**Szabályok:**

A=1, J=11, Q=12, K=13

11 pontig

3-3 lap a játékosoknak, 4 az asztalra

random kezdés

Kombinációk kirakása, maximum 13, az elvett lapokat, és a sajátokat is magunk mellé kell helyezni, ha nem tudsz ütni, vagy nem akarsz, le kell tenni egy lapot a többi közé. Akkor van vége a körnek, ha mindkét játékos kezéből elfogy a lap.

Osztásnál 3-3 lapot kap minkét játékos, először, aki utoljára ütött. Ő is kezdi a kört. A 8. Kört nyílt lapokkal játszák.

3 pont – legtöbb lap

2 pont – legtöbb pikk

1 pont – minden ász

2 pont - ♦ **10-es**

1 pont - ♠ 2-es

1 pont minden tábláért.

Akkor tábla, ha minden asztalon lévő lapot elviszel, ezután az ellenfél jön kétszer

Ha a végén maradnak lapok, azokat az viszi aki az utolsó ütést csinálta

**Elvárások**

Normális játékmenet a számítógép ellen

A játék play gombra indul

Menü, ahol a játék újrakezdhető és megtekinthető egy statisztika, ami mutatja a rekordot

A kártya egy kétoldalú objektum, ami megfordul

Odaúszás animáció, ütés esetén kártyák megjelenítése és összehasonlítása

**Folyamat**

Create a card object that has a back and a front and can be flipped.

Create a deck of cards

Shuffle deck

Try and create a card object dinamically with JS that displays a random card from the deck as a card object, similar tot he one in the first step

Generate the play area with js when you click the deck

Select and deselect cards

Implement take logic

display taken cards

error message if numbers are not correct

Handle table event

implement cpu logic

make the game turn based

count score

handle game over mechanic

Handle extra game mechanic if neither player has >= 11 points

Menu: Rules, New game, stats

Animations

**Felépítés**

**1. HTML Structure:**

**Step 1: Document Structure**

* Start by creating the basic HTML document structure, including <!DOCTYPE html>, <html>, <head>, and <body>.

**Step 2: Title and Heading**

* Add a title for your game using the <title> tag.
* Include a heading (e.g., <h1>) to display the name of your game.

**Step 3: Containers**

* Create div elements to serve as containers for different sections of your game, such as the game table and the player's hand.

**2. CSS Styling:**

**Step 4: Layout Styling**

* Apply styles to set the overall layout, background colors, fonts, and spacing.
* Ensure that the game elements are visually organized and appealing.

**Step 5: Card Styling**

* Style the appearance of the cards, both on the table and in the player's hand.
* Consider using CSS classes to represent different card suits and ranks.

**3. JavaScript Initialization:**

**Step 6: Defining Card Values**

* Create arrays to represent the suits and ranks of the cards.

**Step 7: Creating a Deck**

* Generate a deck of cards by combining the suits and ranks.

**Step 8: Shuffling the Deck**

* Implement a function to shuffle the deck randomly.

**Step 9: Dealing Cards**

* Create a function to deal cards to the player and the table.
* Update the HTML and CSS to display the initial state of the game.

**4. Gameplay Logic:**

**Step 10: Player's Turn**

* Allow the player to select a card from their hand and make a move on the table.
* Implement basic card selection and placement logic.

**Step 11: Capturing Cards**

* Implement rules for capturing cards based on matching ranks.
* Update the game state to reflect card captures.

**Step 12: Building**

* Add logic for building on existing cards on the table.
* Allow the player to announce builds and capture cards accordingly.

**Step 13: Scoring**

* Implement scoring rules, considering different combinations and card captures.
* Update the display to show the current score.

**5. Game Flow:**

**Step 14: Turns and Rounds**

* Manage turns between the player and possibly other opponents.
* Determine how the game progresses from one turn to the next.

**Step 15: Ending the Game**

* Define the conditions for ending the game.
* Declare a winner and display the result.

**6. User Interface Updates:**

**Step 16: Updating the Table**

* Dynamically update the display of cards on the table based on the game state.
* Ensure that card movements and captures are visually represented.

**Step 17: Player's Hand**

* Update the player's hand based on the cards they have.
* Provide visual feedback for the player's actions.

**Step 18: Scoring Display**

* Show the current score and any other relevant information on the user interface.

**7. Event Handling:**

**Step 19: Click Events**

* Handle clicks on cards to trigger moves.
* Implement event listeners for user interactions.

**Step 20: Input Validation**

* Ensure that player moves are valid based on the game rules.
* Provide feedback to the player if an invalid move is attempted.

**8. Testing and Debugging:**

**Step 21: Console Logging**

* Use console.log statements to debug and understand the flow of your code.
* Verify that variables and functions are behaving as expected.