

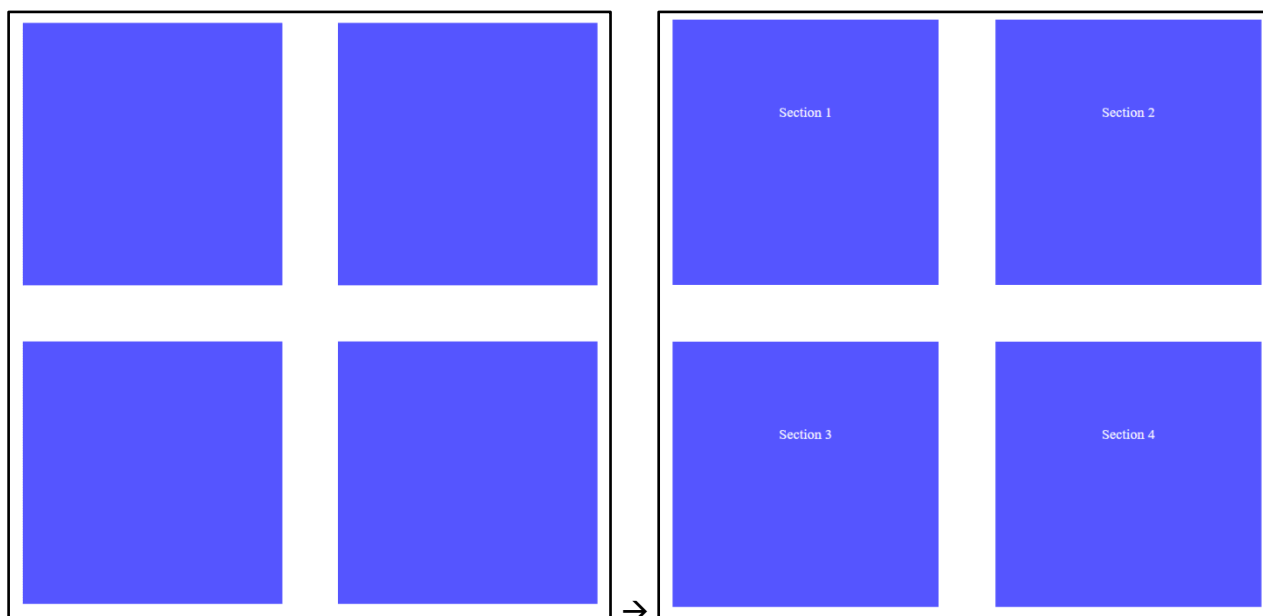
Exercise: DOM Manipulations

Problems for exercises and homework for the ["JavaScript Advanced" course @ SoftUni](https://softuni.org/courses/javascript-advanced). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1550/Exercise-DOM-Manipulations>.

1. Sections

You will receive an **array** of strings. For each string, create a **div** with a **paragraph** with the **string** in it. Each paragraph is initially **hidden (display:none)**. Add a **click event listener** to **each div** that **displays** the **hidden** paragraph. Finally, you should **append** all divs to the element with an **id "content"**.

