CAVE-GAME

cave-game.netlify.com

by Martin Carriel

About Me

Aspiring software developer utilizing web development software, Python, and database management software (SQL and NoSQL) to produce actionable analysis with clarity and creativity. Communicates complex ideas to people thoughtfully and succinctly. Actively seeking a junior or entry level software developer role or web development position utilizing current / new technologies to continue to grow within technology.

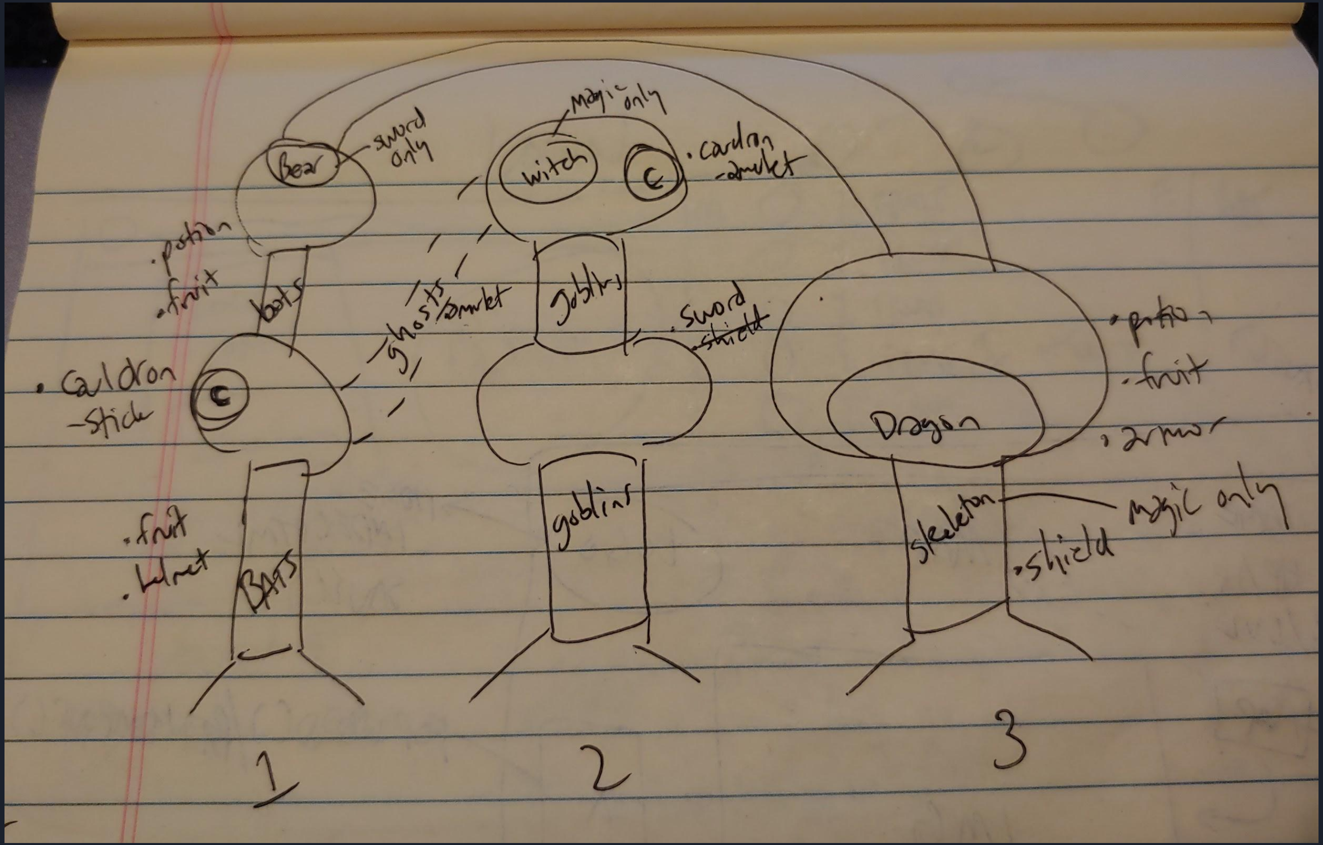
Cave-Game is a short RPG/puzzle game where you *explore caves*, *collect items*, and *slay the dragon*

Inspired by RPGs from my childhood (*Runescape*, *Pokémon*), I came up with the idea for a simple game that anyone could quickly play with.

I also wanted to challenge myself by combining the **Object-Oriented Programming** (OOP) of game development with the **functional programming** of a **state-driven Single Page Application** (SPA).

Early Wireframes/Mockups

https://github.com/ambientstl/cave-game/tree/master/wireframes



First sketch of the game map displaying the three caves and enemy and item placement

