| Sprint 10, Assignment 6.2 Please also update the doc name with correct numbers. | |
| --- | --- |
| Assignment type: JS Interactive | |
| Assignment name  Can remain the same as the assignment topic name, or…  can be phrased in the active/imperative voice (depending on the task), e.g. “Create your own X” | “Sprint Exit Assignment: Connect Four” |
| Description  The essential tasks a student must perform in order to complete the assignment  ?+ a general, colorful introduction to the assignment? | * U |
| Link to full code file (Answer)  Push a full code file to our [GitHub repo](https://github.com/bitdegree-foundation/academy-assignments-code-files) for this sprint’s assignment and paste a link to that file here. Please write the file in a concise and clear format, according to the “[example-js-code-file](https://github.com/bitdegree-foundation/academy-assignments-code-files/blob/master/example-js-code-file.js)” found in our repo. | (Upload) |

| # | Step  Write each small step of the task. These assignments should be doable by students on their own (without BitDegree-instructor help). It can be any number of steps, but keep them pretty clear and separate (don’t combine 2 steps into 1 step). | 1x hint  Write brief text or partial code that will help the student figure out how to either move forward or completely accomplish the given step. | Output Expected code that our platform will be able to run & check against student submissions |
| --- | --- | --- | --- |
| 1 | Create an HTML document with the following starter code |  | <!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>Connect Four Project (Assignment)</title>  <script *src*="./6.2.Assignment.script.js" *charset*="UTF-8"></script>  <style></style>  </head>  <body></body>  </html> |
| 2 | In the body, create a parent div with the class of "grid", and a child div - with no class, but inner text of the number "1"; check that your webpage is displaying the 1 successfully |  | <!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>Connect Four Project (Assignment)</title>  <script *src*="./6.2.Assignment.script.js" *charset*="UTF-8"></script>  <style></style>  </head>  <body>  <div *id*="grid">  <div>1</div>  </div>  </body>  </html> |
| 3 | Since Connect Four is normally played on a vertical grid 7 spaces wide x 6 spaces tall, create another 41 child divs. Give the innertext of "35" and "42" to their respective ones |  | <!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>Connect Four Project (Assignment)</title>  <script *src*="./6.2.Assignment.script.js" *charset*="UTF-8"></script>  <style></style>  </head>  <body>  <div *id*="grid">  <div>1</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>35</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>42</div>  </div>  </body>  </html> |
| 4 | Create css style entries in your header for 1) the ID of "grid" and 2) any divs that are nested under an element with the ID of "grid":  - give them 1px solid black borders,  - border radius' of 10px,  - backgrounds that contrast (say, dark gray for the "grid" parent, and bright yellow for the children), and  - make the parent "grid" display flex and flex-wrap normally |  | <!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>Connect Four Project (Assignment)</title>  <script *src*="./6.2.Assignment.script.js" *charset*="UTF-8"></script>  <style>  #grid {  border: 1px solid black;  border-radius: 10px;  display: flex;  flex-wrap: wrap;  background: rgb(24, 24, 24);  }  #grid div {  border: 1px solid black;  border-radius: 10px;  background: yellow;  }  </style>  </head>  <body>  <div *id*="grid">  <div>1</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>35</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>42</div>  </div>  </body>  </html> |