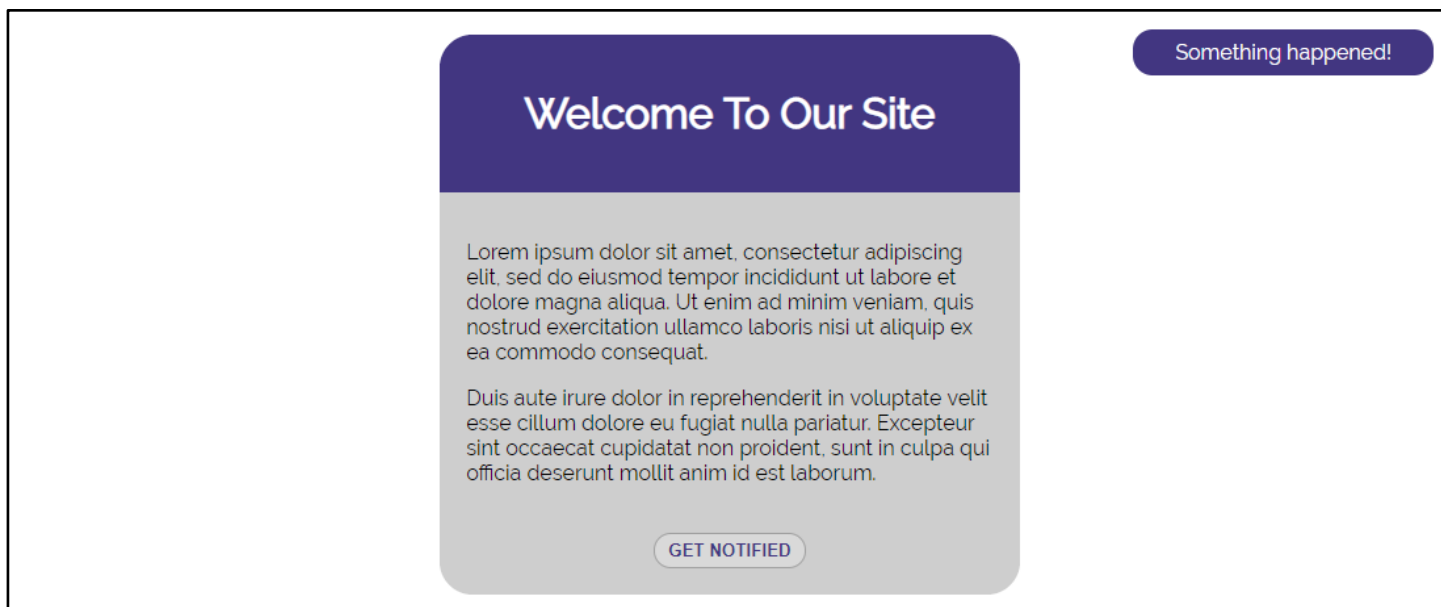
Example



2. Notification

Write a **function** that receives a string **message** and **displays** it inside a div with an id **"notification"** for 2 seconds. The div is initially **hidden** and when the function is called, it must be **shown**. After 2 seconds, **hide** the div. In the example below, a notification is shown when you **click** the button.

Example



When we click the **[GET NOTIFIED]** button, a **div** appears in our upper-right corner. It should **disappear** in 2 seconds.

3. Time Converter

Create a program that **converts** different time units. Your task is to add a **click** event listener to **all [CONVERT] buttons**. When a button is **clicked**, read the **corresponding** input field, **convert** the value to the **three other** time units and **display** it in the input fields.

Example

Time Converter

Days:

CONVERT

Hours:

CONVERT

Minutes:

CONVERT


Seconds:

CONVERT

One day is equal to 24 hours/1440 minutes/86400 seconds. Whichever button we **click**, the input fields should **change** depending on the added value on the left. (For example, if we write 48 hours and click convert the days, the field value should change to 2).

4. Locked Profile

In this problem, you should **create a JS functionality** which **shows** and **hides** the additional information about users.