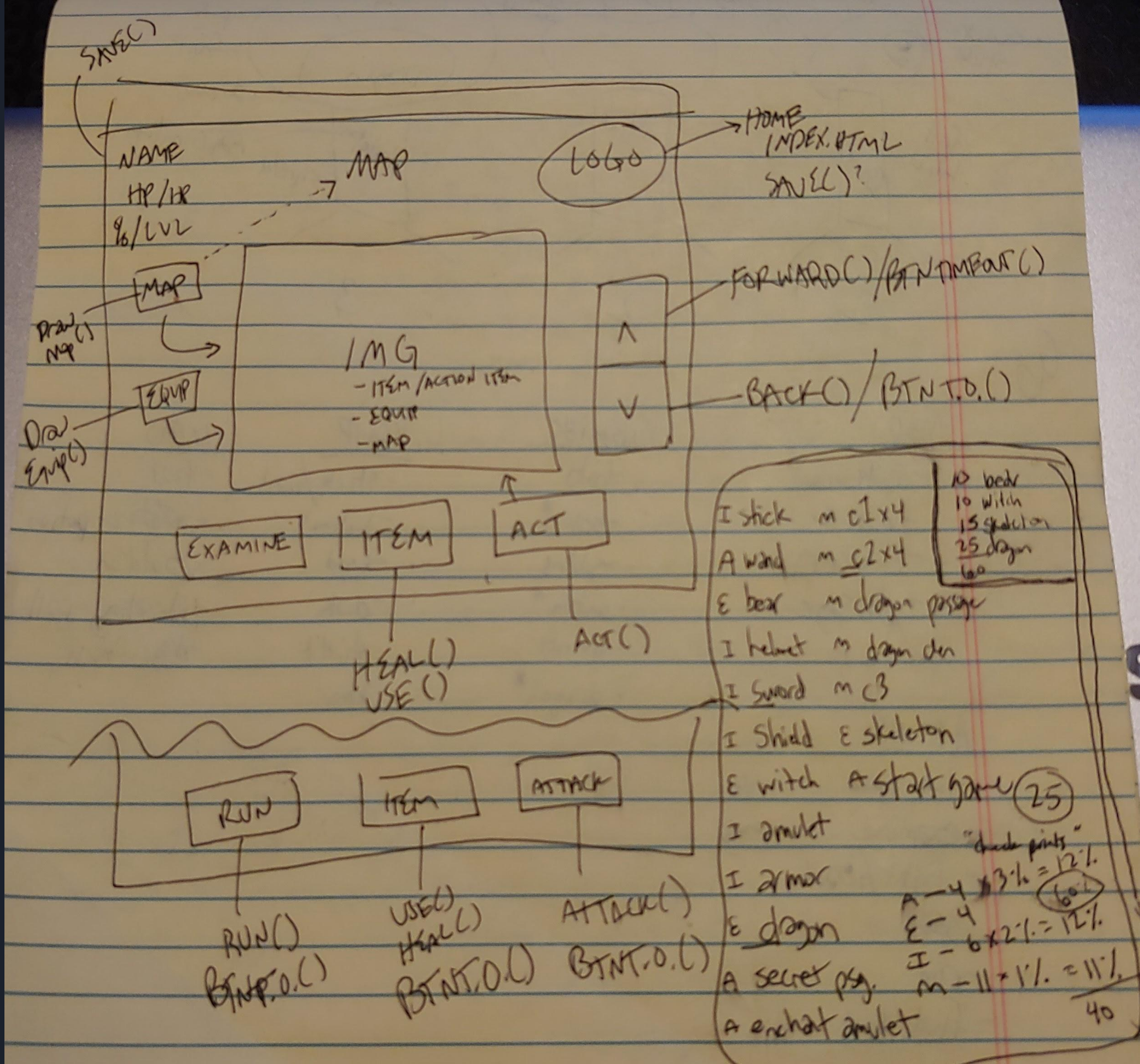
https://github.com/ambientstl/cave-game/blob/master/wireframes/1%20-%20game%20map.jpg

Top Left: Early wireframe displaying placement of buttons and information on the page and possible functions which the buttons will trigger.

Bottom Left: Buttons to display during a fight with an enemy

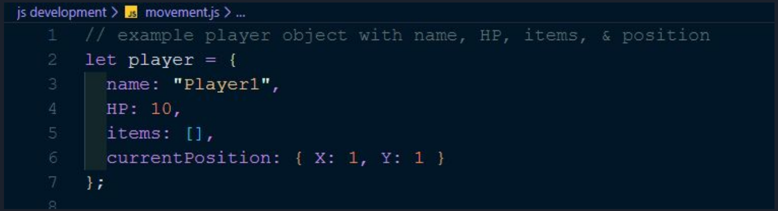
Bottom Right: List of very “achievement” in the game to calculate progress percentage

https://github.com/ambientstl/cave-game/blob/master/wireframes/4%20-%20main%20wireframe%20(left)%20%26%20list%20of%20game%20events%20(bottom%20right).jpg