| 5 | Give a symmetrical size to your children grid square divs (say, 60px by 60px) in the CSS, and give your parent element enough height and width to frame them. | if you have 7\*60 going across, and 6\*60 going down, that would require at least 420x360 pixels ... but each square also has a border of 1 pixel on each side - so you will need to add 7\*2 & 6\*2 to those totals, respectively! | <!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>Connect Four Project (Assignment)</title>  <script *src*="./6.2.Assignment.script.js" *charset*="UTF-8"></script>  <style>  #grid {  border: 1px solid black;  border-radius: 10px;  display: flex;  flex-wrap: wrap;  background: rgb(24, 24, 24);  height: 372px;  width: 434px;  }  #grid div {  border: 1px solid black;  border-radius: 10px;  background: yellow;  height: 60px;  width: 60px;  }  </style>  </head>  <body>  <div *id*="grid">  <div>1</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>35</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>42</div>  </div>  </body>  </html> |
| 6 | Because Connect 4 disks fall down through the air and need to hit a "floor" to stop, add 7 more children divs after your 42'nd one, and give each of them the classes of "taken" and "floor" |  | <!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>Connect Four Project (Assignment)</title>  <script *src*="./6.2.Assignment.script.js" *charset*="UTF-8"></script>  <style>  #grid {  border: 1px solid black;  border-radius: 10px;  display: flex;  flex-wrap: wrap;  background: rgb(24, 24, 24);  height: 372px;  width: 434px;  }  #grid div {  border: 1px solid black;  border-radius: 10px;  background: yellow;  height: 60px;  width: 60px;  }  </style>  </head>  <body>  <div *id*="grid">  <div>1</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>35</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>42</div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  </div>  </body>  </html> |
| 7 | In order to hide these floor tiles from the player, create a new css entry for elements with the class of "floor"; give them borders that are 1 pixel, solid and white, and backgrounds that are white. Make sure to have these CSS properties override any conflicting, earlier formatting. | Hint: use !important | <!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>Connect Four Project (Assignment)</title>  <script *src*="./6.2.Assignment.script.js" *charset*="UTF-8"></script>  <style>  #grid {  border: 1px solid black;  border-radius: 10px;  display: flex;  flex-wrap: wrap;  background: rgb(24, 24, 24);  height: 372px;  width: 434px;  }  #grid div {  border: 1px solid black;  border-radius: 10px;  background: yellow;  height: 60px;  width: 60px;  }  .floor {  border: 1px solid white !important;  background: white !important;  }  </style>  </head>  <body>  <div *id*="grid">  <div>1</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>35</div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div></div>  <div>42</div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  <div *class*="taken floor"></div>  </div>  </body>  </html> |
| 8a | Create a Javascript file named "6.2.Assignment.script.js" |  | <!DOCTYPE *html*>  …  .player-one {  background-color: red !important;  border-radius: 10px;  }  .player-two {  background-color: blue !important;  border-radius: 10px;  }  …  <body>  <h3 *id*="result"></h3>  …  </body>  </html> |