Lock • Unlock •

Username

User 1 Userov

Show more



Lock • Unlock •

Username

User 2 Userov

Show more



Lock • Unlock •

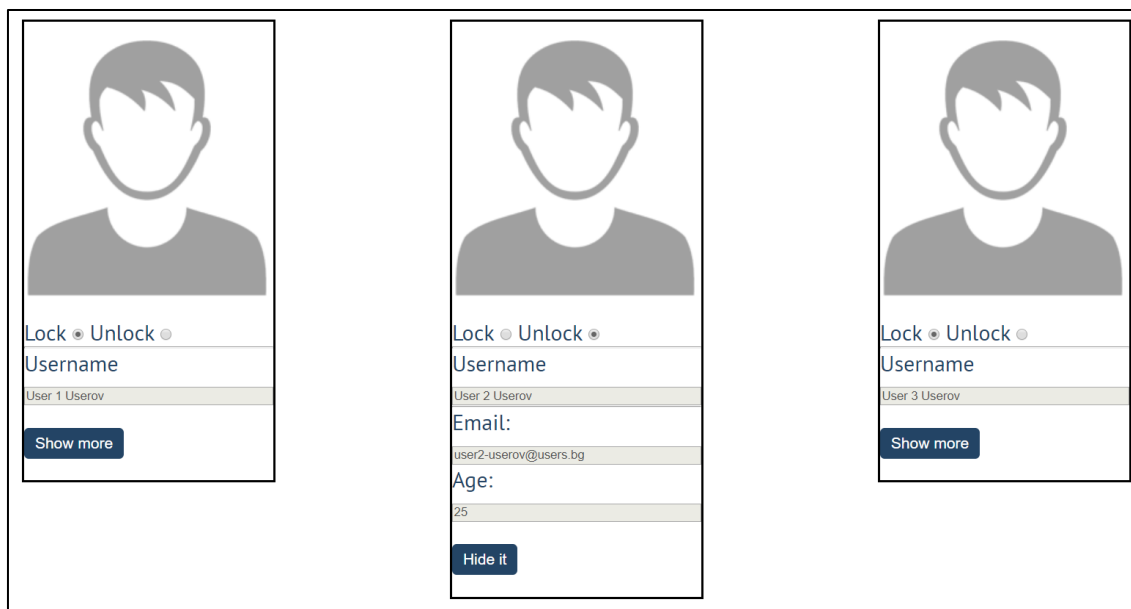
Username

User 3 Userov

Show more

When one of the [Show more] buttons is clicked, the **hidden information** inside the div should

be shown, only if **the profile is not locked**! If the current profile is **locked**, nothing should happen.

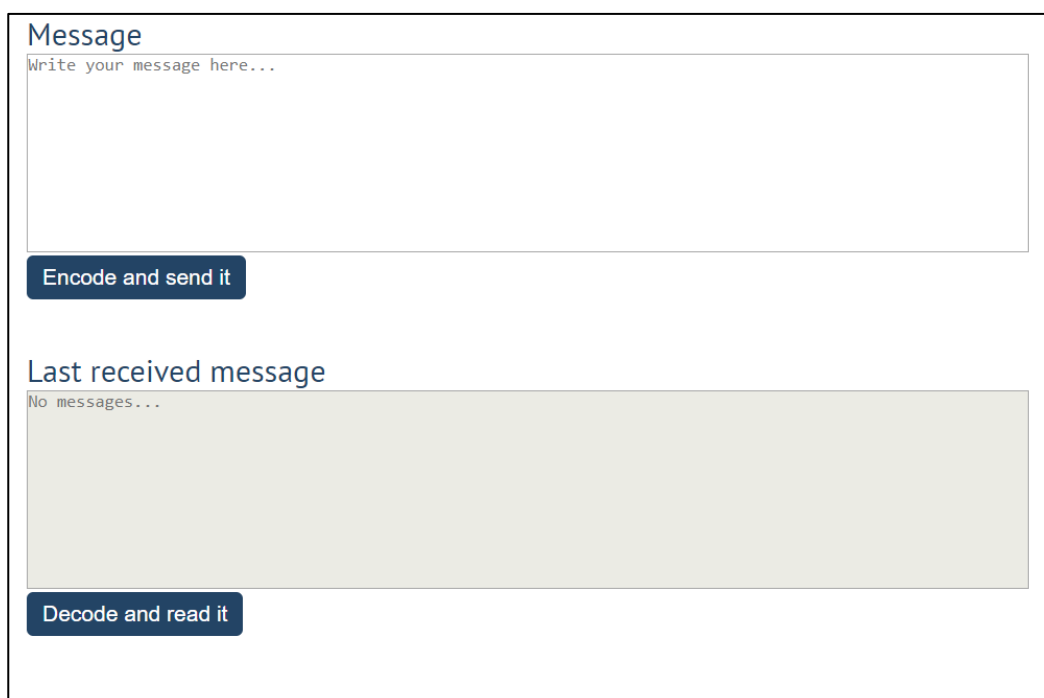


The image shows three user profile cards. Each card has a placeholder for a profile picture at the top. Below the picture is a status bar with a 'Lock' button and an 'Unlock' button. Underneath is a 'Username' field. The first and third cards have a 'Show more' button. The middle card has a 'Hide it' button and displays additional information: an 'Email' field with the value 'user2-userov@users.bg' and an 'Age' field with the value '25'.

If the **hidden information is displayed** and we **lock the profile again**, the **[Hide it]** button should **not be working**! Otherwise, when the profile is **unlocked** and we click on the **[Hide it]** button, the new fields must hide again.

5. Encode and Decode Messages

In this problem, you should **create a JS functionality** which **encodes and decodes some messages** which travel to the network.