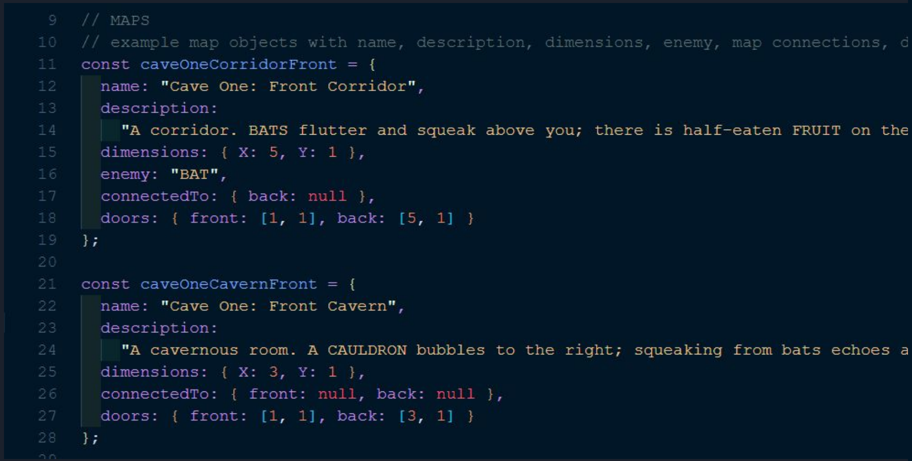


First Efforts at Functionality

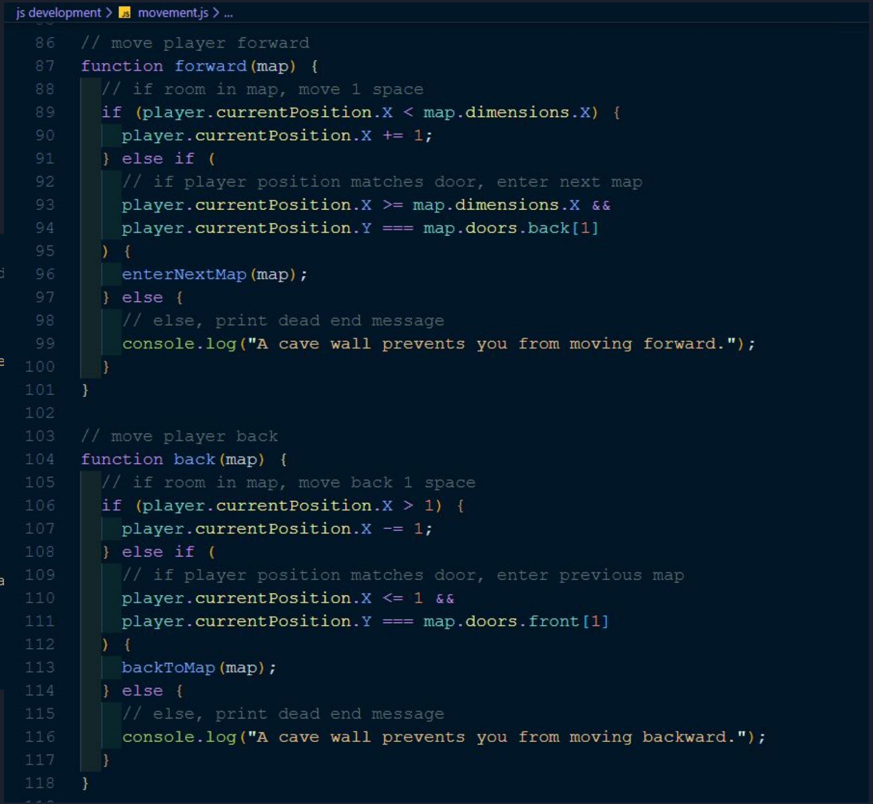
https://github.com/ambientstl/cave-game/tree/master/js%20development


js development > movement.js > ...

```
// example player object with name, HP, items, & position
let player = {
  name: "Player1",
  HP: 10,
  items: [],
  currentPosition: { X: 1, Y: 1 }
};
```



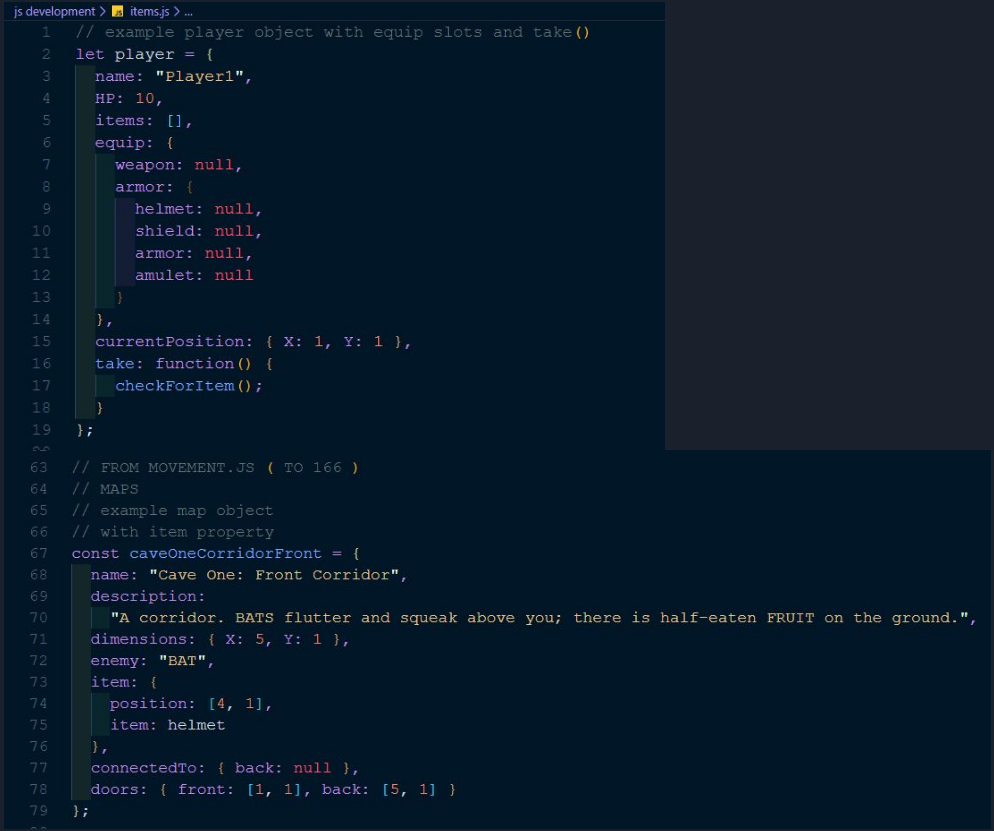
```
// MAPS
// example map objects with name, description, dimensions, enemy, map connections, d
const caveOneCorridorFront = {
  name: "Cave One: Front Corridor",
  description:
    "A corridor. BATS flutter and squeak above you; there is half-eaten FRUIT on the
  dimensions: { X: 5, Y: 1 },
  enemy: "BAT",
  connectedTo: { back: null },
  doors: { front: [1, 1], back: [5, 1] }
};

const caveOneCavernFront = {
  name: "Cave One: Front Cavern",
  description:
    "A cavernous room. A CAULDRON bubbles to the right; squeaking from bats echoes a
  dimensions: { X: 3, Y: 1 },
  connectedTo: { front: null, back: null },
  doors: { front: [1, 1], back: [3, 1] }
};
```


js development > movement.js > ...

```
// move player forward
function forward(map) {
  // if room in map, move 1 space
  if (player.currentPosition.X < map.dimensions.X) {
    player.currentPosition.X += 1;
  } else if (
    // if player position matches door, enter next map
    player.currentPosition.X >= map.dimensions.X &&
    player.currentPosition.Y === map.doors.back[1]
  ) {
    enterNextMap(map);
  } else {
    // else, print dead end message
    console.log("A cave wall prevents you from moving forward.");
  }
}

// move player back
function back(map) {
  // if room in map, move back 1 space
  if (player.currentPosition.X > 1) {
    player.currentPosition.X -= 1;
  } else if (
    // if player position matches door, enter previous map
    player.currentPosition.X <= 1 &&
    player.currentPosition.Y === map.doors.front[1]
  ) {
    backToMap(map);
  } else {
    // else, print dead end message
    console.log("A cave wall prevents you from moving backward.");
  }
}
```