| 8b | Add an event listener to the document that, when the DOM content is loaded, will log "Content loaded succesfully!"; check that your JavaScript is being loaded by your HTML. |  | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  }) |
| 9 | Using query selectors, store all of your child divs in a constant called "squares"; log them - your node list should be 49 elements long. | Hint: since you're asking for more than one element, you will need to use querySelectorAll  Hint: Remember that you can query the children of a parent element using a ">" | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  }) |
| 10 | Create a 'for' loop that will itterate through the squares array (using "i"). Give the loop an event listener that will alert "Can't go here!" when a square is clicked. | Hint: remember that though clicks can be listened to in multiple ways, they are such common events that a special .onclick syntax exists! | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  alert("Can't go here!");  }  }  }) |
| 11 | Add a conditional to your onclick listener that checks whether the square is taken, and logs "This square is available" if true | Hint: you can make your conditional check whether something is \*not\* true by using a "!" prefix | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  alert("Can't go here!");  }  }  }) |
| 12 | Add a second conditional to your onclick listener that checks whether the square below is taken, and logs "The square BELOW this is taken" if true (you should have two "if" statements by the end of this step) | Hint: You want your code to ask "Does the square 7 spaces after the one I've clicked on have a class of "taken"?  Hint: use .classList.contains  Note: Currently, this log message should only trigger when you click on any of the bottom seven squares (#36-42), while the first log message and the alert should trigger for all squares. | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  alert("Can't go here!");  }  }  }) |
| 13 | Combine your two conditionals into a third one that logs "Player CAN go here!!" if true. Move your alert so that it is the result of either condition being false | Hint: use && gates to combine your two conditions  Hint: use "else" to trigger your alert  Note: Now, you should have two behaviors - if you try to place a piece higher-up in the grid, the space is technically open ... but gravity would not let it stay there (so the player gets an alert). If you try to place a piece on the bottom row, the space is logged as both open \*and\* having a piece below it, mimicking a piece falling to the bottom (or as far as it can). Our combo-conditional will now allow us to store valid moves! | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  } else alert("Can't go here!");  }  }  }) |
| 14 | At the beginning of your JavaScript, declare currentPlayer (using let) as the string "Player 1" (Make sure to declare it before your 'for' loop).  In your HTMl's "style" section, add a class of "player-one", give it background-color of red !important, and a border radius of 10 pixels. |  | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  let currentPlayer = "Player 1"    for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  } else alert("Can't go here!");  }  }  }) |
| 15 | (Back in your JavaScript) In your third, combo-conditional, add a nested conditional (after your "player CAN go here!" message) to check whether currentPlayer is player 1 - if it is, add the classes of "taken" and "player-one" to the given square | Hint: make sure you're checking for equivalency and not accidentally redefining currentPpayer  Note: the parent conditional is checking to see whether the space is a valid 'move' - the child conditional is checking to see which player is making it (and what player color to change the space into) | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  let currentPlayer = "Player 1"    for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  if (currentPlayer == "Player 1"){  squares[i].classList.add('taken');  squares[i].classList.add('player-one');  }  } else alert("Can't go here!");  }  }  }) |
