|  |  |
| --- | --- |
| Error/ challenge | Final code |
| When the user sends messages, the message should show up on the right side of the screen | function sendMessage(a) {      let divSelector = document.getElementById("chat");      let addDiv = document.createElement('div');      addDiv.className = "message parker";      divSelector.appendChild(addDiv);      addDiv.innerText = a;  } |
| Upon sending a message, the user is to receive a reply. This function makes the reply appear on the left side of the screen | function sendReply(a) {      delay = 0;      let divSelector = document.getElementById("chat");      optionTimer = responseList[a].length;      for (elem in responseList[a]) {          let addDiv = document.createElement("div");          addDiv.className = "message stark";          delay = 1000 \* (Number(elem)+1);          setTimeout(function(){              divSelector.appendChild(addDiv);          }, delay)          addDiv.innerText = (responseList[a])[elem];      }  } |
| When the user selects an option, their message is predetermined. This function checks if the message is sent. If not, it will replace the users keypress with a character from the predetermined message. If the message is complete, it will process sendMessage, as well as sendReply | function keyPress() {      if (messageSent) {          return;      } else {          if (tempWord.length == max) {              tempWord = "";              inputText.value = "";              messageSent = true;              indexNum=0;              sendMessage(storedA);              sendReply(storedA);              setTimeout(function() {                  showButton(storedA);              }, (1000 \* (optionTimer + 1)))              inputText.removeEventListener("keydown", keyPress);              return;          } else {              if (tempWord.length < max) {              tempWord += storedA.charAt(indexNum);              indexNum += 1;              inputText.value = tempWord;          }}      }  } |

|  |  |
| --- | --- |
| When the user presses the send button, this function processes send message and sendReply | function pressSend() {      if (messageSent) {          return;      } else {          sendMessage(storedA);          sendReply(storedA);          messageSent = true;          inputText.value = "";          setTimeout(function() {              showButton(storedA);          }, (1000 \* (optionTimer + 1)))          sendButton.removeEventListener("click",pressSend);      }  } |
| After the user presses a button, the options will temporarily disappear while the story from the dialogue plays out. This function helps the buttons disappear | function hideButtons() {      option11.innerText = "";      option12.innerText = "";  } |
| After the dialogue plays out, the user will be presented with a fresh set of options. This function helps determine the buttons from the button changes array set at the beginning of the code | function showButton(a) {      option11.innerText = (buttonChanges[a])[0];      option12.innerText = (buttonChanges[a])[1];  } |
| This function is processed when the user clicks one of the dialogue options. It temporarily creates event listeners for the send button and keypresses. It also hidesbuttons. It essentially combines all the helper functions from above | function pressOption(a) {      max = a.length;      storedA = a;      tempWord = "";      indexNum = 0;      hideButtons();      messageSent=false;      document.getElementById("lastSeen").innerText = "Currently active"      sendButton.addEventListener("click", pressSend);      inputText.addEventListener("keydown", keyPress);  } |
| This function is processed when the user clicks on the different contacts. It opens up the selected contact | function openContact(contactName) {      for (let elem in contactConversation) {          contactConversation[elem].style.display = "none";      }      contactName.style.display = "flex";  } |

|  |  |
| --- | --- |
| There is a separate section of the game where the user speaks to a “digital assistant”, “iris”. This function checks if the response is correct and processes the helper function, responsPicker | function tempText() {      let tov = textOption.value      if (tov != "") {          if (event.key == "Enter" || event.button == 0) {              textOption.removeEventListener("keydown", tempText)              textSendButton.removeEventListener("click", tempText)              responsePicker(tov)          } else {              return          }      }  } |

|  |  |
| --- | --- |
| This is the helper function for the response picker. It determines the response, depending on whether the user has provided the correct password and decryption code using helper functions | function responsePicker(tov) {      if (tov in responseList == false) {          if (password == false) {              sendMessage(tov);              sendReply("wrongPassword");          } else {              sendMessage(tov);              sendReply("wrongDecryption");          }      } else {          if (password == false) {              if (tov == "110420") {                  sendMessage(tov);                  sendReply("110420");                  password = true;              } else {                  sendMessage(tov);                  sendReply("wrongPassword");              }          } else {              if (password == true) {                  if (tov == "94") {                      sendMessage(tov);                      sendReply("94");                      decryption = true;                      textOption.disabled = true;                  } else {                      sendMessage(tov);                      sendReply("wrongDecryption");                  }              }          }      }      textOption.value = "";      setTimeout(function() {          textOption.addEventListener("keydown", tempText)      }, 1000 \* (optionTimer))      setTimeout(function() {          textSendButton.addEventListener("click", tempText)      }, 1000 \* (optionTimer))  } |

|  |  |
| --- | --- |
| This is a helper function which checks if the user has provided the correct password, and provides the appropriate options | function checkPassword() {      if (password == true) {          clearInterval(passwordStatus)          option11.innerText = btoa("Yo, watcha doin?")          option12.innerText = btoa("how's it going?")          eventListenerCompendium()          decryptionStatus = setInterval(checkDecryption, 1000)      } else {          return      }  } |
| This is a helper function which checks if the user has provided the correct decryption code, and provides the appropriate options | function checkDecryption() {      if (decryption == true) {          clearInterval(decryptionStatus)          option11.innerText = "Hey, actually, I've been meaning to talk to you"          option12.innerText = "Can I be real for a while?"      }  } |

|  |  |
| --- | --- |
| This is a function which has most of the eventListeners required for the page.  This was implemented because the event listeners change depending on the state of the game. After the password is provided, the event listeners are active. This is done to prevent any events from occurring before the correct password is provided. It also alter the behaviour if the password is provided, and the decryption code is provided. | function eventListenerCompendium() {      // event listner to listen for when the user clicks the option. this advances story      option11.addEventListener("click", function() {          if (option11.innerText == false) {              return;          } else {              console.log(option11.innerText)              if (option11.innerText == "that sounds great!" || option11.innerText == "honesty is one thing. Pessimistic is another. I don't need your attitude") {                  setTimeout(function() {                      alert("You have completed the game. Refresh the page to try different dialogue options")                  },5000)              }              pressOption(option11.innerText);              option11.classList.remove("ifHover");          }      })      option12.addEventListener("click", function() {          if (option12.innerText == false) {              return;          } else {              if (option12.innerText == "dang. I might be busy. I'll let you know by end of the week" || option12.innerText == "you clearly don't understand what I'm going through") {                  setTimeout(function() {                      alert("You have completed the game. Refresh the page to try different dialogue options")                  },5000)              }              pressOption(option12.innerText);              option12.classList.remove("ifHover");          }      })      option11.addEventListener("mouseover", function() {          if (option11.innerText == false) {              option11.classList.remove("ifHover")          } else {              option11.classList.add("ifHover")          }      })      option11.addEventListener("mouseout", function() {          option11.classList.remove("ifHover")      })      option12.addEventListener("mouseover", function() {          if (option12.innerText == false) {              option12.classList.remove("ifHover")          } else {              option12.classList.add("ifHover")          }      })      option12.addEventListener("mouseout", function() {          option12.classList.remove("ifHover")      })  } |