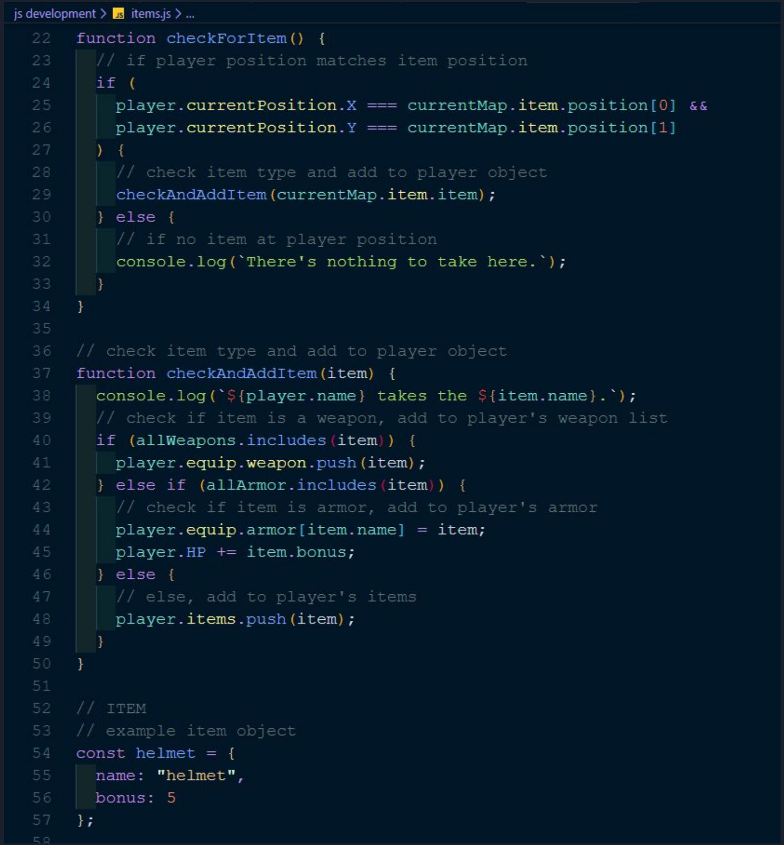
Early attempts at ***movement*** *functionality: Player object, Map objects, and forward/back functions*

https://github.com/ambientstl/cave-game/blob/master/js%20development/movement.js


js development > items.js > ...

```javascript
// example player object with equip slots and take()
let player = {
  name: "Player1",
  HP: 10,
  items: [],
  equip: {
    weapon: null,
    armor: {
      helmet: null,
      shield: null,
      armor: null,
      amulet: null
    }
  },
  currentPosition: { X: 1, Y: 1 },
  take: function() {
    checkForItem();
  }
};
```

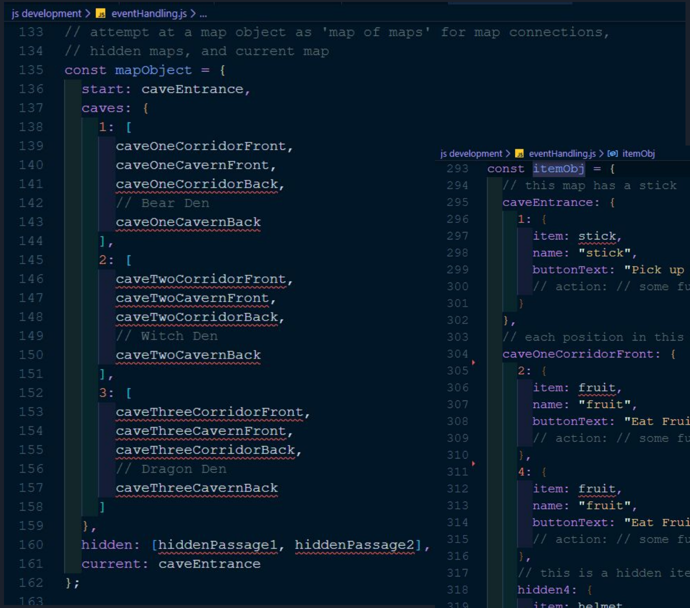
```javascript
// FROM MOVEMENT.JS ( TO 166 )
// MAPS
// example map object
// with item property
const caveOneCorridorFront = {
  name: "Cave One: Front Corridor",
  description:
    "A corridor. BATS flutter and squeak above you; there is half-eaten FRUIT on the ground.",
  dimensions: { X: 5, Y: 1 },
  enemy: "BAT",
  item: {
    position: [4, 1],
    item: helmet
  },
  connectedTo: { back: null },
  doors: { front: [1, 1], back: [5, 1] }
};
```


js development > items.js > ...

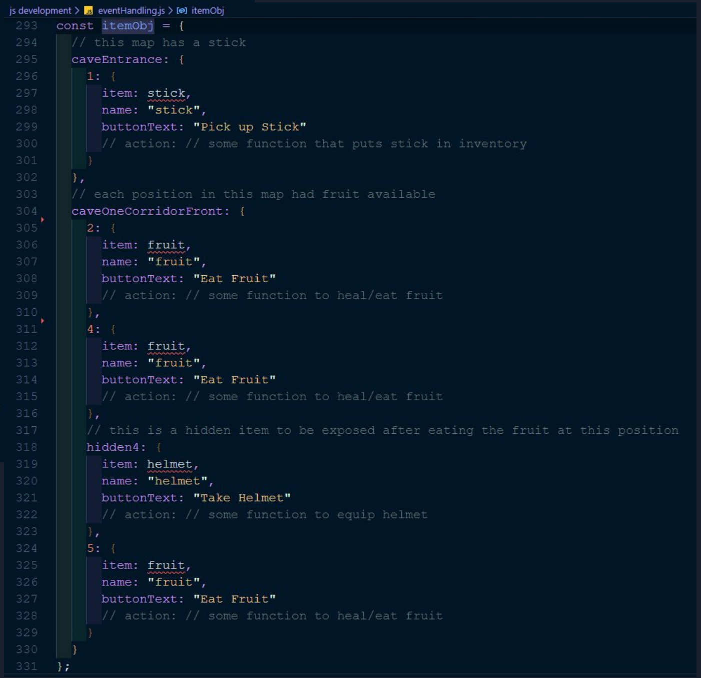
```javascript
function checkForItem() {
  // if player position matches item position
  if (
    player.currentPosition.X === currentMap.item.position[0] &&
    player.currentPosition.Y === currentMap.item.position[1]
  ) {
    // check item type and add to player object
    checkAndAddItem(currentMap.item.item);
  } else {
    // if no item at player position
    console.log(`There's nothing to take here.`);
  }
}

// check item type and add to player object
function checkAndAddItem(item) {
  console.log(`${player.name} takes the ${item.name}.`);
  // check if item is a weapon, add to player's weapon list
  if (allWeapons.includes(item)) {
    player.equip.weapon.push(item);
  } else if (allArmor.includes(item)) {
    // check if item is armor, add to player's armor
    player.equip.armor[item.name] = item;
    player.HP += item.bonus;
  } else {
    // else, add to player's items
    player.items.push(item);
  }
}

// ITEM
// example item object
const helmet = {
  name: "helmet",
  bonus: 5
};
```