| 16 | On your webpage, test whether you can turn squares red. If succseful, (in your JavaScript) make your nested conditional also re-define currentPlayer as "Player 2".  In your HTML, add a player-two class to your styling, with background color of blue important, and border-radius of 10 pixels. | Hint: your nested conditional should effectively read, "If currentPlayer is Player 1, add the 'taken' and 'player-one' classes to the current square and update currentPlayer to be Player 2" | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  let currentPlayer = "Player 1"    for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  if (currentPlayer == "Player 1"){  squares[i].classList.add('taken');  squares[i].classList.add('player-one');  currentPlayer = "Player 2"  }  } else alert("Can't go here!");  }  }  }) |
| 17 | Add an else statement to your nested conditional, have it check whether currentPlayer is Player 2 and (if 'yes'), add the 'taken' and 'player-two' classes to the square, and set currentPlayer back to Player 1 | Hint: follow your nested 'if' with 'else if' | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  let currentPlayer = "Player 1"  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  if (currentPlayer == "Player 1"){  squares[i].classList.add('taken');  squares[i].classList.add('player-one');  currentPlayer = "Player 2"  } else if (currentPlayer == "Player 2"){  squares[i].classList.add('taken');  squares[i].classList.add('player-two');  currentPlayer = "Player 1"  }  } else alert("Can't go here!");  }  }  }) |
| 18 | After your alert call, invoke a function called verifyWin (which you will declare in the next step) - use "verifyWin()". | Hint: you should have  … alert("Can't go here!")  verifyWin() |  |
| 19 | At the beginning of your JavaScript, declare a constant named "winningArrays" and store in it the following set of arrays:  [  [0, 1, 2, 3],  [41, 40, 39, 38],  [7, 8, 9, 10],  [34, 33, 32, 31],  [14, 15, 16, 17],  [27, 26, 25, 24],  [21, 22, 23, 24],  [20, 19, 18, 17],  [28, 29, 30, 31],  [13, 12, 11, 10],  [35, 36, 37, 38],  [6, 5, 4, 3],  [0, 7, 14, 21],  [41, 34, 27, 20],  [1, 8, 15, 22],  [40, 33, 26, 19],  [2, 9, 16, 23],  [39, 32, 25, 18],  [3, 10, 17, 24],  [38, 31, 24, 17],  [4, 11, 18, 25],  [37, 30, 23, 16],  [5, 12, 19, 26],  [36, 29, 22, 15],  [6, 13, 20, 27],  [35, 28, 21, 14],  [0, 8, 16, 24],  [41, 33, 25, 17],  [7, 15, 23, 31],  [34, 26, 18, 10],  [14, 22, 30, 38],  [27, 19, 11, 3],  [35, 29, 23, 17],  [6, 12, 18, 24],  [28, 22, 16, 10],  [13, 19, 25, 31],  [21, 15, 9, 3],  [20, 26, 32, 38],  [36, 30, 24, 18],  [5, 11, 17, 23],  [37, 31, 25, 19],  [4, 10, 16, 22],  [2, 10, 18, 26],  [39, 31, 23, 15],  [1, 9, 17, 25],  [40, 32, 24, 16],  [9, 17, 25, 33],  [8, 16, 24, 32],  [11, 17, 23, 29],  [12, 18, 24, 30],  [1, 2, 3, 4],  [5, 4, 3, 2],  [8, 9, 10, 11],  [12, 11, 10, 9],  [15, 16, 17, 18],  [19, 18, 17, 16],  [22, 23, 24, 25],  [26, 25, 24, 23],  [29, 30, 31, 32],  [33, 32, 31, 30],  [36, 37, 38, 39],  [40, 39, 38, 37],  [7, 14, 21, 28],  [8, 15, 22, 29],  [9, 16, 23, 30],  [10, 17, 24, 31],  [11, 18, 25, 32],  [12, 19, 26, 33],  [13, 20, 27, 34],  ] |  |  |
