class="tile" data-tile-right>B4</button>
        <button class="tile" data-tile-right>C4</button>
        <button class="tile" data-tile-right>D4</button>
        <button class="tile" data-tile-right>E4</button>
        <button class="tile" data-tile-right>F4</button>
        <button class="tile" data-tile-right>G4</button>
        <button class="tile" data-tile-right>H4</button>
        <button class="tile" data-tile-right>I4</button>
        <button class="tile" data-tile-right>J4</button>
    </div>
</div>
```

[illegible]

```
        <button class="tile" data-tile-right>F10</button>
        <button class="tile" data-tile-right>G10</button>
        <button class="tile" data-tile-right>H10</button>
        <button class="tile" data-tile-right>I10</button>
        <button class="tile" data-tile-right>J10</button>
    </div>
</div>
</div>

<div id="waiting-screen-confirm-1" class="waitingScreen invisible">
    <header>
        <p>BATTLESHIPS</p>
    </header>
    <div class="wave wave1"></div>
    <div class="wave wave2"></div>
    <div class="wave wave3"></div>
    <div class="wave wave4"></div>
    <button id="continue-1" class="continue">Continue</button>
    <a href="" class="hover-1" id="hover"></a>
    <a href="" class="hover-2" id="hover"></a>
    <a href="" class="hover-3" id="hover"></a>
</div>

<div id="waiting-screen-confirm-2" class="waitingScreen invisible">
    <div id="header">
        <p>BATTLESHIPS</p>
    </div>
    <div class="wave wave1"></div>
    <div class="wave wave2"></div>
    <div class="wave wave3"></div>
    <div class="wave wave4"></div>
    <button id="continue-2" class="continue">Continue</button>
    <a href="" class="hover-1" id="hover"></a>
    <a href="" class="hover-2" id="hover"></a>
    <a href="" class="hover-3" id="hover"></a>
</div>

<div id="waiting-screen" class="waitingScreen invisible">

    <div id="header">
        <p>BATTLESHIPS</p>
    </div>
    <div class="wave wave1"></div>
    <div class="wave wave2"></div>
    <div class="wave wave3"></div>
    <div class="wave wave4"></div>
    <button id="continue-battle" class="continue">Continue</button>
    <a href="" class="hover-1" id="hover"></a>
    <a href="" class="hover-2" id="hover"></a>
    <a href="" class="hover-3" id="hover"></a>
</div>

</body>
</html>
```

main.js

```
import {oppShipsGenerator, opponentReset} from "./oppShipSetup.js"
let OpponentShipTiles
//LETTER & NUMBER COORDS
export let NumberCoordinates = ['blank', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
export let LetterCoordinates = ['blank', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
//CONTAINERS
let setupContainer = document.querySelector('#setup-container')
let battleContainer = document.querySelector('#container')
let middle = document.querySelector('#middle')
let messageBox = document.querySelector('#message')
let popup = document.querySelector('#popup')
let waitingScreen = document.querySelector('#waiting')
//TILES OF EACH GRID
export let LeftTiles = document.querySelectorAll('[data-tile-left]')
export let RightTiles = document.querySelectorAll('[data-tile-right]')
let SetupTiles = document.querySelectorAll('[data-tile]')
//GRIDS
let SetupGrid = document.querySelector('#setup-grid')
//BTNS FOR SETUP
let RotateBtn = document.querySelector('[data-rotate-btn]')
let ClearBtn = document.querySelector('[data-clear-btn]')
let RemoveBtn = document.querySelector('[data-remove-btn]')
let StartGameBtn = document.querySelector('[data-start-game-btn]')
let shipBtns = document.querySelectorAll('[data-ship-btn]')

//SETUP VARIABLES
let PlayerShipTiles = ['blank']
let LatestShipTiles = []
let PreviewTiles = []
let latestShipBtn = undefined
let overlapCount
let placedCount = 0
let shipLength
let shipName
let isClicked = false
let isVertical = false
let isShipBtn = false
let isValid = true
let isOverlapping = false

//SHIP TILE ARRAYS
let DestroyerOppTiles
let SubmarineOppTiles
let CruiserOppTiles
let BattleshipOppTiles
let CarrierOppTiles
let DestroyerPlayerTiles
let SubmarinePlayerTiles
let CruiserPlayerTiles
let BattleshipPlayerTiles
let CarrierPlayerTiles
```

```
//Variables for Game Logic
let PlayerPrevClicked = ['blank']
let OppPrevClicked = ['blank']
let PlayerClickedTile
let isPlayerTurn
let isWaste
let OppClickedTile
let direction
let workingOnShip = false
let LetterCoordIndex
let NumberCoordIndex
let startNumCoordIndex
let startLetCoordIndex
let validOpp
let reverse
let sinkOppShip
let sinkPlayerShip
let playerShipSinkCount
let oppShipSinkCount
let waiting
let win
let newGameBtn = document.querySelector('[data-new-game]')
let winText = document.querySelector('[data-win-text]')
//Statistics
let playerBombCount = 0
let playerHitCount = 0
let gameCount = 0
let winCount = 0
let leastBombCount = null
let currentBombCount = 0
let accuracy
let winPercent
let accuracyHTML = document.querySelector('[data-accuracy]')
let winPercentHTML = document.querySelector('[data-win-percentage]')
let gamesPlayedHTML = document.querySelector('[data-games-played]')
let bestGameHTML = document.querySelector('[data-best-game]')
let clearStats = document.querySelector('[data-clear]')

//EVENTLISTENER MOUSELEAVE ON GRID
SetupGrid.addEventListener('mouseleave', () =>{
  clearPreviews()
})
function clearPreviews(){
  SetupTiles.forEach(tile =>{
    tile.classList.remove('ship')
  })
}

//EVENTLISTENER ON REMOVE BTN
RemoveBtn.addEventListener('click', () =>{
  PlayerShipTiles.forEach(playerShipTile =>{
    LatestShipTiles.forEach(latestShipTile =>{
      if(playerShipTile.innerHTML == latestShipTile){
```

```
        PlayerShipTiles[PlayerShipTiles.indexOf(playerShipTile)] = 'removed'
      }
    })
  })
  if(placedCount > 0){
    placedCount--
  }
  SetupTiles.forEach(setupTile =>{
    LatestShipTiles.forEach(latestShipTile =>{
      if(latestShipTile == setupTile.innerHTML){
        setupTile.className = ''
        setupTile.classList.add('tile')
      }
    })
  })
  //get latestshipbtn to work and restart after removed
  shipBtns.forEach(shipBtn =>{
    shipBtn.classList.remove('clicked')
    if(shipBtn.className == ''){
      shipBtn.classList.add('default-ship-btn')
    }
    if(shipBtn.innerHTML == latestShipBtn){
      shipBtn.classList.remove('placed')
      shipBtn.classList.add('default-ship-btn')
    }
    resetAfterPlaced()
    StartGameBtn.classList.remove('ready')
    StartGameBtn.classList.add('default-start-btn')
  })
})

//EVENTLISTENER ON CLEAR BTN
ClearBtn.addEventListener('click', () =>{
  PlayerShipTiles = ['blank']
  LatestShipTiles = []
  PreviewTiles = []
  latestShipBtn = undefined
  overlapCount
  placedCount = 0
  shipLength
  shipName
  isClicked = false
  isVertical = false
  isShipBtn = false
  isValid = true
  isOverlapping = false

  //SHIPBTNS CLASSES
  shipBtns.forEach(shipBtn =>{
    shipBtn.className = 'default-ship-btn'
  })

  //Setup Tile Classes
  SetupTiles.forEach(setupTile =>{
    setupTile.className = ''
```

```
        setupTile.classList.add('tile')
    })

    //Startgame btn class
    StartGameBtn.className = 'default-start-btn'
})

//EVENTLISTENER ON ROTATEBTN
RotateBtn.addEventListener('click', () =>{
    isVertical = isVertical !== true
})

//EVENTLISTENER ON STARTBTN
StartGameBtn.addEventListener('click', () =>{
    if(StartGameBtn.classList.contains('ready')){
        setupContainer.classList.add('invisible')
        battleContainer.classList.remove('invisible')
        startGame()
    }
})

//EVENTLISTENER ON SHIP BTN
document.addEventListener('click', (e) =>{
    isShipBtn = e.target.matches('[data-ship-btn]')
    if(isShipBtn && !e.target.classList.contains('placed')){
        document.querySelectorAll('[data-ship-btn]').forEach(shipBtn =>{
            shipBtn.classList.remove('clicked')
            if(shipBtn.className == ''){
                shipBtn.classList.add('default-ship-btn')
            }
        })
        e.target.classList.remove('default-ship-btn')
        e.target.classList.add('clicked')
        shipName = e.target.innerHTML
        isClicked = true
        getShipLength()
    }
})

//GET SHIP LENGTH
function getShipLength(){
    switch (shipName) {
        case "Destroyer":
            shipLength = 2
            break;
        case "Submarine":
            shipLength = 3
            break;
        case "Cruiser":
            shipLength = 3
            break;
        case "Battleship":
            shipLength = 4
            break;
        case "Carrier":
            shipLength = 5
```

```
        break;
    default:
        break;
    }
}

//EVENTLISTENER ON TILE MOUSEOVER
SetupTiles.forEach(tile =>{
    tile.addEventListener('mouseover', (TileIndex, LetterCoordIndex, NumberCoordIndex) => {
        //REMOVE CLASS FROM EACH TILE
        SetupTiles.forEach(tile => {
            tile.classList.remove('ship')
        })
        overlapCount = 0

        //GET TILE INDEX IN ARRAY
        for(let i = 0; i <= SetupTiles.length; i++){
            if(tile == SetupTiles[i]){
                TileIndex = i
                PreviewTiles = []
            }
        }
        //GET PREVIEW TILES --- Horizontal
        if(!isVertical){
            for(let i=0; i<shipLength;i++){
                PreviewTiles.push(SetupTiles[TileIndex+i].innerHTML)
            }
        }

        //GET PREVIEW TILES --- Vertical
        if(isVertical){
            for(let i=0; i<shipLength*10;i =i+10){
                PreviewTiles.push(SetupTiles[TileIndex+i].innerHTML)
            }
        }

        //GET LETTERCOORD in Array (index)
        for(let i = 0; i <LetterCoordinates.length; i++){
            if(LetterCoordinates[i] == tile.innerHTML.slice(0,1)){
                LetterCoordIndex = i
            }
        }

        //GET NUMCOORD in ARRAY (index)
        for(let i = 0; i <NumberCoordinates.length; i++){
            if(NumberCoordinates[i] == tile.innerHTML.slice(1,3)){
                NumberCoordIndex = i
            }
        }

        //CHECK IF VALID --- Horizontal
        if(!isVertical){
            if(isClicked && LetterCoordIndex + shipLength <=11){
                isValid=true
            }
        }
    })
}
```