*Early attempts at **item** functionality: Player object, Map object, and “equip” functionality*

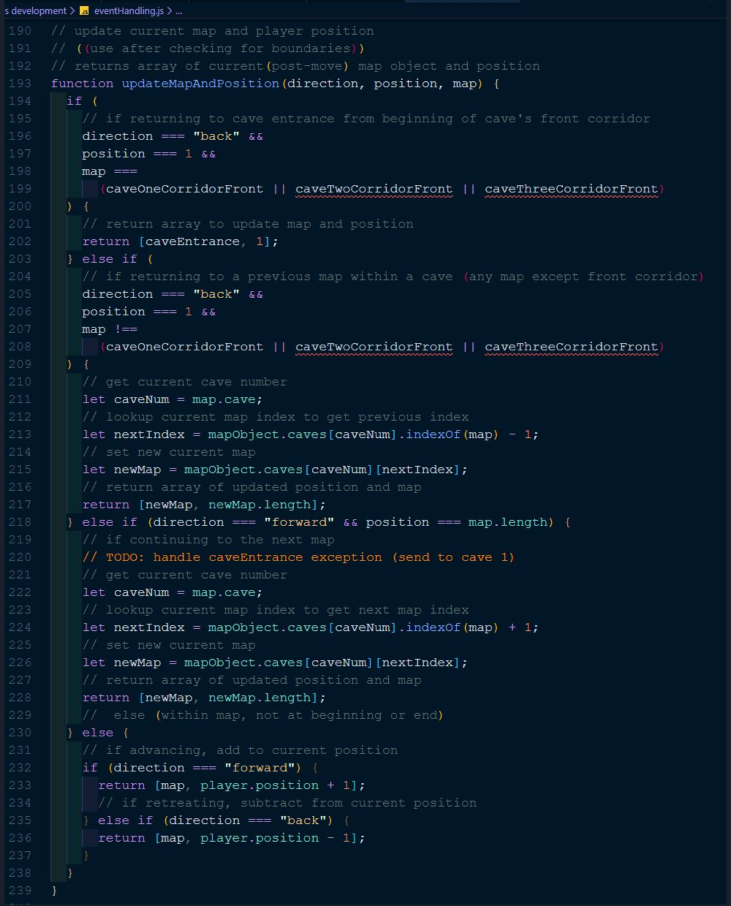
https://github.com/ambientstl/cave-game/blob/master/js%20development/items.js



```javascript
// attempt at a map object as 'map of maps' for map connections,
// hidden maps, and current map
const mapObject = {
  start: caveEntrance,
  caves: {
    1: [
      caveOneCorridorFront,
      caveOneCavernFront,
      caveOneCorridorBack,
      // Bear Den
      caveOneCavernBack
    ],
    2: [
      caveTwoCorridorFront,
      caveTwoCavernFront,
      caveTwoCorridorBack,
      // Witch Den
      caveTwoCavernBack
    ],
    3: [
      caveThreeCorridorFront,
      caveThreeCavernFront,
      caveThreeCorridorBack,
      // Dragon Den
      caveThreeCavernBack
    ]
  },
  hidden: [hiddenPassage1, hiddenPassage2],
  current: caveEntrance
};
```



```javascript
const itemObj = {
  // this map has a stick
  caveEntrance: {
    1: {
      item: stick,
      name: "stick",
      buttonText: "Pick up Stick"
      // action: // some function that puts stick in inventory
    }
  },
  // each position in this map had fruit available
  caveOneCorridorFront: {
    2: {
      item: fruit,
      name: "fruit",
      buttonText: "Eat Fruit"
      // action: // some function to heal/eat fruit
    },
    4: {
      item: fruit,
      name: "fruit",
      buttonText: "Eat Fruit"
      // action: // some function to heal/eat fruit
    },
    // this is a hidden item to be exposed after eating the fruit at this position
    hidden4: {
      item: helmet,
      name: "helmet",
      buttonText: "Take Helmet"
      // action: // some function to equip helmet
    },
    5: {
      item: fruit,
      name: "fruit",
      buttonText: "Eat Fruit"
      // action: // some function to heal/eat fruit
    }
  }
};
```