The image shows a message interface. At the top is a 'Message' section with a text input field containing the placeholder 'Write your message here...'. Below the input field is a button labeled 'Encode and send it'. Below this is a 'Last received message' section with a text area containing the placeholder 'No messages...'. At the bottom of this section is a button labeled 'Decode and read it'.

This program should contain **two functionalities**.

The first one is to **encode the given message** and **send it** to the **receiver**.

The second one is to **decode the received message** and **read it (display it)**.

When the **[Encode and send it] button** is clicked, you should get the given message from the first textarea. When you get the current message, you should encode it as follows:

- **Change the ASCII CODE on every single character** in that message when you **add 1** to the current **ASCII NUMBER**, that represent the current character in that message
- **Clear the sender textarea** and **append** the encoded message to the **receiver textarea**

Message

The password for my bank account is 123pass321

Encode and send it

Last received message

No messages...

Decode and read it

After clicking **[Encode and send it] button** the result should be:

Message

Write your message here...

Encode and send it

Last received message

Uif!qbttxpse!gps!nz!cbo!bddpvou!jt!234qbt432

Decode and read it

After that, when the **[Decode and read it] button** is clicked. You need to get the **encoded message** from the **receiver textarea** and do the **opposite logic** from encoding:

- **Subtract 1** from the current **ASCII NUMBER**, that represents the current character in that message
- Replace the **encoded message** with the already **decoded message** in the receiver textrea, to make it readable

Message

Write your message here...

Encode and send it

Last received message

The password for my bank account is 123pass321

Decode and read it

6. Table – Search Engine

Write a function that **searches** in a **table** by given input.

Student name	Student email	Student course
John Dan	john@john-dan.com	JS-CORE
Max Peterson	max@softuni.bg	JS-WEB
Philip Anderson	philip@softuni.bg	FRONT-END
Sam Lima	sam@gmail.com	TECH-JS
Eva Longoria	eva@gmail.com	All possible courses
<input type="text"/> <input type="button" value="SEARCH"/>		

When the **"Search" button** is **clicked**, go through all cells in the table except for the first row (Student name, Student email and Student course) and check if the given input has a match (check for both **full words** and **single letters**).

If any of the rows contain the submitted string, add a **select class** to that row. Note that more than one row may contain the given string.

Otherwise, if there is no match, **nothing should happen**.

Note: After every search ("Search" button is clicked), **clear the input field** and **remove all already selected classes** (if any) from the previous search, in order for the **new search** to contain only the **new result**.

For instance, if we try to find **eva**:

Student name	Student email	Student course
John Dan	john@john-dan.com	JS-CORE
Max Peterson	max@softuni.bg	JS-WEB
Philip Anderson	philip@softuni.bg	FRONT-END
Sam Lima	sam@gmail.com	TECH-JS
Eva Longoria	eva@gmail.com	All possible courses
<input type="text" value="eva"/> <input type="button" value="SEARCH"/>		

The result should be:

Student name	Student email	Student course
John Dan	john@john-dan.com	JS-CORE
Max Peterson	max@softuni.bg	JS-WEB
Philip Anderson	philip@softuni.bg	FRONT-END
Sam Lima	sam@gmail.com	TECH-JS
Eva Longoria	eva@gmail.com	All possible courses
<input type="text"/> <input type="button" value="SEARCH"/>		

If we try to find all students who have email addresses in **softuni** domain, the expected result should be:

Student name	Student email	Student course
John Dan	john@john-dan.com	JS-CORE
Max Peterson	max@softuni.bg	JS-WEB
Philip Anderson	philip@softuni.bg	FRONT-END
Sam Lima	sam@gmail.com	TECH-JS
Eva Longoria	eva@gmail.com	All possible courses
<input type="text"/> <input type="button" value="SEARCH"/>		

7. Furniture

You will be given some furniture as an **array of objects**. Each object will have a **name**, a **price** and a **decoration factor**.

When the **"Generate"** button is clicked, add a **new row to the table** for each piece of furniture with **image**, **name**, **price** and **decoration factor** (code example below).

When the **"Buy"** button is clicked, get all **checkboxes that are marked** and show in the **result textbox** the **names** of the piece of furniture that **were checked**, separated by a **comma** and **single space** (", ") in the following format: **"Bought furniture: {furniture1} {furniture2}..."**.