```
}else{
    isValid = false
}
}

//CHECK IF VALID --- Vertical
if(isVertical){
    if(isClicked && NumberCoordIndex + shipLength <=11){
        isValid = true
    }else{
        isValid = false
    }
}

//CHECK OVERLAP -- Horizontal
if(!isVertical){
    PlayerShipTiles.forEach(playerTile =>{
        PreviewTiles.forEach(previewTile =>{
            if(previewTile == playerTile.innerHTML){
                isOverlapping = true
                overlapCount++
                console.log('overlap')
            }
        })
        if(overlapCount>0){
            isOverlapping = false
        }
    })
}

//CHECK OVERLAP -- Vertical
if(isVertical){
    PlayerShipTiles.forEach(playerTile =>{
        PreviewTiles.forEach(previewTile =>{
            if(previewTile == playerTile.innerHTML){
                isOverlapping = true
                overlapCount++
                console.log('overlap')
            }
        })
        if(overlapCount>0){
            isOverlapping = false
        }
    })
}

//ADD CLASS TO EACH SHIP TILE --- Horizontal
if(isClicked && !isVertical && isValid){
    if(!isOverlapping){
        for(let i = 0; i < shipLength; i++){
            SetupTiles[TileIndex + i].classList.add('ship')
        }
    }
}
```

```
//ADD CLASS TO EACH SHIP TILE --- Vertical
if(isClicked && isVertical && isValid){
    if(!isOverlapping){
        for(let i = 0; i < shipLength * 10; i = i + 10){
            SetupTiles[TileIndex + i].classList.add('ship')
        }
    }
}

})

})

//EVENTLISTENER ON TILE CLICK
SetupTiles.forEach(tile =>{
    tile.addEventListener('click', () =>{
        LatestShipTiles = []
        if(isValid && !isOverlapping){
            playClick()
            latestShipBtn = shipName
            document.querySelectorAll('.ship').forEach(shipTile =>{
                PlayerShipTiles.push(shipTile)
                LatestShipTiles.push(shipTile.innerHTML)
                shipTile.classList.add('shipClicked')
                switch (shipName) {
                    case "Destroyer":
                        for(let i = 0; i < shipLength; i++){
                            document.querySelectorAll('.ship').forEach(shipTile =>{
                                shipTile.classList.add('destroyer')
                            })
                        }
                        break;
                    case "Submarine":
                        for(let i = 0; i < shipLength; i++){
                            document.querySelectorAll('.ship').forEach(shipTile =>{
                                shipTile.classList.add('submarine')
                            })
                        }
                        break;
                    case "Cruiser":
                        for(let i = 0; i < shipLength; i++){
                            document.querySelectorAll('.ship').forEach(shipTile =>{
                                shipTile.classList.add('cruiser')
                            })
                        }
                        break;
                    case "Battleship":
                        for(let i = 0; i < shipLength; i++){
                            document.querySelectorAll('.ship').forEach(shipTile =>{
                                shipTile.classList.add('battleship')
                            })
                        }
                        break;
                    case "Carrier":
                        for(let i = 0; i < shipLength; i++){
                            document.querySelectorAll('.ship').forEach(shipTile =>{
```

```
        shipTile.classList.add('carrier')
    })
    }
    break;
default:
    break;
}
resetAfterPlaced()
})
placedCount++
if(placedCount >= 5){
    StartGameBtn.classList.add('ready')
    StartGameBtn.classList.remove('default-start-btn')
}
document.querySelector('.clicked').classList.add('placed')
document.querySelector('.clicked').classList.remove('clicked')
}

})
})

function resetAfterPlaced(){
    isClicked = false
    isValid = true
    isShipBtn = false
    isVertical = false
    shipLength = undefined
    shipName = undefined
    isOverlapping = false
}

//START GAME
function startGame(){
    oppShipsGenerator()
    assignValues()
    addPlayerShipClasses()
    isPlayerTurn = true
    workingOnShip = false
    isOpposite = false
    reverse = false
}
function assignValues(){
    DestroyerOppTiles = document.querySelectorAll('.destroyerOpp')
    SubmarineOppTiles = document.querySelectorAll('.submarineOpp')
    CruiserOppTiles = document.querySelectorAll('.cruiserOpp')
    BattleshipOppTiles = document.querySelectorAll('.battleshipOpp')
    CarrierOppTiles = document.querySelectorAll('.carrierOpp')
    OpponentShipTiles = document.querySelectorAll('.shipOpp')
    DestroyerPlayerTiles = document.querySelectorAll('.destroyer')
    SubmarinePlayerTiles = document.querySelectorAll('.submarine')
    CruiserPlayerTiles = document.querySelectorAll('.cruiser')
    BattleshipPlayerTiles = document.querySelectorAll('.battleship')
    CarrierPlayerTiles = document.querySelectorAll('.carrier')
    direction = undefined
}
```

```
waiting = false
win = false
playerShipSinkCount = 0
oppShipSinkCount = 0
LeftTiles.forEach(LeftTile =>{
    PlayerShipTiles.forEach(PlayerShipTile =>{
        if(PlayerShipTile.innerHTML == LeftTile.innerHTML){
            LeftTile.classList.add('playerShip')
        }
    })
})
PlayerShipTiles = []
document.querySelectorAll('.playerShip').forEach(playerShip =>{
    PlayerShipTiles.push(playerShip)
})
}

function addPlayerShipClasses(){
    LeftTiles.forEach(leftTile =>{
        DestroyerPlayerTiles.forEach(shipTile =>{
            if(shipTile.innerHTML == leftTile.innerHTML){
                leftTile.classList.add('playerDestroyer')
            }
        })
    })
    LeftTiles.forEach(leftTile =>{
        SubmarinePlayerTiles.forEach(shipTile =>{
            if(shipTile.innerHTML == leftTile.innerHTML){
                leftTile.classList.add('playerSubmarine')
            }
        })
    })
    LeftTiles.forEach(leftTile =>{
        CruiserPlayerTiles.forEach(shipTile =>{
            if(shipTile.innerHTML == leftTile.innerHTML){
                leftTile.classList.add('playerCruiser')
            }
        })
    })
    LeftTiles.forEach(leftTile =>{
        BattleshipPlayerTiles.forEach(shipTile =>{
            if(shipTile.innerHTML == leftTile.innerHTML){
                leftTile.classList.add('playerBattleship')
            }
        })
    })
    LeftTiles.forEach(leftTile =>{
        CarrierPlayerTiles.forEach(shipTile =>{
            if(shipTile.innerHTML == leftTile.innerHTML){
                leftTile.classList.add('playerCarrier')
            }
        })
    })
    DestroyerPlayerTiles = document.querySelectorAll('.playerDestroyer')
    SubmarinePlayerTiles = document.querySelectorAll('.playerSubmarine')
```

```
CruiserPlayerTiles = document.querySelectorAll('.playerCruiser')
BattleshipPlayerTiles = document.querySelectorAll('.playerBattleship')
CarrierPlayerTiles = document.querySelectorAll('.playerCarrier')
}

/////GAME LOGIC
RightTiles.forEach(rightTile =>{
  rightTile.addEventListener('click', (e) =>{
    console.log(playerShipSinkCount, oppShipSinkCount)
    if(!waiting){
      PlayerClickedTile = e.target.innerHTML
      if(isPlayerTurn){
        PlayerPrevClicked.forEach(playerprevclickedtile =>{
          if(PlayerClickedTile == playerprevclickedtile){
            isWaste = true
            //alert
          }
        })
      }
      if(isPlayerTurn && !isWaste){
        PlayerPrevClicked.push(PlayerClickedTile)
        rightTile.classList.add('bomb')
        checkHit()
      }
      if(!isWaste && isPlayerTurn){
        isPlayerTurn = false
        setTimeout(() => {
          oppTurn()
        }, 300);
      }
      isWaste = false
    }
  })
})

function checkHit(currentHitOpp){
  if(isPlayerTurn){
    playerBombCount++
    currentBombCount++
    OpponentShipTiles.forEach(OpponentShipTile =>{
      if(PlayerClickedTile == OpponentShipTile.innerHTML){
        playerHitCount++
        OpponentShipTile.classList.add('tileSink')
        OpponentShipTile.classList.remove('bomb')
        checkSink()
      }
    })
  }
  if(!isPlayerTurn){
    currentHitOpp = false
    PlayerShipTiles.forEach(playerShipTile =>{
      if(playerShipTile.innerHTML == OppClickedTile){
        playerShipTile.classList.add('tileSink')
        playerShipTile.classList.remove('bomb')
        if(!workingOnShip){
```

```
        startLetCoordIndex = LetterCoordIndex
        startNumCoordIndex = NumberCoordIndex
    }
    currentHitOpp = true
    workingOnShip = true
    checkSink()
}
})
if(!currentHitOpp && workingOnShip){
    reverse = true
}
}
}

function checkSink(){
    if(isPlayerTurn){
        sinkOppShip = false
        destroyerOppSink()
        submarineOppSink()
        cruiserOppSink()
        battleshipOppSink()
        carrierOppSink()
        //ALERT PLAYER SUNK OPP SHIP
        if(sinkOppShip){
            middle.classList.remove('invisible')
            playBomb()
            checkWin()
            isPlayerTurn = false
            setTimeout(() => {
                if(!win){
                    middle.classList.add('invisible')
                    setTimeout(() => {
                        oppTurn()
                    }, 300);
                }
                messageBox.innerHTML = ''
            }, 2000);
        }
    }
    if(!isPlayerTurn){
        sinkPlayerShip = false
        destroyerPlayerSink()
        submarinePlayerSink()
        cruiserPlayerSink()
        battleshipPlayerSink()
        carrierPlayerSink()
        if(sinkPlayerShip){
            middle.classList.remove('invisible')
            workingOnShip = false
            reverse = false
            direction = undefined
            playBomb()
            checkWin()
            isPlayerTurn = false
            waiting = true
        }
    }
}
```

```
        setTimeout(() => {
            if(!win){
                middle.classList.add('invisible')
            }
            setTimeout(() => {
                isPlayerTurn = true
                waiting = false
                messageBox.innerHTML = ''
            }, 300);
        }, 2000);
    }
}
}
```

```
function checkWin(){
    if(oppShipSinkCount == 5){
        gameCount++
        winCount++
        popupFunction()
        statistics()
        isPlayerTurn = false
        winText.innerHTML = 'You won'
        console.log('playerWin')
    }
    if(playerShipSinkCount == 5){
        gameCount++
        popupFunction()
        statistics()
        isPlayerTurn = false
        winText.innerHTML = 'You lost'
        console.log('oppWin')
    }
}
```

```
function popupFunction(){
    middle.classList.add('background')
    middle.classList.remove('invisible')
    popup.classList.remove('invisible')
    win = true
}
```