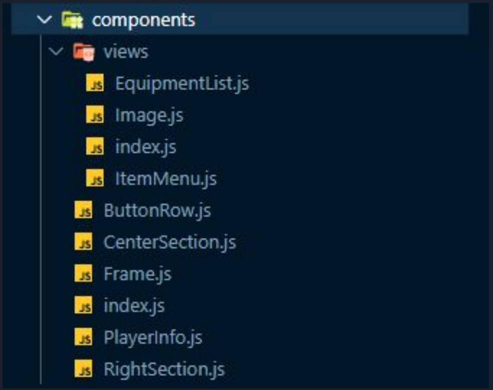
```javascript
// update current map and player position
// ((use after checking for boundaries))
// returns array of current(post-move) map object and position
function updateMapAndPosition(direction, position, map) {
  if (
    // if returning to cave entrance from beginning of cave's front corridor
    direction === "back" &&
    position === 1 &&
    map ===
      (caveOneCorridorFront || caveTwoCorridorFront || caveThreeCorridorFront)
  ) {
    // return array to update map and position
    return [caveEntrance, 1];
  } else if (
    // if returning to a previous map within a cave (any map except front corridor)
    direction === "back" &&
    position === 1 &&
    map !==
      (caveOneCorridorFront || caveTwoCorridorFront || caveThreeCorridorFront)
  ) {
    // get current cave number
    let caveNum = map.cave;
    // lookup current map index to get previous index
    let nextIndex = mapObject.caves[caveNum].indexOf(map) - 1;
    // set new current map
    let newMap = mapObject.caves[caveNum][nextIndex];
    // return array of updated position and map
    return [newMap, newMap.length];
  } else if (direction === "forward" && position === map.length) {
    // if continuing to the next map
    // TODO: handle caveEntrance exception (send to cave 1)
    // get current cave number
    let caveNum = map.cave;
    // lookup current map index to get next map index
    let nextIndex = mapObject.caves[caveNum].indexOf(map) + 1;
    // set new current map
    let newMap = mapObject.caves[caveNum][nextIndex];
    // return array of updated position and map
    return [newMap, newMap.length];
    //  else (within map, not at beginning or end)
  } else {
    // if advancing, add to current position
    if (direction === "forward") {
      return [map, player.position + 1];
      // if retreating, subtract from current position
    } else if (direction === "back") {
      return [map, player.position - 1];
    }
  }
}
```

Early attempts at combining functionality: "Maps" of Map and Item objects, unwieldy updatePosition function

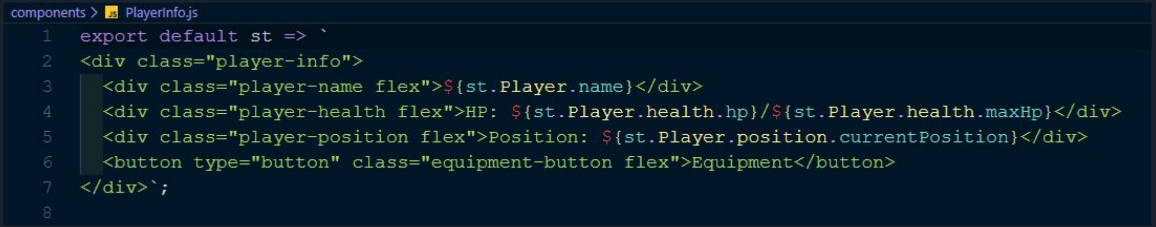
https://github.com/ambientstl/cave-game/blob/master/js%20development/eventHandling.js

SPA Architecture

SPA: Functional Components & Views

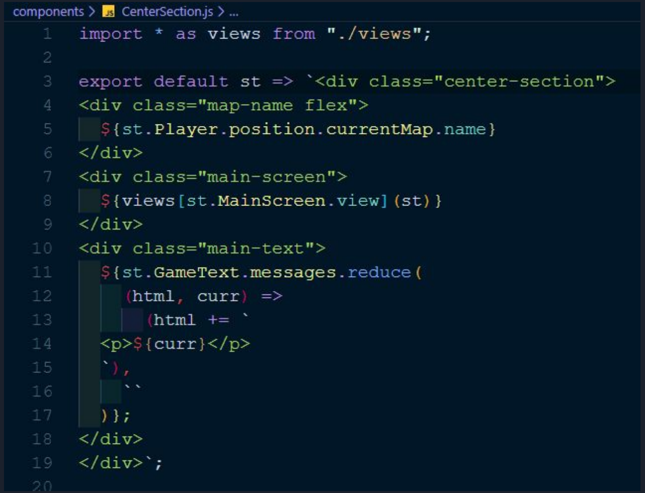


```
components
  views
    EquipmentList.js
    Image.js
    index.js
    ItemMenu.js
  ButtonRow.js
  CenterSection.js
  Frame.js
  index.js
  PlayerInfo.js
  RightSection.js
```



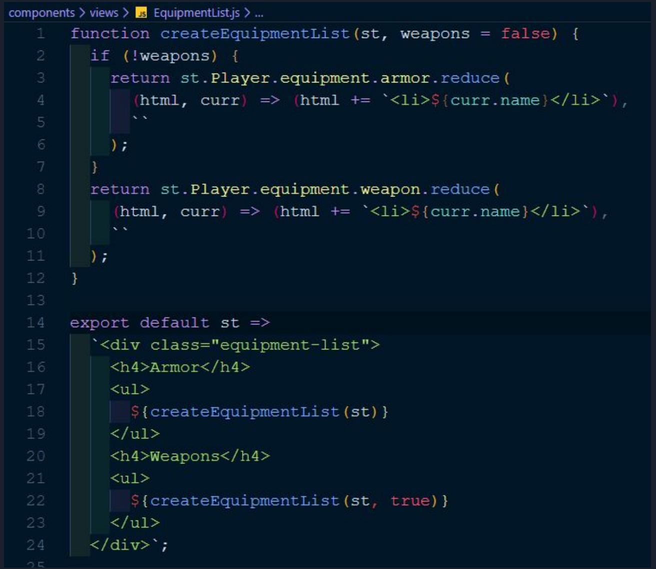
components > PlayerInfo.js

```
export default st => `
<div class="player-info">
  <div class="player-name flex">${st.Player.name}</div>
  <div class="player-health flex">HP: ${st.Player.health.hp}/${st.Player.health.maxHp}</div>
  <div class="player-position flex">Position: ${st.Player.position.currentPosition}</div>
  <button type="button" class="equipment-button flex">Equipment</button>
</div>`;
```



components > CenterSection.js > ...

