Starting

Show the existing HTML and code. Show the HTML and IMGs
Diagram on the board what we will be creating:

1. grid
2. pirates
3. ship
4. treature
5. how to move
6. how to win

Creating the GRID

Explain the goal with the grid, pirates, starting position of the ship, and the treasure

1. Need the DOMContentLoaded event

```
document.addEventListener('DOMContentLoaded', () => {
```

2. Complete the CreateGrid Method

```
function createGrid() {
   const frame = document.getElementById('frame');

   for (let i = 0; i < 10 ; i++) {
       buildRow(frame);
   }
   resetGame();
}
```

3. Complete the buildRow() method

```
function buildRow(frame) {
   const row = document.createElement('div');
   row.classList.add('row');
   frame.insertAdjacentElement('beforeend', row);
   for (let i = 0; i < 10 ; i++) {
       buildSquare(row, i);
   }
}
```

4. Complete the buildSquare() method
   Talk about how we need to add the random pirates

```
function buildSquare(row, count) {
   const container = document.createElement('div');
   container.classList.add('square');
```

```
    if (count > 1 && count < 89) {
        if ((Math.floor(Math.random() * 100) + 1) > 85) {
            container.classList.add('pirate');
        }
    }
    row.insertAdjacentElement('beforeend', container);
}
```

5. Build resetGame()
   Talk about how the Ship and Treasure will work with the css classes
   a. Build getShipLocation()
   ```
   function getShipLocation() {
      return document.getElementById('frame').querySelector('.boat');
   }
   ```
   b. Build the rest of the method
   ```
   function resetGame() {

      // Inform the user they can start
      const announce = document.querySelector('.announce');
      announce.innerText = "Play!";

      // Set the starting location of the boat and treasure
      const frame = document.getElementById('frame');
      frame.firstElementChild.firstElementChild.classList.add('boat');

      // If the last position is a pirate, remove it before adding the treasure
      const lastPosition = frame.lastElementChild.lastElementChild;
       if (lastPosition.classList.contains('pirate')) {
           lastPosition.classList.remove('pirate');
       }
       frame.lastElementChild.lastElementChild.classList.add('treasure');

   }
   ```

Moving the Ship

Explain how we will move the ship by using the ship's current location and its relationships in the DOM. We will create methods for each movement. Build the method signatures for each movement method as we go.

1. Add An EventListener for the keypress, let's use the Arrow Keys
   ```
   document.addEventListener('DOMContentLoaded', () => {
      createGrid();
   ```

```
    document.querySelector('body').addEventListener('keyup', (event) => {
        if (event.key === 'ArrowRight') {
            moveShipRight();
        }
        if (event.key === 'ArrowLeft') {
            moveShipLeft();
        }
        if (event.key === 'ArrowDown') {
            moveShipDown();
        }
        if (event.key === 'ArrowUp') {
            moveShipUp();
        }
    });

});
```

2. Populate the methods to move the ship. Create Left and Right first, and then up and down. Talk about how we will build a moveShip() method so the movement is consistent once we know where we are moving. Talk about how we need to calculate the index it is at, so we can put it in the same place… let's build some methods to do that.

```
function moveShipRight() {
    const ship = getShipLocation();
    const right = ship.nextElementSibling;
    moveShip(ship, right);
}

function moveShipLeft() {
    const ship = getShipLocation();
    const left = ship.previousElementSibling;
    moveShip(ship, left);
}

function moveShipDown() {
    const ship = getShipLocation();
    const down = getUpperOrLowerElementAtIndex(ship,
ship.parentElement.nextElementSibling);
    moveShip(ship, down);
}

function moveShipUp() {
    const ship = getShipLocation();
```

```
    const up = getUpperOrLowerElementAtIndex(ship,
ship.parentElement.previousElementSibling);
    moveShip(ship, up);
}

function moveShip(shipElement, newElement) {
    shipElement.classList.remove('boat');
    newElement.classList.add('boat');
}


function getUpperOrLowerElementAtIndex(ship, newElement) {
    let elementAtIndex = null;
    if (newElement != null) {
        const index = getIndexOfElement(ship);
        elementAtIndex = newElement.childNodes[index];
    }
    return elementAtIndex;
}

function getIndexOfElement(element) {
    return Array.from(element.parentNode.children).indexOf(element);
}
```

3. Show the problem with moving off the screen and over pirates. How can we fix this?

```
function canMoveToElement(element) {
    if (element == null || element.classList.contains('pirate')) {
        return false;
    }
    return true;
}

UPDATE moveShip()
function moveShip(shipElement, newElement) {
    if (canMoveToElement(newElement)) {
        shipElement.classList.remove('boat');
        newElement.classList.add('boat');
    }
}
```

4. Add an Event for the Reset button

```
document.addEventListener('DOMContentLoaded', () => {
```

```
    createGrid();

    document.querySelector('body').addEventListener('keyup', (event) => {
        if (event.key === 'ArrowRight') {
            moveShipRight();
        }
        if (event.key === 'ArrowLeft') {
            moveShipLeft();
        }
        if (event.key === 'ArrowDown') {
            moveShipDown();
        }
        if (event.key === 'ArrowUp') {
            moveShipUp();
        }
    });

    document.getElementById('resetButton').addEventListener('click', (event) => {
        resetGame();
        event.preventDefault();
    });
});
```

5. Update the Reset to restart the Ship in the correct location

```
function resetGame() {
    const ship = getShipLocation();
    if (ship != null) {
        ship.classList.remove('boat');
    }

    const announce = document.querySelector('.announce');
    announce.innerText = "Play!";

    const frame = document.getElementById('frame');
    frame.firstElementChild.firstElementChild.classList.add('boat');
    frame.lastElementChild.lastElementChild.classList.add('treasure');
}
```

Winning the Game

We need a Win condition… what does it mean to win the game?

1. Create the isWin() function

```
function isWin(nextElement) {
   if (nextElement.classList.contains("treasure")) {
       return true;
   }
   return false;
}
```

2. Create the win() method to show the Win condition

```
function win() {
   const announce = document.querySelector('.announce');
   announce.classList.add('winText');
   announce.innerText = "You Win!";
   getShipLocation().classList.remove('boat');
}
```

3. Update moveShip to check for the win condition when the ship is moved

```
function moveShip(shipElement, newElement) {
   if (canMoveToElement(newElement)) {
       if (isWin(newElement)) {
           win();
       } else {
             shipElement.classList.remove('boat');
             newElement.classList.add('boat');
      }
   }
}
```

4. Update reset game to remove the winText class from the announce.

Challenge: Extend the Game

1. Create a loose condition - ship hits pirate
2. Add icebergs (show the image)
3. Make the pirates move randomly
4. Give the pirates a basic AI so they chase the ship
5. Make it so the ship can shoot a cannon at obstacles
6. Update the pirates so they can shoot a cannon at the ship
7. Update the grid to be dynamically sized
8. Create a scoring system
9. Create a timed game
10. Create different difficulties (easy, intermediate, advanced, torment)