```
function statistics(){
    console.log(playerHitCount, playerBombCount, gameCount, winCount)
    accuracy = Math.round(playerHitCount/playerBombCount * 100) + '%'
    winPercent = Math.round(winCount/gameCount * 100) + '%'
    if(leastBombCount < currentBombCount || leastBombCount == null){
        leastBombCount = currentBombCount
    }
    localStorage.setItem('leastBombCount', leastBombCount)
    localStorage.setItem('accuracy', accuracy)
    localStorage.setItem('winPercent', winPercent)
    localStorage.setItem('gameCount', gameCount)
    localStorage.setItem('playerHitCount', playerHitCount)
    localStorage.setItem('playerBombCount', playerBombCount)
    localStorage.setItem('winCount', winCount)
```

```
}
function oppTurn(){
  if(reverse){
    reverseDir()
  }
  if(!isPlayerTurn && !workingOnShip){
    if(!reverse){
      getRandTile()
    }
  }
  if(workingOnShip && direction !== undefined){
    if(!reverse){
      addDirection()
    }
  }
  if(workingOnShip && direction === undefined){
    if(!reverse){
      getRandDir()
      addDirection()
    }
  }
  checkValid()
  if(!validOpp){
    workingOnShip = false
    direction = undefined
    getRandTile()
  }

  for(let i = 0; i<=LetterCoordinates.length; i++){
    if(OppClickedTile.slice(0,1) == LetterCoordinates[i]){
      LetterCoordIndex = i
    }
  }
  for(let i = 0; i<=NumberCoordinates.length; i++){
    if(OppClickedTile.slice(1,3) == NumberCoordinates[i]){
      NumberCoordIndex = i
    }
  }

  reverse = false
  //add bomb class
  LeftTiles.forEach(LeftTile =>{
    if(LeftTile.innerHTML == OppClickedTile){
      LeftTile.classList.add('bomb')
    }
  })
  //add tile to prev clicked
  OppPrevClicked.push(OppClickedTile)
  checkHit()

  waitingScreen.classList.remove('invisible')
  setTimeout(() => {
    isPlayerTurn = true
    waitingScreen.classList.add('invisible')
  }, 1000)
```



```
    }, 300);
}

function addDirection(){
    switch (direction) {
        case 1:
            OppClickedTile =
LetterCoordinates[LetterCoordIndex]+NumberCoordinates[NumberCoordIndex-1]
        case 2:
            OppClickedTile =
LetterCoordinates[LetterCoordIndex+1]+NumberCoordinates[NumberCoordIndex]
            break;
        case 3:
            OppClickedTile =
LetterCoordinates[LetterCoordIndex]+NumberCoordinates[NumberCoordIndex+1]
            break;
        case 4:
            OppClickedTile = LetterCoordinates[LetterCoordIndex-
1]+NumberCoordinates[NumberCoordIndex]
            break;
        default:
            break;
    }
}

function getRandDir(){
    direction = Math.floor(Math.random()*4+1)
}

function getRandTile(repeat){
    repeat = false
    OppClickedTile = LetterCoordinates[Math.floor(Math.random()*10+1)] +
NumberCoordinates[Math.floor(Math.random()*10+1)]
    OppPrevClicked.forEach(prevClicked =>{
        if(OppClickedTile == prevClicked){
            repeat = true
        }
    })
    if(repeat){
        getRandTile()
    }
}

function checkValid(boardTile, prevClickedTileOpp){
    validOpp = false
    boardTile = false
    prevClickedTileOpp = false
    LeftTiles.forEach(leftTile =>{
        if(leftTile.innerHTML == OppClickedTile){
            boardTile = true
        }
    })
    OppPrevClicked.forEach(prevClicked =>{
        if(prevClicked == OppClickedTile){
            prevClickedTileOpp = true
        }
    })
}
```

```
    })
    if(boardTile && !prevClickedTileOpp){
        validOpp = true
    }
}

function reverseDir(){
    switch (direction) {
        case 1:
            direction = 3
            OppClickedTile =
LetterCoordinates[startLetCoordIndex]+NumberCoordinates[startNumCoordIndex+1]
        case 2:
            direction = 4
            OppClickedTile = LetterCoordinates[startLetCoordIndex-
1]+NumberCoordinates[startNumCoordIndex]
            break;
        case 3:
            direction = 1
            OppClickedTile =
LetterCoordinates[startLetCoordIndex]+NumberCoordinates[startNumCoordIndex-1]
            break;
        case 4:
            direction = 2
            OppClickedTile =
LetterCoordinates[startLetCoordIndex+1]+NumberCoordinates[startNumCoordIndex]
            break;
        default:
            break;
    }
}

//SINK FUNCTIONS
function destroyerOppSink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    DestroyerOppTiles.forEach(destroyerOppTile =>{
        if(destroyerOppTile.classList.contains('tileSink')){
            count++
        }
        if(destroyerOppTile.innerHTML == PlayerClickedTile){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == DestroyerOppTiles.length){
        sinkOppShip = true
        currentShipSink = true
    }
    if(currentShipSink){
        DestroyerOppTiles.forEach(oppTile =>{
            oppTile.classList.add('shipSink')
            oppTile.classList.remove('tileSink')
        })
        oppShipSinkCount++
    }
}
```

```
        messageBox.innerHTML = "You sank the opponent's Destroyer"
    }
}
function submarineOppSink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    SubmarineOppTiles.forEach(submarineOppTile =>{
        if(submarineOppTile.classList.contains('tileSink')){
            count++
        }
        if(submarineOppTile.innerHTML == PlayerClickedTile){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == SubmarineOppTiles.length){
        sinkOppShip = true
        currentShipSink = true
    }
    if(currentShipSink){
        SubmarineOppTiles.forEach(oppTile =>{
            oppTile.classList.add('shipSink')
            oppTile.classList.remove('tileSink')
        })
        oppShipSinkCount++
        messageBox.innerHTML = "You sank the opponent's Submarine"
    }
}
function cruiserOppSink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    CruiserOppTiles.forEach(cruiserOppTile =>{
        if(cruiserOppTile.classList.contains('tileSink')){
            count++
        }
        if(cruiserOppTile.innerHTML == PlayerClickedTile){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == CruiserOppTiles.length){
        sinkOppShip = true
        currentShipSink = true
    }
    if(currentShipSink){
        CruiserOppTiles.forEach(oppTile =>{
            oppTile.classList.add('shipSink')
            oppTile.classList.remove('tileSink')
        })
        oppShipSinkCount++
        messageBox.innerHTML = "You sank the opponent's Cruiser"
    }
}
function battleshipOppSink(belongsToShip, count, currentShipSink){
```

```
count = 0
belongsToShip = false
currentShipSink = false
BattleshipOppTiles.forEach(battleshipOppTile =>{
    if(battleshipOppTile.classList.contains('tileSink')){
        count++
    }
    if(battleshipOppTile.innerHTML == PlayerClickedTile){
        belongsToShip = true
    }
})
if(belongsToShip && count == BattleshipOppTiles.length){
    sinkOppShip = true
    currentShipSink = true
}
if(currentShipSink){
    BattleshipOppTiles.forEach(oppTile =>{
        oppTile.classList.add('shipSink')
        oppTile.classList.remove('tileSink')
    })
    oppShipSinkCount++
    messageBox.innerHTML = "You sank the opponent's Battleship"
}
}

function carrierOppSink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    CarrierOppTiles.forEach(battleshipOppTile =>{
        if(battleshipOppTile.classList.contains('tileSink')){
            count++
        }
        if(battleshipOppTile.innerHTML == PlayerClickedTile){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == CarrierOppTiles.length){
        sinkOppShip = true
        currentShipSink = true
    }
    if(currentShipSink){
        CarrierOppTiles.forEach(oppTile =>{
            oppTile.classList.add('shipSink')
            oppTile.classList.remove('tileSink')
        })
        oppShipSinkCount++
        messageBox.innerHTML = "You sank the opponent's Carrier"
    }
}

function destroyerPlayerSink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    DestroyerPlayerTiles.forEach(destroyerPlayerTile =>{
```

```
        if(destroyerPlayerTile.classList.contains('tileSink')){
            count++
        }
        if(destroyerPlayerTile.innerHTML == OppClickedTile){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == DestroyerPlayerTiles.length){
        sinkPlayerShip = true
        currentShipSink = true
    }
    if(currentShipSink){
        DestroyerPlayerTiles.forEach(oppTile =>{
            oppTile.classList.add('shipSink')
            oppTile.classList.remove('tileSink')
        })
        playerShipSinkCount++
        messageBox.innerHTML = "The opponent sank your Destroyer"
    }
}

function submarinePlayerSink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    SubmarinePlayerTiles.forEach(submarinePlayerTile =>{
        if(submarinePlayerTile.classList.contains('tileSink')){
            count++
        }
        if(submarinePlayerTile.innerHTML == OppClickedTile){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == SubmarinePlayerTiles.length){
        sinkPlayerShip = true
        currentShipSink = true
    }
    if(currentShipSink){
        SubmarinePlayerTiles.forEach(oppTile =>{
            oppTile.classList.add('shipSink')
            oppTile.classList.remove('tileSink')
        })
        playerShipSinkCount++
        messageBox.innerHTML = "The opponent sank your Submarine"
    }
}

function cruiserPlayerSink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    CruiserPlayerTiles.forEach(cruiserPlayerTile =>{
        if(cruiserPlayerTile.classList.contains('tileSink')){
            count++
        }
        if(cruiserPlayerTile.innerHTML == OppClickedTile){
            belongsToShip = true
        }
    })
}
```

```
    }
  })
  if(belongsToShip && count == CruiserPlayerTiles.length){
    sinkPlayerShip = true
    currentShipSink = true
  }
  if(currentShipSink){
    CruiserPlayerTiles.forEach(oppTile =>{
      oppTile.classList.add('shipSink')
      oppTile.classList.remove('tileSink')
    })
    playerShipSinkCount++
    messageBox.innerHTML = "The opponent sank your Cruiser"
  }
}

function battleshipPlayerSink(belongsToShip, count, currentShipSink){
  count = 0
  belongsToShip = false
  currentShipSink = false
  BattleshipPlayerTiles.forEach(battleshipPlayerTile =>{
    if(battleshipPlayerTile.classList.contains('tileSink')){
      count++
    }
    if(battleshipPlayerTile.innerHTML == OppClickedTile){
      belongsToShip = true
    }
  })
  if(belongsToShip && count == BattleshipPlayerTiles.length){
    sinkPlayerShip = true
    currentShipSink = true
  }
  if(currentShipSink){
    BattleshipPlayerTiles.forEach(oppTile =>{
      oppTile.classList.add('shipSink')
      oppTile.classList.remove('tileSink')
    })
    playerShipSinkCount++
    messageBox.innerHTML = "The opponent sank your Battleship"
  }
}

function carrierPlayerSink(belongsToShip, count, currentShipSink){
  count = 0
  belongsToShip = false
  currentShipSink = false
  CarrierPlayerTiles.forEach(carrierPlayerTile =>{
    if(carrierPlayerTile.classList.contains('tileSink')){
      count++
    }
    if(carrierPlayerTile.innerHTML == OppClickedTile){
      belongsToShip = true
    }
  })
  if(belongsToShip && count == CarrierPlayerTiles.length){
    sinkPlayerShip = true
    currentShipSink = true
  }
}
```