| 20 | Declare the function "verifyWin" before your 'for' loop. Declare another 'for' loop \*within\* verifyWin. Make that loop check each set of numbers in winningArrays. Make the loop also declare four variables - called square1, square2, square3, square4. Make the loop store each of the four numbers of the given winning array in these variables. Log the variables. | Hint: the \*last\* set in winning Arrays is [13,20,27,34], so, you should see your loop's square1 containing index 0's number ("13"), square2 containing the second index ("20"), square3 containing the number "27", and square four containing the number "34". | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  let currentPlayer = "Player 1"  const winningArrays = [  [0, 1, 2, 3],  [41, 40, 39, 38],  [7, 8, 9, 10],  [34, 33, 32, 31],  [14, 15, 16, 17],  [27, 26, 25, 24],  [21, 22, 23, 24],  [20, 19, 18, 17],  [28, 29, 30, 31],  [13, 12, 11, 10],  [35, 36, 37, 38],  [6, 5, 4, 3],  [0, 7, 14, 21],  [41, 34, 27, 20],  [1, 8, 15, 22],  [40, 33, 26, 19],  [2, 9, 16, 23],  [39, 32, 25, 18],  [3, 10, 17, 24],  [38, 31, 24, 17],  [4, 11, 18, 25],  [37, 30, 23, 16],  [5, 12, 19, 26],  [36, 29, 22, 15],  [6, 13, 20, 27],  [35, 28, 21, 14],  [0, 8, 16, 24],  [41, 33, 25, 17],  [7, 15, 23, 31],  [34, 26, 18, 10],  [14, 22, 30, 38],  [27, 19, 11, 3],  [35, 29, 23, 17],  [6, 12, 18, 24],  [28, 22, 16, 10],  [13, 19, 25, 31],  [21, 15, 9, 3],  [20, 26, 32, 38],  [36, 30, 24, 18],  [5, 11, 17, 23],  [37, 31, 25, 19],  [4, 10, 16, 22],  [2, 10, 18, 26],  [39, 31, 23, 15],  [1, 9, 17, 25],  [40, 32, 24, 16],  [9, 17, 25, 33],  [8, 16, 24, 32],  [11, 17, 23, 29],  [12, 18, 24, 30],  [1, 2, 3, 4],  [5, 4, 3, 2],  [8, 9, 10, 11],  [12, 11, 10, 9],  [15, 16, 17, 18],  [19, 18, 17, 16],  [22, 23, 24, 25],  [26, 25, 24, 23],  [29, 30, 31, 32],  [33, 32, 31, 30],  [36, 37, 38, 39],  [40, 39, 38, 37],  [7, 14, 21, 28],  [8, 15, 22, 29],  [9, 16, 23, 30],  [10, 17, 24, 31],  [11, 18, 25, 32],  [12, 19, 26, 33],  [13, 20, 27, 34],  ]  function verifyWin() {  for (let y = 0; y < winningArrays.length; y++) {  const square1 = winningArrays[y][0];  console.log("This is square 1");  console.log(square1);  const square2 = winningArrays[y][1];  console.log("This is square 2");  console.log(square2);  const square3 = winningArrays[y][2];  console.log("This is square 3");  console.log(square3);  const square4 = winningArrays[y][3];  console.log("This is square 4");  console.log(square4);  }}  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  if (currentPlayer == "Player 1"){  squares[i].classList.add('taken');  squares[i].classList.add('player-one');  currentPlayer = "Player 2"  } else if (currentPlayer == "Player 2"){  squares[i].classList.add('taken');  squares[i].classList.add('player-two');  currentPlayer = "Player 1"  }  } else alert("Can't go here!");  verifyWin()  }  }  }) |
| 21 | nstead of making your loop store winningArray's numbers, now use them to store the variable square's div objects | Hint: use winningArrays' numbers to find and store div objects of the equivilent index place number (in the "squares" array). So, if the first set in winningArray has [0,1,2,3], how would you use that to store the zero'th, first, second, and third elements of the "squares" array? How would you use [13,20,27,34] to get the 13th, 20th, 27th, and 34th elements? | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  let currentPlayer = "Player 1"  const winningArrays = [  [0, 1, 2, 3],  [41, 40, 39, 38],  [7, 8, 9, 10],  [34, 33, 32, 31],  [14, 15, 16, 17],  [27, 26, 25, 24],  [21, 22, 23, 24],  [20, 19, 18, 17],  [28, 29, 30, 31],  [13, 12, 11, 10],  [35, 36, 37, 38],  [6, 5, 4, 3],  [0, 7, 