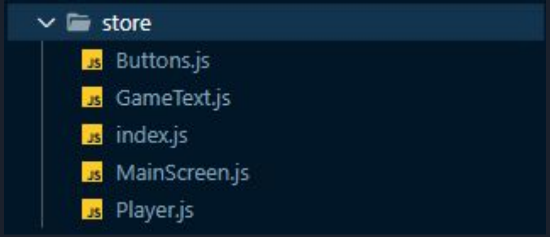
```
import * as views from "./views";

export default st => `<div class="center-section">
<div class="map-name flex">
  ${st.Player.position.currentMap.name}
</div>
<div class="main-screen">
  ${views[st.MainScreen.view](st)}
</div>
<div class="main-text">
  ${st.GameText.messages.reduce(
    (html, curr) =>
      (html += `
  <p>${curr}</p>
  `),
    ``
  )};
</div>
</div>`;
```



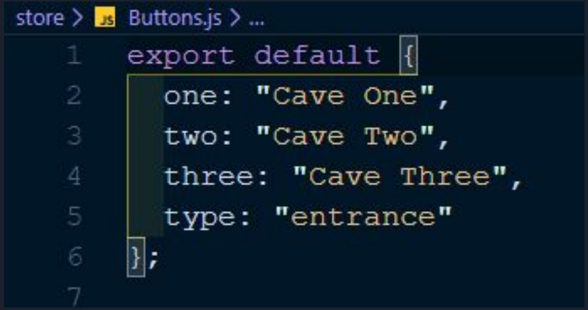
components > views > EquipmentList.js > ...

```
function createEquipmentList(st, weapons = false) {
  if (!weapons) {
    return st.Player.equipment.armor.reduce(
      (html, curr) => (html += `<li>${curr.name}</li>`),
      ``
    );
  }
  return st.Player.equipment.weapon.reduce(
    (html, curr) => (html += `<li>${curr.name}</li>`),
    ``
  );
}

export default st =>
  `<div class="equipment-list">
    <h4>Armor</h4>
    <ul>
      ${createEquipmentList(st)}
    </ul>
    <h4>Weapons</h4>
    <ul>
      ${createEquipmentList(st, true)}
    </ul>
  </div>`;
```

SPA: State/Store

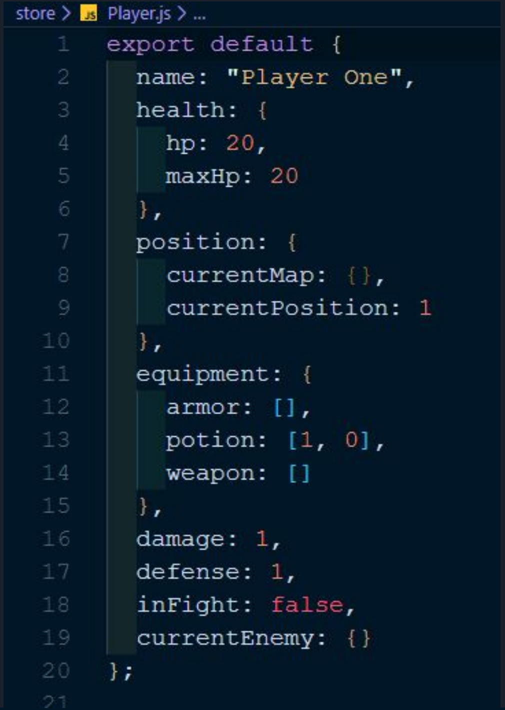


```
store
  Buttons.js
  GameText.js
  index.js
  MainScreen.js
  Player.js
```



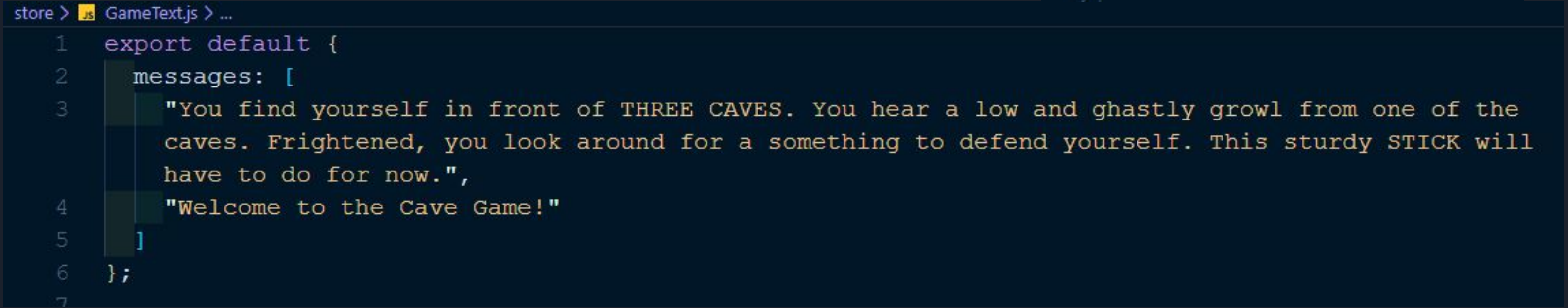
store > Buttons.js > ...

```js
export default {
  one: "Cave One",
  two: "Cave Two",
  three: "Cave Three",
  type: "entrance"
};
```



store > Player.js > ...

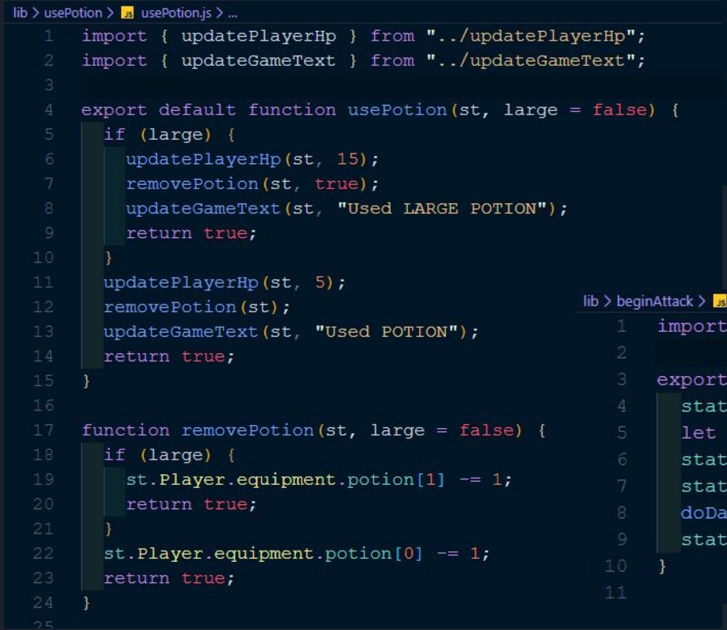
```js
export default {
  name: "Player One",
  health: {
    hp: 20,
    maxHp: 20
  },
  position: {
    currentMap: {},
    currentPosition: 1
  },
  equipment: {
    armor: [],
    potion: [1, 0],
    weapon: []
  },
  damage: 1,
  defense: 1,
  inFight: false,
  currentEnemy: {}
};
```