```

    }
    if(currentShipSink){
        CarrierPlayerTiles.forEach(oppTile =>{
            oppTile.classList.add('shipSink')
            oppTile.classList.remove('tileSink')
        })
        playerShipSinkCount++
        messageBox.innerHTML = "The opponent sank your Carrier"
    }
}

function playBomb(){
    var audio = new Audio('./stuff/audio.wav')
    audio.play()
}
function playClick(){
    var audio = new Audio('./stuff/click.wav')
    audio.play()
}

window.addEventListener('load', () =>{
    accuracyHTML.innerHTML = 'Hit Accuracy: ' + localStorage.getItem('accuracy')
    winPercentHTML.innerHTML = 'Win Percentage: ' + localStorage.getItem('winPercent')
    gamesPlayedHTML.innerHTML = 'Games Played: ' + localStorage.getItem('gameCount')
    bestGameHTML.innerHTML = 'Best Game: ' + localStorage.getItem('leastBombCount') + ' Hits'
    playerBombCount = localStorage.getItem('playerBombCount')
    playerHitCount = localStorage.getItem('playerHitCount')
    gameCount = localStorage.getItem('gameCount')
    winCount = localStorage.getItem('winCount')
})

clearStats.addEventListener('click', () =>{
    window.localStorage.clear()
})

```

multiplayer.js

```

const NumberCoordinates = ['blank', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
const LetterCoordinates = ['blank', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']

let player1SetupTiles = document.querySelectorAll('[data-tile-player-1]')
let player2SetupTiles = document.querySelectorAll('[data-tile-player-2]')
const setupGrids = document.querySelectorAll('#setup-grid')
const setupContainer1 = document.querySelector('#setup-container-player-1')
const setupContainer2 = document.querySelector('#setup-container-player-2')

const battleContainer = document.querySelector('#container')
const leftTiles = document.querySelectorAll('[data-tile-left]')
const rightTiles = document.querySelectorAll('[data-tile-right]')

let middle = document.querySelector('#middle')
let messageBox = document.querySelector('[data-win-text]')

```

```
const waitingConf1 = document.querySelector('#waiting-screen-confirm-1')
const waitingConf2 = document.querySelector('#waiting-screen-confirm-2')
const waitingBattle = document.querySelector('#waiting-screen')
const continue1 = document.querySelector('#continue-1')
const continue2 = document.querySelector('#continue-2')
const continueBattle = document.querySelector('#continue-battle')

const removeBtn1 = document.querySelector('[data-remove-btn-player-1]')
const removeBtn2 = document.querySelector('[data-remove-btn-player-2]')
const rotateBtn1 = document.querySelector('[data-rotate-btn-player-1]')
const rotateBtn2 = document.querySelector('[data-rotate-btn-player-2]')
const clearBtn1 = document.querySelector('[data-clear-btn-player-1]')
const clearBtn2 = document.querySelector('[data-clear-btn-player-2]')
const confirm1 = document.querySelector('[data-confirm-player-1]')
const confirm2 = document.querySelector('[data-confirm-player-2]')
const shipBtns1 = document.querySelectorAll('[data-ship-btn-player-1]')
const shipBtns2 = document.querySelectorAll('[data-ship-btn-player-2]')

let latestShipBtn = undefined
let overlapCount
let placedCount = 0
let shipLength
let shipName
let isClicked = false
let isVertical = false
let isShipBtn = false
let isValid = true
let isOverlapping = false

let player1ShipTiles = ['blank']
let player2ShipTiles = ['blank']
let latestShipTiles = []
let previewTiles = []

let destroyerPlayer1 = []
let submarinePlayer1 = []
let cruiserPlayer1 = []
let battleshipPlayer1 = []
let carrierPlayer1 = []
let destroyerPlayer2 = []
let submarinePlayer2 = []
let cruiserPlayer2 = []
let battleshipPlayer2 = []
let carrierPlayer2 = []

let player1PrevClicked = []
let player2PrevClicked = []
let player1Turn
let isWaste
let player1ShipSinkCount
let player2ShipSinkCount
let player1Current
let player2Current
let waiting
```



```
let win

let sinkPlayer1Ship
let sinkPlayer2Ship

//Event listener mouseleave
setupGrids.forEach(setupGrid =>{
  setupGrid.addEventListener('mouseleave', () =>{
    clearPreviews()
  })
})
function clearPreviews(){
  player1SetupTiles.forEach(tile =>{
    tile.classList.remove('ship')
  })
  player2SetupTiles.forEach(tile =>{
    tile.classList.remove('ship')
  })
}

//EventListener remove 1
removeBtn1.addEventListener('click', () =>{
  player1ShipTiles.forEach(player1ShipTile =>{
    latestShipTiles.forEach(latestShipTile =>{
      if(player1ShipTile.innerHTML == latestShipTile){
        player1ShipTiles[player1ShipTiles.indexOf(player1ShipTile)] = 'removed'
      }
    })
  })
  if(placedCount > 0){
    placedCount--
  }
  player1SetupTiles.forEach(setupTile =>{
    latestShipTiles.forEach(latestShipTile =>{
      if(latestShipTile == setupTile.innerHTML){
        setupTile.className = ''
        setupTile.classList.add('tile')
      }
    })
  })
  shipBtns1.forEach(shipBtn =>{
    shipBtn.classList.remove('clicked')
    if(shipBtn.className == ''){
      shipBtn.classList.add('default-ship-btn')
    }
    if(shipBtn.innerHTML == latestShipBtn){
      shipBtn.classList.remove('placed')
      shipBtn.classList.add('default-ship-btn')
    }
    resetAfterPlaced()
    confirm1.classList.remove('ready')
    confirm1.classList.add('default-start-btn')
  })
})
})
```

```
//Event listener remove 2
removeBtn2.addEventListener('click', () =>{
  player2ShipTiles.forEach(player1ShipTile =>{
    latestShipTiles.forEach(latestShipTile =>{
      if(player1ShipTile.innerHTML == latestShipTile){
        player2ShipTiles[player2ShipTiles.indexOf(player1ShipTile)] = 'removed'
      }
    })
  })
})
if(placedCount > 0){
  placedCount--
}
player2SetupTiles.forEach(setupTile =>{
  latestShipTiles.forEach(latestShipTile =>{
    if(latestShipTile == setupTile.innerHTML){
      setupTile.className = ''
      setupTile.classList.add('tile')
    }
  })
})
shipBtns2.forEach(shipBtn =>{
  shipBtn.classList.remove('clicked')
  if(shipBtn.className == ''){
    shipBtn.classList.add('default-ship-btn')
  }
  if(shipBtn.innerHTML == latestShipBtn){
    shipBtn.classList.remove('placed')
    shipBtn.classList.add('default-ship-btn')
  }
  resetAfterPlaced()
  confirm2.classList.remove('ready')
  confirm2.classList.add('default-start-btn')
})
})

//Event listener Clear 1
clearBtn1.addEventListener('click', () =>{
  player1ShipTiles = ['blank']
  latestShipTiles = []
  previewTiles = []
  latestShipBtn = undefined
  overlapCount
  placedCount = 0
  shipLength
  shipName
  isClicked = false
  isVertical = false
  isShipBtn = false
  isValid = true
  isOverlapping = false

  //SHIPBTNS CLASSES
  shipBtns1.forEach(shipBtn =>{
    shipBtn.className = 'default-ship-btn'
```

```
    })

    //Setup Tile Classes
    player1SetupTiles.forEach(setupTile =>{
        setupTile.className = ''
        setupTile.classList.add('tile')
    })

    //Startgame btn class
    confirm1.className = 'default-start-btn'
})

//Event listener Clear 2
clearBtn2.addEventListener('click', () =>{
    player2ShipTiles = ['blank']
    latestShipTiles = []
    previewTiles = []
    latestShipBtn = undefined
    overlapCount
    placedCount = 0
    shipLength
    shipName
    isClicked = false
    isVertical = false
    isShipBtn = false
    isValid = true
    isOverlapping = false

    //SHIPBTNS CLASSES
    shipBtns2.forEach(shipBtn =>{
        shipBtn.className = 'default-ship-btn'
    })

    //Setup Tile Classes
    player2SetupTiles.forEach(setupTile =>{
        setupTile.className = ''
        setupTile.classList.add('tile')
    })

    //Startgame btn class
    confirm2.className = 'default-start-btn'
})

//Event listener Rotate 1
rotateBtn1.addEventListener('click', () =>{
    isVertical = isVertical !== true
})

//Event listener Rotate 2
rotateBtn2.addEventListener('click', () =>{
    isVertical = isVertical !== true
})

//Event listener Confirm 1
confirm1.addEventListener('click', () =>{
```

```
if(confirm1.classList.contains('ready')){
    setupContainer1.classList.add('invisible')
    waitingConf1.classList.remove('invisible')
    destroyerPlayer1 = document.querySelectorAll('.destroyer-1')
    submarinePlayer1 = document.querySelectorAll('.submarine-1')
    cruiserPlayer1 = document.querySelectorAll('.cruiser-1')
    battleshipPlayer1 = document.querySelectorAll('.battleship-1')
    carrierPlayer1 = document.querySelectorAll('.carrier-1')
    placedCount = 0
}
})
continue1.addEventListener('click', () =>{
    continue1.classList.add('hover')
    setTimeout(() => {
        continue1.classList.add('animate__animated', 'animate__bounceOutUp')
    }, 900);
    setTimeout(() => {
        setupContainer2.classList.remove('invisible')
        waitingConf1.classList.add('invisible')
    }, 1600);
})

//Event listener Confirm 2
confirm2.addEventListener('click', () =>{
    if(confirm2.classList.contains('ready')){
        waitingConf2.classList.remove('invisible')
        setupContainer2.classList.add('invisible')
        destroyerPlayer2 = document.querySelectorAll('.destroyer-2')
        submarinePlayer2 = document.querySelectorAll('.submarine-2')
        cruiserPlayer2 = document.querySelectorAll('.cruiser-2')
        battleshipPlayer2 = document.querySelectorAll('.battleship-2')
        carrierPlayer2 = document.querySelectorAll('.carrier-2')
        startGame()
    }
})
continue2.addEventListener('click', () =>{
    continue2.classList.add('hover')
    setTimeout(() => {
        continue2.classList.add('animate__animated', 'animate__bounceOutUp')
    }, 900);
    setTimeout(() => {
        battleContainer.classList.remove('invisible')
        waitingConf2.classList.add('invisible')
    }, 1600);
})