14, 21],  [41, 34, 27, 20],  [1, 8, 15, 22],  [40, 33, 26, 19],  [2, 9, 16, 23],  [39, 32, 25, 18],  [3, 10, 17, 24],  [38, 31, 24, 17],  [4, 11, 18, 25],  [37, 30, 23, 16],  [5, 12, 19, 26],  [36, 29, 22, 15],  [6, 13, 20, 27],  [35, 28, 21, 14],  [0, 8, 16, 24],  [41, 33, 25, 17],  [7, 15, 23, 31],  [34, 26, 18, 10],  [14, 22, 30, 38],  [27, 19, 11, 3],  [35, 29, 23, 17],  [6, 12, 18, 24],  [28, 22, 16, 10],  [13, 19, 25, 31],  [21, 15, 9, 3],  [20, 26, 32, 38],  [36, 30, 24, 18],  [5, 11, 17, 23],  [37, 31, 25, 19],  [4, 10, 16, 22],  [2, 10, 18, 26],  [39, 31, 23, 15],  [1, 9, 17, 25],  [40, 32, 24, 16],  [9, 17, 25, 33],  [8, 16, 24, 32],  [11, 17, 23, 29],  [12, 18, 24, 30],  [1, 2, 3, 4],  [5, 4, 3, 2],  [8, 9, 10, 11],  [12, 11, 10, 9],  [15, 16, 17, 18],  [19, 18, 17, 16],  [22, 23, 24, 25],  [26, 25, 24, 23],  [29, 30, 31, 32],  [33, 32, 31, 30],  [36, 37, 38, 39],  [40, 39, 38, 37],  [7, 14, 21, 28],  [8, 15, 22, 29],  [9, 16, 23, 30],  [10, 17, 24, 31],  [11, 18, 25, 32],  [12, 19, 26, 33],  [13, 20, 27, 34],  ]  function verifyWin() {  for (let y = 0; y < winningArrays.length; y++) {  const square1 = squares[winningArrays[y][0]];  console.log("This is square 1");  console.log(square1);  const square2 = squares[winningArrays[y][1]];  console.log("This is square 2");  console.log(square2);  const square3 = squares[winningArrays[y][2]];  console.log("This is square 3");  console.log(square3);  const square4 = squares[winningArrays[y][3]];  console.log("This is square 4");  console.log(square4);  }}  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  if (currentPlayer == "Player 1"){  squares[i].classList.add('taken');  squares[i].classList.add('player-one');  currentPlayer = "Player 2"  } else if (currentPlayer == "Player 2"){  squares[i].classList.add('taken');  squares[i].classList.add('player-two');  currentPlayer = "Player 1"  }  } else alert("Can't go here!");  verifyWin()  }  }  }) |
| 22 | At the very beginning of your HTML's body, add h3 tags with the ID of "#result". Back in your JavaScript (and before your loops), declare a constant also named "result", and store the HTML element in it. Log your variable to make sure it is working | Hint: use querySelector | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  let currentPlayer = "Player 1"  const result = document.querySelector('#result')  console.log(result);  const winningArrays = [  [0, 1, 2, 3],  [41, 40, 39, 38],  [7, 8, 9, 10],  [34, 33, 32, 31],  [14, 15, 16, 17],  [27, 26, 25, 24],  [21, 22, 23, 24],  [20, 19, 18, 17],  [28, 29, 30, 31],  [13, 12, 11, 10],  [35, 36, 37, 38],  [6, 5, 4, 3],  [0, 7, 14, 21],  [41, 34, 27, 20],  [1, 8, 15, 22],  [40, 33, 26, 19],  [2, 9, 16, 23],  [39, 32, 25, 18],  [3, 10, 17, 24],  [38, 31, 24, 17],  [4, 11, 18, 25],  [37, 30, 23, 16],  [5, 12, 19, 26],  [36, 29, 22, 15],  [6, 13, 20, 27],  [35, 28, 21, 14],  [0, 8, 16, 24],  [41, 33, 25, 17],  [7, 15, 23, 31],  [34, 26, 18, 10],  [14, 22, 30, 38],  [27, 19, 11, 3],  [35, 29, 23, 17],  [6, 12, 18, 24],  [28, 22, 16, 10],  [13, 19, 25, 31],  [21, 15, 9, 3],  [20, 26, 32, 38],  [36, 30, 24, 18],  [5, 11, 17, 23],  [37, 31, 25, 19],  [4, 10, 16, 22],  [2, 10, 18, 26],  [39, 31, 23, 15],  [1, 9, 17, 25],  [40, 32, 24, 16],  [9, 17, 25, 33],  [8, 16, 24, 32],  [11, 17, 23, 29],  [12, 18, 24, 30],  [1, 2, 3, 4],  [5, 4, 3, 2],  [8, 9, 10, 11],  [12, 11, 10, 9],  [15, 16, 17, 18],  [19, 18, 17, 16],  [22, 23, 24, 25],  [26, 25, 24, 23],  [29, 30, 31, 32],  [33, 32, 31, 30],  [36, 37, 38, 39],  [40, 39, 38, 37],  [7, 14, 21, 28],  [8, 15, 22, 29],  [9, 16, 23, 30],  [10, 17, 24, 31],  [11, 18, 25, 32],  [12, 19, 26, 