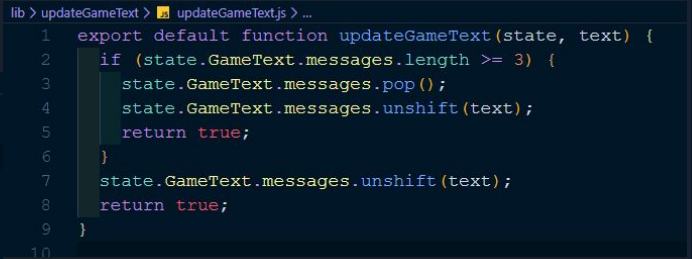
store > GameText.js > ...

```js
export default {
  messages: [
    "You find yourself in front of THREE CAVES. You hear a low and ghastly growl from one of the caves. Frightened, you look around for a something to defend yourself. This sturdy STICK will have to do for now.",
    "Welcome to the Cave Game!"
  ]
};
```

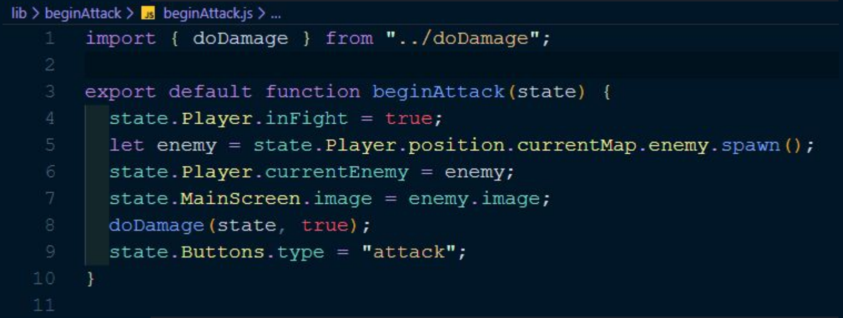
SPA: Module Library


lib > usePotion > usePotion.js > ...

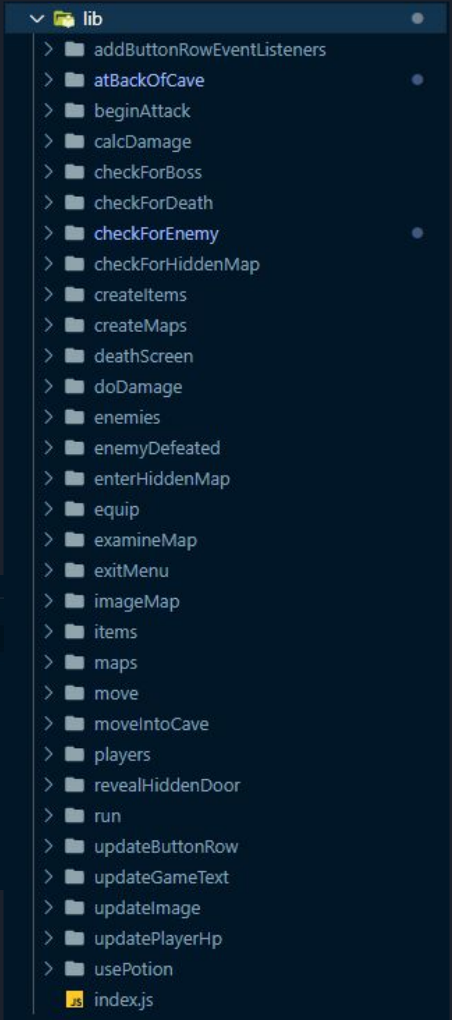
```js
import { updatePlayerHp } from "../updatePlayerHp";
import { updateGameText } from "../updateGameText";

export default function usePotion(st, large = false) {
  if (large) {
    updatePlayerHp(st, 15);
    removePotion(st, true);
    updateGameText(st, "Used LARGE POTION");
    return true;
  }
  updatePlayerHp(st, 5);
  removePotion(st);
  updateGameText(st, "Used POTION");
  return true;
}

function removePotion(st, large = false) {
  if (large) {
    st.Player.equipment.potion[1] -= 1;
    return true;
  }
  st.Player.equipment.potion[0] -= 1;
  return true;
}
```


lib > updateGameText > updateGameText.js > ...

```js
export default function updateGameText(state, text) {
  if (state.GameText.messages.length >= 3) {
    state.GameText.messages.pop();
    state.GameText.messages.unshift(text);
    return true;
  }
  state.GameText.messages.unshift(text);
  return true;
}
```


lib > beginAttack > beginAttack.js > ...

```js
import { doDamage } from "../doDamage";

export default function beginAttack(state) {
  state.Player.inFight = true;
  let enemy = state.Player.position.currentMap.enemy.spawn();
  state.Player.currentEnemy = enemy;
  state.MainScreen.image = enemy.image;
  doDamage(state, true);
  state.Buttons.type = "attack";
}
```



```
lib
  addButtonRowEventListeners
  atBackOfCave
  beginAttack
  calcDamage
  checkForBoss
  checkForDeath
  checkForEnemy
  checkForHiddenMap
  createItems
  createMaps
  deathScreen
  doDamage
  enemies
  enemyDefeated
  enterHiddenMap
  equip
  examineMap
  exitMenu
  imageMap
  items
  maps
  move
  moveIntoCave
  players
  revealHiddenDoor
  run
  updateButtonRow
  updateGameText
  updateImage
  updatePlayerHp
  usePotion
  index.js
```

Future Work

- Log-in page & functionality
  - Save progress
  - Keep track of previous attempts/best scores
- Track completion percentage
- Add button stylings & button timeout
- Music / Soundtrack