//EVENTLISTENER ON SHIP BTN
document.addEventListener('click', (e) =>{
    isShipBtn = e.target.matches('[data-ship-btn]')
    console.log(isShipBtn)
    if(isShipBtn && !e.target.classList.contains('placed')){
        shipBtns1.forEach(shipBtn =>{
            shipBtn.classList.remove('clicked')
            if(shipBtn.className == ''){
                shipBtn.classList.add('default-ship-btn')
            }
        })
    }
})
```

```
    })
    shipBtns2.forEach(shipBtn =>{
        shipBtn.classList.remove('clicked')
        if(shipBtn.className == ''){
            shipBtn.classList.add('default-ship-btn')
        }
    })
    e.target.classList.remove('default-ship-btn')
    e.target.classList.add('clicked')
    shipName = e.target.innerHTML
    isClicked = true
    getShipLength()
}
}))

//GET SHIP LENGTH
function getShipLength(){
    switch (shipName) {
        case "Destroyer":
            shipLength = 2
            break;
        case "Submarine":
            shipLength = 3
            break;
        case "Cruiser":
            shipLength = 3
            break;
        case "Battleship":
            shipLength = 4
            break;
        case "Carrier":
            shipLength = 5
            break;
        default:
            break;
    }
}

//EVENTLISTENER ON TILE MOUSEOVER Player 1
player1SetupTiles.forEach(tile =>{
    tile.addEventListener('mouseover', (TileIndex, LetterCoordIndex, NumberCoordIndex) => {
        //REMOVE CLASS FROM EACH TILE
        player1SetupTiles.forEach(tile => {
            tile.classList.remove('ship')
        })
        overlapCount = 0

        //GET TILE INDEX IN ARRAY
        for(let i = 0; i <= player1SetupTiles.length; i++){
            if(tile == player1SetupTiles[i]){
                TileIndex = i
                previewTiles = []
            }
        }
    })
    //GET PREVIEW TILES --- Horizontal
```

```
if(!isVertical){
    for(let i=0; i<shipLength;i++){
        previewTiles.push(player1SetupTiles[TileIndex+i].innerHTML)
    }
}

//GET PREVIEW TILES --- Vertical
if(isVertical){
    for(let i=0; i<shipLength*10;i =i+10){
        previewTiles.push(player1SetupTiles[TileIndex+i].innerHTML)
    }
}

//GET LETTERCOORD in Array (index)
for(let i = 0; i <LetterCoordinates.length; i++){
    if(LetterCoordinates[i] == tile.innerHTML.slice(0,1)){
        LetterCoordIndex = i
    }
}

//GET NUMCOORD in ARRAY (index)
for(let i = 0; i <NumberCoordinates.length; i++){
    if(NumberCoordinates[i] == tile.innerHTML.slice(1,3)){
        NumberCoordIndex = i
    }
}

//CHECK IF VALID --- Horizontal
if(!isVertical){
    if(isClicked && LetterCoordIndex + shipLength <=11){
        isValid=true
    }else{
        isValid = false
    }
}

//CHECK IF VALID --- Vertical
if(isVertical){
    if(isClicked && NumberCoordIndex + shipLength <=11){
        isValid = true
    }else{
        isValid = false
    }
}

//CHECK OVERLAP -- Horizontal
if(!isVertical){
    player1ShipTiles.forEach(playerTile =>{
        previewTiles.forEach(previewTile =>{
            if(previewTile == playerTile.innerHTML){
                isOverlapping = true
                overlapCount++
                console.log('overlap')
            }
        })
    })
}
```

```
        if(overlapCount<=0){
            isOverlapping = false
        }
    })
}

//CHECK OVERLAP -- Vertical
if(isVertical){
    player1ShipTiles.forEach(playerTile =>{
        previewTiles.forEach(previewTile =>{
            if(previewTile == playerTile.innerHTML){
                isOverlapping = true
                overlapCount++
                console.log('overlap')
            }
        })
        if(overlapCount<=0){
            isOverlapping = false
        }
    })
}

//ADD CLASS TO EACH SHIP TILE --- Horizontal
if(isClicked && !isVertical && isValid){
    if(!isOverlapping){
        for(let i = 0; i < shipLength; i++){
            player1SetupTiles[TileIndex + i].classList.add('ship')
        }
    }
}

//ADD CLASS TO EACH SHIP TILE --- Vertical
if(isClicked && isVertical && isValid){
    if(!isOverlapping){
        for(let i = 0; i < shipLength * 10; i = i + 10){
            player1SetupTiles[TileIndex + i].classList.add('ship')
        }
    }
}

    })
})

//EVENTLISTENER ON TILE MOUSEOVER Player 2
player2SetupTiles.forEach(tile =>{
    tile.addEventListener('mouseover', (TileIndex, LetterCoordIndex, NumberCoordIndex) => {
        //REMOVE CLASS FROM EACH TILE
        player2SetupTiles.forEach(tile => {
            tile.classList.remove('ship')
        })
        overlapCount = 0

        //GET TILE INDEX IN ARRAY
        for(let i = 0; i <= player2SetupTiles.length; i++){
            if(tile == player2SetupTiles[i]){
```

```
        TileIndex = i
        previewTiles = []
    }
}
//GET PREVIEW TILES --- Horizontal
if(!isVertical){
    for(let i=0; i<shipLength;i++){
        previewTiles.push(player2SetupTiles[TileIndex+i].innerHTML)
    }
}

//GET PREVIEW TILES --- Vertical
if(isVertical){
    for(let i=0; i<shipLength*10;i =i+10){
        previewTiles.push(player2SetupTiles[TileIndex+i].innerHTML)
    }
}

//GET LETTERCOORD in Array (index)
for(let i = 0; i <LetterCoordinates.length; i++){
    if(LetterCoordinates[i] == tile.innerHTML.slice(0,1)){
        LetterCoordIndex = i
    }
}

//GET NUMCOORD in ARRAY (index)
for(let i = 0; i <NumberCoordinates.length; i++){
    if(NumberCoordinates[i] == tile.innerHTML.slice(1,3)){
        NumberCoordIndex = i
    }
}

//CHECK IF VALID --- Horizontal
if(!isVertical){
    if(isClicked && LetterCoordIndex + shipLength <=11){
        isValid=true
    }else{
        isValid = false
    }
}

//CHECK IF VALID --- Vertical
if(isVertical){
    if(isClicked && NumberCoordIndex + shipLength <=11){
        isValid = true
    }else{
        isValid = false
    }
}

//CHECK OVERLAP -- Horizontal
if(!isVertical){
    player2ShipTiles.forEach(playerTile =>{
        previewTiles.forEach(previewTile =>{
            if(previewTile == playerTile.innerHTML){
```



```
        isOverlapping = true
        overlapCount++
        console.log('overlap')
    }
})
if(overlapCount<=0){
    isOverlapping = false
}
})
}

//CHECK OVERLAP -- Vertical
if(isVertical){
    player2ShipTiles.forEach(playerTile =>{
        previewTiles.forEach(previewTile =>{
            if(previewTile == playerTile.innerHTML){
                isOverlapping = true
                overlapCount++
                console.log('overlap')
            }
        })
        if(overlapCount<=0){
            isOverlapping = false
        }
    })
}

//ADD CLASS TO EACH SHIP TILE --- Horizontal
if(isClicked && !isVertical && isValid){
    if(!isOverlapping){
        for(let i = 0; i < shipLength; i++){
            player2SetupTiles[TileIndex + i].classList.add('ship')
        }
    }
}

//ADD CLASS TO EACH SHIP TILE --- Vertical
if(isClicked && isVertical && isValid){
    if(!isOverlapping){
        for(let i = 0; i < shipLength * 10; i = i + 10){
            player2SetupTiles[TileIndex + i].classList.add('ship')
        }
    }
}

})
})

//EVENTLISTENER ON TILE CLICK
player1SetupTiles.forEach(tile =>{
    tile.addEventListener('click', () =>{
        latestShipTiles = []
        if(isValid && !isOverlapping){
            playClick()
            latestShipBtn = shipName
        }
    })
})
})
```

```
document.querySelectorAll('.ship').forEach(shipTile =>{
  player1ShipTiles.push(shipTile)
  latestShipTiles.push(shipTile.innerHTML)
  shipTile.classList.add('shipClicked')
  switch (shipName) {
    case "Destroyer":
      for(let i = 0; i<shipLength; i++){
        document.querySelectorAll('.ship').forEach(shipTile =>{
          shipTile.classList.add('destroyer-1')
        })
      }
      break;
    case "Submarine":
      for(let i = 0; i<shipLength; i++){
        document.querySelectorAll('.ship').forEach(shipTile =>{
          shipTile.classList.add('submarine-1')
        })
      }
      break;
    case "Cruiser":
      for(let i = 0; i<shipLength; i++){
        document.querySelectorAll('.ship').forEach(shipTile =>{
          shipTile.classList.add('cruiser-1')
        })
      }
      break;
    case "Battleship":
      for(let i = 0; i<shipLength; i++){
        document.querySelectorAll('.ship').forEach(shipTile =>{
          shipTile.classList.add('battleship-1')
        })
      }
      break;
    case "Carrier":
      for(let i = 0; i<shipLength; i++){
        document.querySelectorAll('.ship').forEach(shipTile =>{
          shipTile.classList.add('carrier-1')
        })
      }
      break;
    default:
      break;
  }
  resetAfterPlaced()
})
placedCount++
if(placedCount >= 5){
  confirm1.classList.add('ready')
  confirm1.classList.remove('default-start-btn')
}
document.querySelector('.clicked').classList.add('placed')
document.querySelector('.clicked').classList.remove('clicked')
}

})
```

```
    })

    //EVENTLISTENER ON TILE CLICK
    player2SetupTiles.forEach(tile =>{
        tile.addEventListener('click', () =>{
            playClick()
            latestShipTiles = []
            if(isValid && !isOverlapping){
                latestShipBtn = shipName
                document.querySelectorAll('.ship').forEach(shipTile =>{
                    player2ShipTiles.push(shipTile)
                    latestShipTiles.push(shipTile.innerHTML)
                    shipTile.classList.add('shipClicked')
                    switch (shipName) {
                        case "Destroyer":
                            for(let i = 0; i<shipLength; i++){
                                document.querySelectorAll('.ship').forEach(shipTile =>{
                                    shipTile.classList.add('destroyer-2')
                                })
                            }
                            break;
                        case "Submarine":
                            for(let i = 0; i<shipLength; i++){
                                document.querySelectorAll('.ship').forEach(shipTile =>{
                                    shipTile.classList.add('submarine-2')
                                })
                            }
                            break;
                        case "Cruiser":
                            for(let i = 0; i<shipLength; i++){
                                document.querySelectorAll('.ship').forEach(shipTile =>{
                                    shipTile.classList.add('cruiser-2')
                                })
                            }
                            break;
                        case "Battleship":
                            for(let i = 0; i<shipLength; i++){
                                document.querySelectorAll('.ship').forEach(shipTile =>{
                                    shipTile.classList.add('battleship-2')
                                })
                            }
                            break;
                        case "Carrier":
                            for(let i = 0; i<shipLength; i++){
                                document.querySelectorAll('.ship').forEach(shipTile =>{
                                    shipTile.classList.add('carrier-2')
                                })
                            }
                            break;
                        default:
                            break;
                    }
                })
                resetAfterPlaced()
            })
            placedCount++
        })
    })
}
```