33],  [13, 20, 27, 34],  ]  function verifyWin() {  for (let y = 0; y < winningArrays.length; y++) {  const square1 = squares[winningArrays[y][0]];  console.log("This is square 1");  console.log(square1);  const square2 = squares[winningArrays[y][1]];  console.log("This is square 2");  console.log(square2);  const square3 = squares[winningArrays[y][2]];  console.log("This is square 3");  console.log(square3);  const square4 = squares[winningArrays[y][3]];  console.log("This is square 4");  console.log(square4);  }}  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  if (currentPlayer == "Player 1"){  squares[i].classList.add('taken');  squares[i].classList.add('player-one');  currentPlayer = "Player 2"  } else if (currentPlayer == "Player 2"){  squares[i].classList.add('taken');  squares[i].classList.add('player-two');  currentPlayer = "Player 1"  }  } else alert("Can't go here!");  verifyWin()  }  }  }) |
| 23 | Within verifyWin's 'for' loop, declare a conditional after your four variables (square1 through square4) that will check to see if each variable has a class of 'player-one' - and will set result's innerHTML to 'Player One Wins!' if true | Hint: target square1, use classList, and ask if it contains the class 'player-one'  Hint: if (...) {result.innerHTML = 'Player One Wins!'} | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  let currentPlayer = "Player 1"  const result = document.querySelector('#result')  console.log(result);  const winningArrays = [  [0, 1, 2, 3],  [41, 40, 39, 38],  [7, 8, 9, 10],  [34, 33, 32, 31],  [14, 15, 16, 17],  [27, 26, 25, 24],  [21, 22, 23, 24],  [20, 19, 18, 17],  [28, 29, 30, 31],  [13, 12, 11, 10],  [35, 36, 37, 38],  [6, 5, 4, 3],  [0, 7, 14, 21],  [41, 34, 27, 20],  [1, 8, 15, 22],  [40, 33, 26, 19],  [2, 9, 16, 23],  [39, 32, 25, 18],  [3, 10, 17, 24],  [38, 31, 24, 17],  [4, 11, 18, 25],  [37, 30, 23, 16],  [5, 12, 19, 26],  [36, 29, 22, 15],  [6, 13, 20, 27],  [35, 28, 21, 14],  [0, 8, 16, 24],  [41, 33, 25, 17],  [7, 15, 23, 31],  [34, 26, 18, 10],  [14, 22, 30, 38],  [27, 19, 11, 3],  [35, 29, 23, 17],  [6, 12, 18, 24],  [28, 22, 16, 10],  [13, 19, 25, 31],  [21, 15, 9, 3],  [20, 26, 32, 38],  [36, 30, 24, 18],  [5, 11, 17, 23],  [37, 31, 25, 19],  [4, 10, 16, 22],  [2, 10, 18, 26],  [39, 31, 23, 15],  [1, 9, 17, 25],  [40, 32, 24, 16],  [9, 17, 25, 33],  [8, 16, 24, 32],  [11, 17, 23, 29],  [12, 18, 24, 30],  [1, 2, 3, 4],  [5, 4, 3, 2],  [8, 9, 10, 11],  [12, 11, 10, 9],  [15, 16, 17, 18],  [19, 18, 17, 16],  [22, 23, 24, 25],  [26, 25, 24, 23],  [29, 30, 31, 32],  [33, 32, 31, 30],  [36, 37, 38, 39],  [40, 39, 38, 37],  [7, 14, 21, 28],  [8, 15, 22, 29],  [9, 16, 23, 30],  [10, 17, 24, 31],  [11, 18, 25, 32],  [12, 19, 26, 33],  [13, 20, 27, 34],  ]  function verifyWin() {  for (let y = 0; y < winningArrays.length; y++) {  const square1 = squares[winningArrays[y][0]];  console.log("This is square 1");  console.log(square1);  const square2 = squares[winningArrays[y][1]];  console.log("This is square 2");  console.log(square2);  const square3 = squares[winningArrays[y][2]];  console.log("This is square 3");  console.log(square3);  const square4 = squares[winningArrays[y][3]];  console.log("This is square 4");  console.log(square4);  if (  square1.classList.contains('player-one') &&  square2.classList.contains('player-one') &&  square3.classList.contains('player-one') &&  square4.classList.contains('player-one')  ) {  result.innerHTML = 'Player One Wins!'  }  }}  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  if (currentPlayer == "Player 1"){  squares[i].classList.add('taken');  squares[i].classList.add('player-one');  currentPlayer = "Player 2"  } else if (currentPlayer == "Player 2"){  squares[i].classList.add('taken');  squares[i].classList.add('player-two');  currentPlayer = "Player 1"  }  } else alert("Can't go here!");  verifyWin()  }  }  }) |