On the next line, print the total price in format: **"Total price: {totalPrice}"** (formatted to the second decimal point). Finally, print the average decoration factor in the format: **"Average decoration factor: {decFactor}"**

Input Example




```
[{"name": "Sofa", "img":  
"https://res.cloudinary.com/maisonsdumonde/image/upload/q_auto,f_auto/w_200/img/  
grey-3-seater-sofa-bed-200-13-0-175521_9.jpg", "price": 150, "decFactor": 1.2}]
```

Examples

Furniture List

```
"name": "Wardrobe",  
"price": "120",  
"decFactor": "1.2"  
}
```

Generate

Image	Name	Price	Decoration factor	Mark
	Office chair	160	0.5	<input type="checkbox"/>
	Sofa	259	0.4	<input checked="" type="checkbox"/>
	Wardrobe	120	1.2	<input checked="" type="checkbox"/>

Bought furniture: Sofa, Wardrobe
Total price: 379.00
Average decoration factor: 0.8

Buy


```

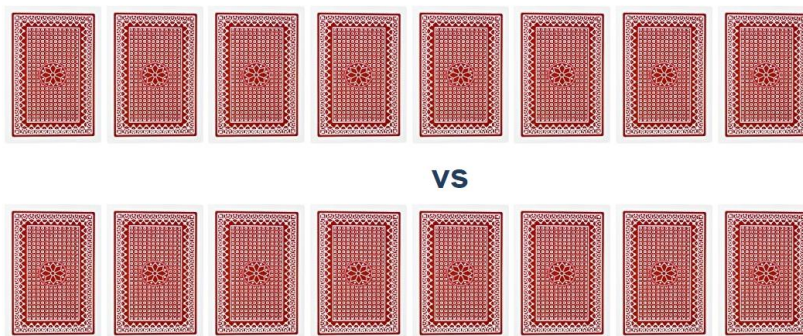
▼<tr>
  ▼<td>
    
  </td>
  ▼<td>
    <p>Sofa</p>
  </td>
  ▼<td>
    <p>259</p>
  </td>
  ▼<td>
    <p>0.4</p>
  </td>
  ▼<td>
    <input type="checkbox">
  </td>
</tr>

```

8. Cards

Write a function which checks cards, shows which one is greater and keeps history of all hands.

Create a functionality which checks all cards, shows which one is greater and keeps history of all hands.



```






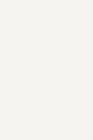


<head>...</head> == $0
<body>
  <section class="description">
    <h2>...</h2>
  </section>
  <section class="cards">
    <div id="player1Div">
      
      
      
      
      
      
      
      
    </div>
    <div id="result">
      <span></span>
      <span>vs</span>
      <span></span>
    </div>
    <div id="player2Div">
      
      
      
      
      
      
      
      
    </div>
    <div id="history">
    </div>
  </div>

```









Firstly, add click events to all cards. When one of the cards is clicked, the current background card must be changed with the "whiteCard.jpg" picture (it is given in the skeleton) and the **card name should be appended** to one of the **span** elements in the **div** with **id="result"**.

If a card **from the top side** is **clicked**, **append** the **card name** to the **left span** (first empty **span**), otherwise **append** the card name to the **right span** (second/last **span**).

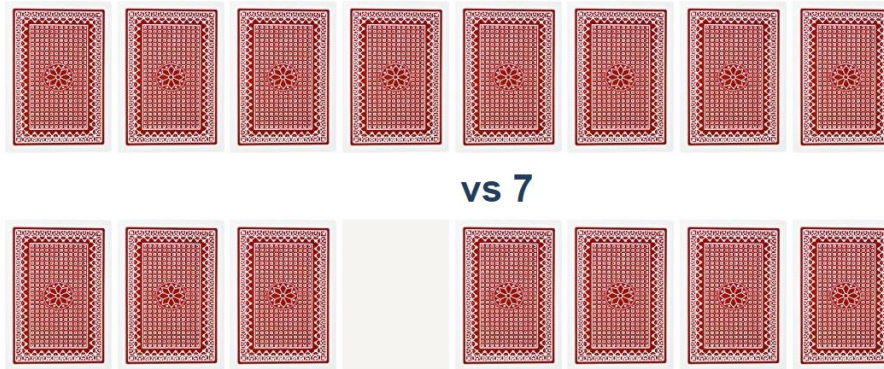
Create a functionality which checks all cards, shows which one is greater and keeps history of all hands.