```
        if(placedCount >= 5){
            confirm2.classList.add('ready')
            confirm2.classList.remove('default-start-btn')
        }
        document.querySelector('.clicked').classList.add('placed')
        document.querySelector('.clicked').classList.remove('clicked')
    }

})

function resetAfterPlaced(){
    isClicked = false
    isValid = true
    isShipBtn = false
    isVertical = false
    shipLength = undefined
    shipName = undefined
    isOverlapping = false
}

//START GAME
function startGame(){
    leftTiles.forEach(leftTile =>{
        destroyerPlayer1.forEach(shipTile =>{
            if(shipTile.innerHTML == leftTile.innerHTML){
                leftTile.classList.add('player1Destroyer')
                leftTile.classList.add('ship1')
            }
        })
    })
    leftTiles.forEach(leftTile =>{
        submarinePlayer1.forEach(shipTile =>{
            if(shipTile.innerHTML == leftTile.innerHTML){
                leftTile.classList.add('ship1')
                leftTile.classList.add('player1Submarine')
            }
        })
    })
    leftTiles.forEach(leftTile =>{
        cruiserPlayer1.forEach(shipTile =>{
            if(shipTile.innerHTML == leftTile.innerHTML){
                leftTile.classList.add('ship1')
                leftTile.classList.add('player1Cruiser')
            }
        })
    })
    leftTiles.forEach(leftTile =>{
        battleshipPlayer1.forEach(shipTile =>{
            if(shipTile.innerHTML == leftTile.innerHTML){
                leftTile.classList.add('ship1')
                leftTile.classList.add('player1Battleship')
            }
        })
    })
})
```

```
leftTiles.forEach(leftTile =>{
    carrierPlayer1.forEach(shipTile =>{
        if(shipTile.innerHTML == leftTile.innerHTML){
            leftTile.classList.add('ship1')
            leftTile.classList.add('player1Carrier')
        }
    })
})
destroyerPlayer1 = document.querySelectorAll('.player1Destroyer')
submarinePlayer1 = document.querySelectorAll('.player1Submarine')
cruiserPlayer1 = document.querySelectorAll('.player1Cruiser')
battleshipPlayer1 = document.querySelectorAll('.player1Battleship')
carrierPlayer1 = document.querySelectorAll('.player1Carrier')
player1ShipTiles = document.querySelectorAll('.ship1')

rightTiles.forEach(leftTile =>{
    destroyerPlayer2.forEach(shipTile =>{
        if(shipTile.innerHTML == leftTile.innerHTML){
            leftTile.classList.add('player2Destroyer')
            leftTile.classList.add('ship2')
        }
    })
})
submarinePlayer2.forEach(shipTile =>{
    if(shipTile.innerHTML == leftTile.innerHTML){
        leftTile.classList.add('player2Submarine')
        leftTile.classList.add('ship2')
    }
})
cruiserPlayer2.forEach(shipTile =>{
    if(shipTile.innerHTML == leftTile.innerHTML){
        leftTile.classList.add('player2Cruiser')
        leftTile.classList.add('ship2')
    }
})
battleshipPlayer2.forEach(shipTile =>{
    if(shipTile.innerHTML == leftTile.innerHTML){
        leftTile.classList.add('player2Battleship')
        leftTile.classList.add('ship2')
    }
})
carrierPlayer2.forEach(shipTile =>{
    if(shipTile.innerHTML == leftTile.innerHTML){
        leftTile.classList.add('player2Carrier')
        leftTile.classList.add('ship2')
    }
})
})
```

```
destroyerPlayer2 = document.querySelectorAll('.player2Destroyer')
submarinePlayer2 = document.querySelectorAll('.player2Submarine')
cruiserPlayer2 = document.querySelectorAll('.player2Cruiser')
battleshipPlayer2 = document.querySelectorAll('.player2Battleship')
carrierPlayer2 = document.querySelectorAll('.player2Carrier')
player2ShipTiles = document.querySelectorAll('.ship2')

player1PrevClicked = ['blank']
player2PrevClicked = ['blank']
player1ShipSinkCount = 0
player2ShipSinkCount = 0
player1Turn = true
waiting = false
win = false
document.body.classList.add('player1')
}
```

```
//BATTLE
```

```
rightTiles.forEach(rightTile =>{
  rightTile.addEventListener('click', (e) =>{
    if(!waiting && player1Turn){
      isWaste = false
      waiting = true
      if(player1Turn){
        player1PrevClicked.forEach(player1PrevClickedTile =>{
          if(e.target.innerHTML == player1PrevClickedTile){
            //alert not valid
            isWaste = true
            waiting = false
          }
        })
      }
      if(player1Turn && !isWaste){
        player1Current = e.target.innerHTML
        player1PrevClicked.push(player1Current)
        e.target.classList.add('bomb')
        checkHit()
      }
    }
  })
})
```

```
leftTiles.forEach(leftTile =>{
  leftTile.addEventListener('click', (e) =>{
    if(!waiting && !player1Turn){
      isWaste = false
      waiting = true
      if(!player1Turn){
        player2PrevClicked.forEach(player2PrevClickedTile =>{
          if(e.target.innerHTML == player2PrevClickedTile){
            //alert not valid
            isWaste = true
            waiting = false
          }
        })
      }
    }
  })
})
```

```
        })
    }
    if(!player1Turn && !isWaste){
        player2Current = e.target.innerHTML
        player2PrevClicked.push(player2Current)
        e.target.classList.add('bomb')
        checkHit()
    }
}
})

function checkHit(){
    if(player1Turn){
        player2ShipTiles.forEach(player2ShipTile =>{
            if(player1Current == player2ShipTile.innerHTML){
                player2ShipTile.classList.add('tileSink')
                player2ShipTile.classList.remove('bomb')
                checkSink()
            }
        })
        setTimeout(() => {
            if(!win){
                switchTurn()
            }
        }, 1000);
    }
    if(!player1Turn){
        player1ShipTiles.forEach(player1ShipTile =>{
            if(player2Current == player1ShipTile.innerHTML){
                player1ShipTile.classList.add('tileSink')
                player1ShipTile.classList.remove('bomb')
                checkSink()
            }
        })
        setTimeout(() => {
            if(!win){
                switchTurn()
            }
        }, 1000);
    }
}

function checkSink(){
    if(player1Turn){
        sinkPlayer2Ship = false
        destroyerPlayer2Sink()
        submarinePlayer2Sink()
        cruiserPlayer2Sink()
        battleshipPlayer2Sink()
        carrierPlayer2Sink()
        if(sinkPlayer2Ship){
            playMusic()
            checkWin()
        }
    }
}
```

```
}
if(!player1Turn){
    sinkPlayer1Ship = false
    destroyerPlayer1Sink()
    submarinePlayer1Sink()
    cruiserPlayer1Sink()
    battleshipPlayer1Sink()
    carrierPlayer1Sink()
    if(sinkPlayer1Ship){
        playMusic()
        checkWin()
    }
}
}

function checkWin(){
    if(player1ShipSinkCount == 5){
        messageBox.innerHTML = 'Player-2 Wins'
        popupFunction()
    }
    if(player2ShipSinkCount == 5){
        messageBox.innerHTML = 'Player-1 Wins'
        popupFunction()
    }
}

function popupFunction(){
    middle.classList.add('background')
    middle.classList.remove('invisible')
    win = true
}

//Switch turns
function switchTurn(){
    if(document.body.className.includes('player1')){
        document.body.className = 'player2'
    }else{
        document.body.className = 'player1'
    }
    if(player1Turn){
        player1Turn = false
    }else{
        player1Turn = true
    }
    waitingBattle.classList.remove('invisible')
    battleContainer.classList.add('invisible')
    isWaste = false
    waiting = false
}

continueBattle.addEventListener('click', () =>{
    continueBattle.classList.add('hover')
    setTimeout(() => {
        continueBattle.classList.add('animate__animated','animate__bounceOutUp')
    }, 900);
    setTimeout(() => {
        battleContainer.classList.remove('invisible')
    }, 900);
})
```



```
        waitingBattle.classList.add('invisible')
        continueBattle.classList.remove('animate__animated', 'animate__bounceOutUp')
        continueBattle.classList.remove('hover')
    }, 1600);
})
```

```
function destroyerPlayer1Sink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    destroyerPlayer1.forEach(destroyerTile =>{
        if(destroyerTile.classList.contains('tileSink')){
            count++
        }
        if(destroyerTile.innerHTML == player2Current){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == destroyerPlayer1.length){
        sinkPlayer1Ship = true
        currentShipSink = true
    }
    if(currentShipSink){
        destroyerPlayer1.forEach(destroyerTile =>{
            destroyerTile.classList.add('shipSink')
            destroyerTile.classList.remove('tileSink')
        })
        player1ShipSinkCount++
    }
}

function submarinePlayer1Sink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    submarinePlayer1.forEach(submarineTile =>{
        if(submarineTile.classList.contains('tileSink')){
            count++
        }
        if(submarineTile.innerHTML == player2Current){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == submarinePlayer1.length){
        sinkPlayer1Ship = true
        currentShipSink = true
    }
    if(currentShipSink){
        submarinePlayer1.forEach(submarineTile =>{
            submarineTile.classList.add('shipSink')
            submarineTile.classList.remove('tileSink')
        })
    }
}
```