| 24 | Repeat step 23, but for player-2 |  | document.addEventListener('DOMContentLoaded', () => {  console.log("Content loaded succesfully!");  const squares = document.querySelectorAll('#grid > div');  console.log(squares);  let currentPlayer = "Player 1"  const result = document.querySelector('#result')  console.log(result);  const winningArrays = [  [0, 1, 2, 3],  [41, 40, 39, 38],  [7, 8, 9, 10],  [34, 33, 32, 31],  [14, 15, 16, 17],  [27, 26, 25, 24],  [21, 22, 23, 24],  [20, 19, 18, 17],  [28, 29, 30, 31],  [13, 12, 11, 10],  [35, 36, 37, 38],  [6, 5, 4, 3],  [0, 7, 14, 21],  [41, 34, 27, 20],  [1, 8, 15, 22],  [40, 33, 26, 19],  [2, 9, 16, 23],  [39, 32, 25, 18],  [3, 10, 17, 24],  [38, 31, 24, 17],  [4, 11, 18, 25],  [37, 30, 23, 16],  [5, 12, 19, 26],  [36, 29, 22, 15],  [6, 13, 20, 27],  [35, 28, 21, 14],  [0, 8, 16, 24],  [41, 33, 25, 17],  [7, 15, 23, 31],  [34, 26, 18, 10],  [14, 22, 30, 38],  [27, 19, 11, 3],  [35, 29, 23, 17],  [6, 12, 18, 24],  [28, 22, 16, 10],  [13, 19, 25, 31],  [21, 15, 9, 3],  [20, 26, 32, 38],  [36, 30, 24, 18],  [5, 11, 17, 23],  [37, 31, 25, 19],  [4, 10, 16, 22],  [2, 10, 18, 26],  [39, 31, 23, 15],  [1, 9, 17, 25],  [40, 32, 24, 16],  [9, 17, 25, 33],  [8, 16, 24, 32],  [11, 17, 23, 29],  [12, 18, 24, 30],  [1, 2, 3, 4],  [5, 4, 3, 2],  [8, 9, 10, 11],  [12, 11, 10, 9],  [15, 16, 17, 18],  [19, 18, 17, 16],  [22, 23, 24, 25],  [26, 25, 24, 23],  [29, 30, 31, 32],  [33, 32, 31, 30],  [36, 37, 38, 39],  [40, 39, 38, 37],  [7, 14, 21, 28],  [8, 15, 22, 29],  [9, 16, 23, 30],  [10, 17, 24, 31],  [11, 18, 25, 32],  [12, 19, 26, 33],  [13, 20, 27, 34],  ]  function verifyWin() {  for (let y = 0; y < winningArrays.length; y++) {  const square1 = squares[winningArrays[y][0]];  console.log("This is square 1");  console.log(square1);  const square2 = squares[winningArrays[y][1]];  console.log("This is square 2");  console.log(square2);  const square3 = squares[winningArrays[y][2]];  console.log("This is square 3");  console.log(square3);  const square4 = squares[winningArrays[y][3]];  console.log("This is square 4");  console.log(square4);  if (  square1.classList.contains('player-one') &&  square2.classList.contains('player-one') &&  square3.classList.contains('player-one') &&  square4.classList.contains('player-one')  ) {  result.innerHTML = 'Player One Wins!'  }  if (  square1.classList.contains('player-two') &&  square2.classList.contains('player-two') &&  square3.classList.contains('player-two') &&  square4.classList.contains('player-two')  ) {  result.innerHTML = 'Player Two Wins!'  }  }}  for (let i = 0; i < squares.length; i++) {  squares[i].onclick = function() {  if (!squares[i].classList.contains('taken')){  console.log("This square is available")  }  if (squares[i + 7].classList.contains('taken')){  console.log("The square BELOW this is taken");  }  if (  !squares[i].classList.contains('taken')  &&  squares[i + 7].classList.contains('taken')){  console.log("Player CAN go here!!");  if (currentPlayer == "Player 1"){  squares[i].classList.add('taken');  squares[i].classList.add('player-one');  currentPlayer = "Player 2"  } else if (currentPlayer == "Player 2"){  squares[i].classList.add('taken');  squares[i].classList.add('player-two');  currentPlayer = "Player 1"  }  } else alert("Can't go here!");  verifyWin()  }  }  }) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| ... |  |  |  |