```
    })
    player1ShipSinkCount++
  }
}

function cruiserPlayer1Sink(belongsToShip, count, currentShipSink){
  count = 0
  belongsToShip = false
  currentShipSink = false
  cruiserPlayer1.forEach(cruiserTile =>{
    if(cruiserTile.classList.contains('tileSink')){
      count++
    }
    if(cruiserTile.innerHTML == player2Current){
      belongsToShip = true
    }
  })
  if(belongsToShip && count == cruiserPlayer1.length){
    sinkPlayer1Ship = true
    currentShipSink = true
  }
  if(currentShipSink){
    cruiserPlayer1.forEach(cruiserTile =>{
      cruiserTile.classList.add('shipSink')
      cruiserTile.classList.remove('tileSink')
    })
    player1ShipSinkCount++
  }
}

function battleshipPlayer1Sink(belongsToShip, count, currentShipSink){
  count = 0
  belongsToShip = false
  currentShipSink = false
  battleshipPlayer1.forEach(battleshipTile =>{
    if(battleshipTile.classList.contains('tileSink')){
      count++
    }
    if(battleshipTile.innerHTML == player2Current){
      belongsToShip = true
    }
  })
  if(belongsToShip && count == battleshipPlayer1.length){
    sinkPlayer1Ship = true
    currentShipSink = true
  }
  if(currentShipSink){
    battleshipPlayer1.forEach(battleshipTile =>{
      battleshipTile.classList.add('shipSink')
      battleshipTile.classList.remove('tileSink')
    })
    player1ShipSinkCount++
  }
}

function carrierPlayer1Sink(belongsToShip, count, currentShipSink){
  count = 0
  belongsToShip = false
```

```
currentShipSink = false
carrierPlayer1.forEach(carrierTile =>{
    if(carrierTile.classList.contains('tileSink')){
        count++
    }
    if(carrierTile.innerHTML == player2Current){
        belongsToShip = true
    }
})
if(belongsToShip && count == carrierPlayer1.length){
    sinkPlayer1Ship = true
    currentShipSink = true
}
if(currentShipSink){
    carrierPlayer1.forEach(carrierTile =>{
        carrierTile.classList.add('shipSink')
        carrierTile.classList.remove('tileSink')
    })
    player1ShipSinkCount++
}
}

function destroyerPlayer2Sink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    console.log(destroyerPlayer2)
    destroyerPlayer2.forEach(destroyerTile =>{
        if(destroyerTile.classList.contains('tileSink')){
            count++
        }
        if(destroyerTile.innerHTML == player1Current){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == destroyerPlayer2.length){
        sinkPlayer2Ship = true
        currentShipSink = true
    }
    if(currentShipSink){
        destroyerPlayer2.forEach(destroyerTile =>{
            destroyerTile.classList.add('shipSink')
            destroyerTile.classList.remove('tileSink')
        })
        player2ShipSinkCount++
    }
}

function submarinePlayer2Sink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    submarinePlayer2.forEach(submarineTile =>{
        if(submarineTile.classList.contains('tileSink')){
            count++
        }
    })
}
```

```
        if(submarineTile.innerHTML == player1Current){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == submarinePlayer2.length){
        sinkPlayer2Ship = true
        currentShipSink = true
    }
    if(currentShipSink){
        submarinePlayer2.forEach(submarineTile =>{
            submarineTile.classList.add('shipSink')
            submarineTile.classList.remove('tileSink')
        })
        player2ShipSinkCount++
    }
}

function cruiserPlayer2Sink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    cruiserPlayer2.forEach(cruiserTile =>{
        if(cruiserTile.classList.contains('tileSink')){
            count++
        }
        if(cruiserTile.innerHTML == player1Current){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == cruiserPlayer2.length){
        sinkPlayer2Ship = true
        currentShipSink = true
    }
    if(currentShipSink){
        cruiserPlayer2.forEach(cruiserTile =>{
            cruiserTile.classList.add('shipSink')
            cruiserTile.classList.remove('tileSink')
        })
        player2ShipSinkCount++
    }
}

function battleshipPlayer2Sink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    battleshipPlayer2.forEach(battleshipTile =>{
        if(battleshipTile.classList.contains('tileSink')){
            count++
        }
        if(battleshipTile.innerHTML == player1Current){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == battleshipPlayer2.length){
        sinkPlayer2Ship = true
        currentShipSink = true
    }
}
```

```

    }
    if(currentShipSink){
        battleshipPlayer2.forEach(battleshipTile =>{
            battleshipTile.classList.add('shipSink')
            battleshipTile.classList.remove('tileSink')
        })
        player2ShipSinkCount++
    }
}

function carrierPlayer2Sink(belongsToShip, count, currentShipSink){
    count = 0
    belongsToShip = false
    currentShipSink = false
    carrierPlayer2.forEach(carrierTile =>{
        if(carrierTile.classList.contains('tileSink')){
            count++
        }
        if(carrierTile.innerHTML == player1Current){
            belongsToShip = true
        }
    })
    if(belongsToShip && count == carrierPlayer2.length){
        sinkPlayer2Ship = true
        currentShipSink = true
    }
    if(currentShipSink){
        carrierPlayer2.forEach(carrierTile =>{
            carrierTile.classList.add('shipSink')
            carrierTile.classList.remove('tileSink')
        })
        player2ShipSinkCount++
    }
}

function playMusic(){
    var audio = new Audio('./stuff/audio.wav')
    audio.play()
}

function playClick(){
    var audio = new Audio('./stuff/click.wav')
    audio.play()
}

```

oppShipSetup.js

```

import {RightTiles, LetterCoordinates, NumberCoordinates} from "./main.js"
let OpponentShipTiles = []
let OpponentDestroyerTiles = []
let DestroyerPossibleMoves = []
let OpponentSubmarineTiles = []
let SubmarinePossibleMoves = []
let OpponentCruiserTiles = []
let CruiserPossibleMoves = []

```

```
let OpponentBattleshipTiles = []
let BattleshipPossibleMoves = []
let OpponentCarrierTiles = []
let CarrierPossibleMoves = []

//////////
///DESTROYER///
//////////
function destroyerGenerator(destroyerLetter, destroyerNumber, destroyerTile, nextMove){

    //GENERATE RANDOM STARTING TILE
    destroyerLetter = Math.floor(Math.random()*10+1)
    destroyerNumber = Math.floor(Math.random()*10+1)
    destroyerTile = LetterCoordinates[destroyerLetter] + NumberCoordinates[destroyerNumber]
    OpponentDestroyerTiles.push(destroyerTile)

    //check next possible tiles

    //NORTH
    if(destroyerNumber - 1 > 0){
        DestroyerPossibleMoves.push('north')
    }
    //EAST
    if(destroyerLetter + 1 <= 10){
        DestroyerPossibleMoves.push('east')
    }
    //SOUTH
    if(destroyerNumber + 1 <= 10){
        DestroyerPossibleMoves.push('south')
    }
    //WEST
    if(destroyerLetter - 1 > 0){
        DestroyerPossibleMoves.push('west')
    }

    //GENERATE NEXT MOVE DIRECTION
    nextMove = Math.floor(Math.random()*DestroyerPossibleMoves.length)

    //GO TO GENERATED DIRECTION
    switch (DestroyerPossibleMoves[nextMove]) {
        case 'north':
            //north
            OpponentDestroyerTiles.push(LetterCoordinates[destroyerLetter] +
            NumberCoordinates[destroyerNumber-1])
            break;
        case 'east':
            //east
            OpponentDestroyerTiles.push(LetterCoordinates[destroyerLetter+1] +
            NumberCoordinates[destroyerNumber])
            break;
        case 'south':
            //south
            OpponentDestroyerTiles.push(LetterCoordinates[destroyerLetter] +
            NumberCoordinates[destroyerNumber+1])
            break;
        case 'west':
```

```
        //west
        OpponentDestroyerTiles.push(LetterCoordinates[destroyerLetter-1] +
NumberCoordinates[destroyerNumber])
        default:
            break;
    }

//ADD CLASSLIST
RightTiles.forEach(e =>{
    for(let i=0; i < OpponentDestroyerTiles.length; i++){
        if(e.innerHTML == OpponentDestroyerTiles[i]){
            e.classList.add('destroyerOpp', 'shipOpp')
            OpponentShipTiles.push(e.innerHTML)
        }
    }
})
}

//////////
///SUBMARINE///
//////////
function submarineGenerator(submarineLetter, submarineNumber, submarineTile, nextMove){
    OpponentSubmarineTiles = []
    SubmarinePossibleMoves = []
    //GENERATE RANDOM STARTING TILE
    submarineLetter = Math.floor(Math.random()*10+1)
    submarineNumber = Math.floor(Math.random()*10+1)
    submarineTile = LetterCoordinates[submarineLetter] + NumberCoordinates[submarineNumber]
    OpponentSubmarineTiles.push(submarineTile)

    //check next possible tiles

    //NORTH
    if(submarineNumber - 2 > 0){
        SubmarinePossibleMoves.push('north')
    }
    //EAST
    if(submarineLetter + 2 <= 10){
        SubmarinePossibleMoves.push('east')
    }
    //SOUTH
    if(submarineNumber + 2 <= 10){
        SubmarinePossibleMoves.push('south')
    }
    //WEST
    if(submarineLetter - 2 > 0){
        SubmarinePossibleMoves.push('west')
    }

    //GENERATE NEXT MOVE DIRECTION
    nextMove = Math.floor(Math.random()*SubmarinePossibleMoves.length)

    //GO TO GENERATED DIRECTION
    switch (SubmarinePossibleMoves[nextMove]) {
        case 'north':
```

```

        //north
        for(let i=1; i<3; i++){
            OpponentSubmarineTiles.push(LetterCoordinates[submarineLetter] +
NumberCoordinates[submarineNumber-i])
        }
        break;
        case 'east':
            //east
            for(let i=1; i<3; i++){
                OpponentSubmarineTiles.push(LetterCoordinates[submarineLetter+i] +
NumberCoordinates[submarineNumber])
            }
            break;
        case 'south':
            //south
            for(let i=1; i<3; i++){
                OpponentSubmarineTiles.push(LetterCoordinates[submarineLetter] +
NumberCoordinates[submarineNumber+i])
            }
            break;
        case 'west':
            //west
            for(let i=1; i<3; i++){
                OpponentSubmarineTiles.push(LetterCoordinates[submarineLetter-i] +
NumberCoordinates[submarineNumber])
            }
            default:
                break;
    }

    //CHECK IF OVERLAP WITH DESTROYER
    OpponentDestroyerTiles.forEach(e =>{
        for( let i=0; i<OpponentSubmarineTiles.length; i++){
            if(e === OpponentSubmarineTiles[i]){
                submarineGenerator()
                return
            }
        }
    })
    //ADD CLASSLIST
    RightTiles.forEach(e =>{
        for(let i=0; i<OpponentSubmarineTiles.length; i++){
            if(e.innerHTML === OpponentSubmarineTiles[i]){
                e.classList.add('submarineOpp', 'shipOpp')
                OpponentShipTiles.push(e.innerHTML)
            }
        }
    })
}

//////////
///CRUISER///
//////////
function cruiserGenerator(cruiserLetter, cruiserNumber, cruiserTile, nextMove){
    OpponentCruiserTiles = []
    CruiserPossibleMoves = []

```



```
//GENERATE RANDOM STARTING TILE
cruiserLetter = Math.floor(Math.random()*10+1)
cruiserNumber = Math.floor(Math.random()*10+1)
cruiserTile = LetterCoordinates[cruiserLetter] + NumberCoordinates[cruiserNumber]
OpponentCruiserTiles.push(cruiserTile)

//check next possible tiles

//NORTH
if(cruiserNumber - 2 > 0){
    CruiserPossibleMoves.push('north')
}
//EAST
if(cruiserLetter + 2 <= 10){
    CruiserPossibleMoves.push('east')
}
//SOUTH
if(cruiserNumber + 2 <= 10){
    CruiserPossibleMoves.push('south')
}
//WEST
if(cruiserLetter - 2 > 0){
    CruiserPossibleMoves.push('west')
}

//GENERATE NEXT MOVE DIRECTION
nextMove = Math.floor(Math.random()*CruiserPossibleMoves.length)

//GO TO GENERATED DIRECTION
switch (CruiserPossibleMoves[nextMove]) {
    case 'north':
        //north
        for(let i=1; i<3; i++){
            OpponentCruiserTiles.push(LetterCoordinates[cruiserLetter] +
NumberCoordinates[cruiserNumber-i])
        }
        break;
    case 'east':
        //east
        for(let i=1; i<3; i++){
            OpponentCruiserTiles.push(LetterCoordinates[cruiserLetter+i] +
NumberCoordinates[cruiserNumber])
        }
        break;
    case 'south':
        //south
        for(let i=1; i<3; i++){
            OpponentCruiserTiles.push(LetterCoordinates[cruiserLetter] +
NumberCoordinates[cruiserNumber+i])
        }
        break;
    case 'west':
        //west
        for(let i=1; i<3; i++){
            OpponentCruiserTiles.push(LetterCoordinates[cruiserLetter-i] +
NumberCoordinates[cruiserNumber])
        }
        break;
}
```

```
        }
        default:
            break;
    }
    //CHECK IF OVERLAP WITH DESTROYER
    OpponentDestroyerTiles.forEach(e =>{
        for(let i=0; i<OpponentCruiserTiles.length; i++){
            if(e === OpponentCruiserTiles[i]){
                cruiserGenerator()
                return
            }
        }
    })
    //CHECK IF OVERLAP WITH SUBMARINE
    OpponentSubmarineTiles.forEach(e =>{
        for(let i=0; i<OpponentCruiserTiles.length; i++){
            if(e === OpponentCruiserTiles[i]){
                cruiserGenerator()
                return
            }
        }
    })
    //ADD CLASSLIST
    RightTiles.forEach(e =>{
        for(let i=0; i<OpponentCruiserTiles.length; i++){
            if(e.innerHTML === OpponentCruiserTiles[i]){
                e.classList.add('cruiserOpp', 'shipOpp')
                OpponentShipTiles.push(e.innerHTML)
            }
        }
    })
}

//////////
///BATTLESHIP///
//////////
function battleshipGenerator(battleshipLetter, battleshipNumber, battleshipTile, nextMove){
    OpponentBattleshipTiles = []
    BattleshipPossibleMoves = []
    //GENERATE RANDOM STARTING TILE
    battleshipLetter = Math.floor(Math.random()*10+1)
    battleshipNumber = Math.floor(Math.random()*10+1)
    battleshipTile = LetterCoordinates[battleshipLetter] + NumberCoordinates[battleshipNumber]
    OpponentBattleshipTiles.push(battleshipTile)

    //check next possible tiles

    //NORTH
    if(battleshipNumber - 3 > 0){
        BattleshipPossibleMoves.push('north')
    }
    //EAST
    if(battleshipLetter + 3 <= 10){
        BattleshipPossibleMoves.push('east')
    }
}
```

```
//SOUTH
if(battleshipNumber + 3 <= 10){
    BattleshipPossibleMoves.push('south')
}
//WEST
if(battleshipLetter - 3 > 0){
    BattleshipPossibleMoves.push('west')
}

//GENERATE NEXT MOVE DIRECTION
nextMove = Math.floor(Math.random()*BattleshipPossibleMoves.length)

//GO TO GENERATED DIRECTION
switch (BattleshipPossibleMoves[nextMove]) {
    case 'north':
        //north
        for(let i=1; i<4; i++){
            OpponentBattleshipTiles.push(LetterCoordinates[battleshipLetter] +
NumberCoordinates[battleshipNumber-i])
        }
        break;
    case 'east':
        //east
        for(let i=1; i<4; i++){
            OpponentBattleshipTiles.push(LetterCoordinates[battleshipLetter+i] +
NumberCoordinates[battleshipNumber])
        }
        break;
    case 'south':
        //south
        for(let i=1; i<4; i++){
            OpponentBattleshipTiles.push(LetterCoordinates[battleshipLetter] +
NumberCoordinates[battleshipNumber+i])
        }
        break;
    case 'west':
        //west
        for(let i=1; i<4; i++){
            OpponentBattleshipTiles.push(LetterCoordinates[battleshipLetter-i] +
NumberCoordinates[battleshipNumber])
        }
        default:
            break;
}

//CHECK IF OVERLAP WITH DESTROYER
OpponentDestroyerTiles.forEach(e =>{
    for(let i=0; i<OpponentBattleshipTiles.length; i++){
        if(e === OpponentBattleshipTiles[i]){
            battleshipGenerator()
            return
        }
    }
})
//CHECK IF OVERLAP WITH SUBMARINE
OpponentSubmarineTiles.forEach(e =>{
```

```
        for(let i=0; i<OpponentBattleshipTiles.length; i++){
            if(e === OpponentBattleshipTiles[i]){
                battleshipGenerator()
                return
            }
        }
    })
    //CHECK IF OVERLAP WITH CRUISER
    OpponentCruiserTiles.forEach(e =>{
        for(let i=0; i<OpponentBattleshipTiles.length; i++){
            if(e === OpponentBattleshipTiles[i]){
                battleshipGenerator()
                return
            }
        }
    })
    //ADD CLASSLIST
    RightTiles.forEach(e =>{
        for(let i=0; i<OpponentBattleshipTiles.length; i++){
            if(e.innerHTML === OpponentBattleshipTiles[i]){
                e.classList.add('battleshipOpp', 'shipOpp')
                OpponentShipTiles.push(e.innerHTML)
            }
        }
    })
}

//////////
///CARRIER///
//////////

function carrierGenerator(carrierLetter, carrierNumber, carrierTile, nextMove){
    OpponentCarrierTiles = []
    CarrierPossibleMoves = []
    //GENERATE RANDOM STARTING TILE
    carrierLetter = Math.floor(Math.random()*10+1)
    carrierNumber = Math.floor(Math.random()*10+1)
    carrierTile = LetterCoordinates[carrierLetter] + NumberCoordinates[carrierNumber]
    OpponentCarrierTiles.push(carrierTile)

    //check next possible tiles

    //NORTH
    if(carrierNumber - 4 > 0){
        CarrierPossibleMoves.push('north')
    }
    //EAST
    if(carrierLetter + 4 <= 10){
        CarrierPossibleMoves.push('east')
    }
    //SOUTH
    if(carrierNumber + 4 <= 10){
        CarrierPossibleMoves.push('south')
    }
    //WEST
    if(carrierLetter - 4 > 0){
```

```
        CarrierPossibleMoves.push('west')
    }

    //GENERATE NEXT MOVE DIRECTION
    nextMove = Math.floor(Math.random()*CarrierPossibleMoves.length)

    //GO TO GENERATED DIRECTION
    switch (CarrierPossibleMoves[nextMove]) {
        case 'north':
            //north
            for(let i=1; i<5; i++){
                OpponentCarrierTiles.push(LetterCoordinates[carrierLetter] +
NumberCoordinates[carrierNumber-i])
            }
            break;
        case 'east':
            //east
            for(let i=1; i<5; i++){
                OpponentCarrierTiles.push(LetterCoordinates[carrierLetter+i] +
NumberCoordinates[carrierNumber])
            }
            break;
        case 'south':
            //south
            for(let i=1; i<5; i++){
                OpponentCarrierTiles.push(LetterCoordinates[carrierLetter] +
NumberCoordinates[carrierNumber+i])
            }
            break;
        case 'west':
            //west
            for(let i=1; i<5; i++){
                OpponentCarrierTiles.push(LetterCoordinates[carrierLetter-i] +
NumberCoordinates[carrierNumber])
            }
            default:
                break;
    }

    //CHECK IF OVERLAP WITH DESTROYER
    OpponentDestroyerTiles.forEach(e =>{
        for(let i=0; i<OpponentCarrierTiles.length; i++){
            if(e === OpponentCarrierTiles[i]){
                carrierGenerator()
                return
            }
        }
    })
    //CHECK IF OVERLAP WITH SUBMARINE
    OpponentSubmarineTiles.forEach(e =>{
        for(let i=0; i<OpponentCarrierTiles.length; i++){
            if(e === OpponentCarrierTiles[i]){
                carrierGenerator()
                return
            }
        }
    })
}
```

```
    })
    //CHECK IF OVERLAP WITH CRUISER
    OpponentCruiserTiles.forEach(e =>{
        for(let i=0; i<OpponentCarrierTiles.length; i++){
            if(e === OpponentCarrierTiles[i]){
                carrierGenerator()
                return
            }
        }
    })
    //CHECK IF OVERLAP WITH BATTLESHIP
    OpponentBattleshipTiles.forEach(e =>{
        for(let i=0; i<OpponentCarrierTiles.length; i++){
            if(e === OpponentCarrierTiles[i]){
                carrierGenerator()
                return
            }
        }
    })
    //ADD CLASSLIST
    RightTiles.forEach(e =>{
        for(let i=0; i<OpponentCarrierTiles.length; i++){
            if(e.innerHTML === OpponentCarrierTiles[i]){
                e.classList.add('carrierOpp', 'shipOpp')
                OpponentShipTiles.push(e.innerHTML)
            }
        }
    })
}

//RESET OPPONENT BOARD
export function opponentReset(){
    RightTiles.forEach(e =>{
        e.classList.remove('shipOpp', 'hit', 'miss')
    })
    OpponentShipTiles = []
    OpponentDestroyerTiles = []
    DestroyerPossibleMoves = []
    OpponentSubmarineTiles = []
    SubmarinePossibleMoves = []
    OpponentCruiserTiles = []
    CruiserPossibleMoves = []
    OpponentBattleshipTiles = []
    BattleshipPossibleMoves = []
    OpponentCarrierTiles = []
    CarrierPossibleMoves = []
}
export function oppShipsGenerator(){

    destroyerGenerator()
    submarineGenerator()
    cruiserGenerator()
    battleshipGenerator()
    carrierGenerator()
}
```

welcome.js

```
let playBtn = document.querySelector('.play-btn')

playBtn.addEventListener('click', () =>{
  window.localStorage.clear()
})
```