10 vs

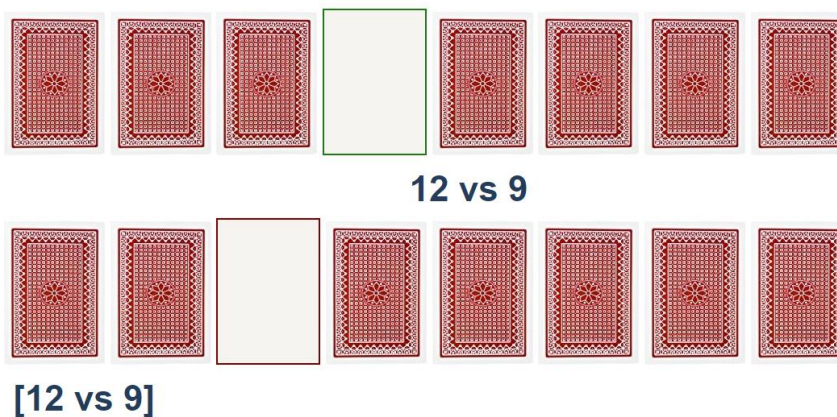









Create a functionality which checks all cards, shows which one is greater and keeps history of all hands.

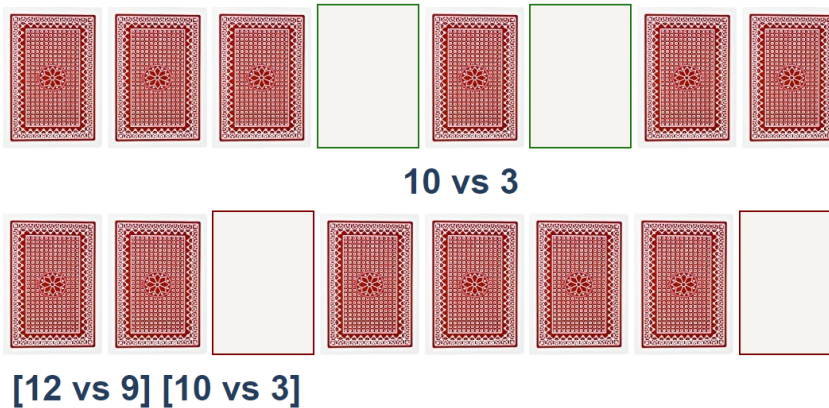


When **cards** from **both sides** are **selected**, check which one is **greater**. The greater card should have border "**2px solid green**" and the lower card - "**2px solid red**".

Create a functionality which checks all cards, shows which one is greater and keeps history of all hands.



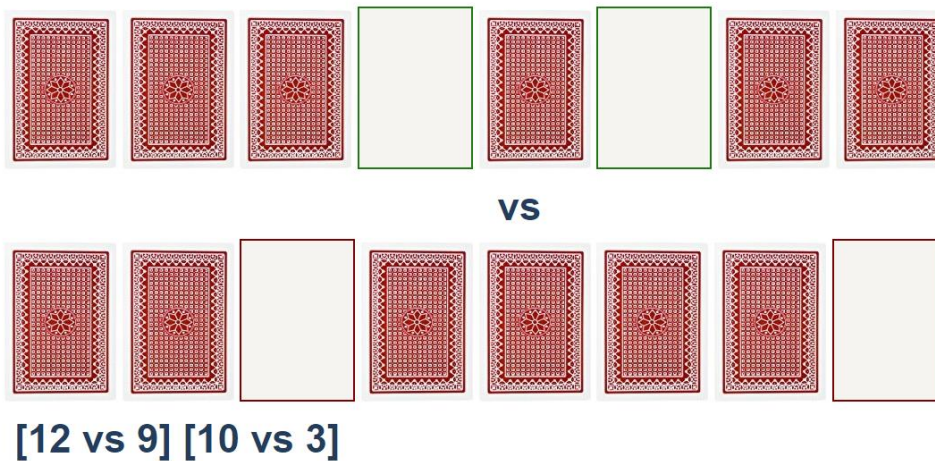
Create a functionality which checks all cards, shows which one is greater and keeps history of all hands.



You should **clear** the **span elements** which **hold the current card names** when both are selected, and the winner is selected. **After every hand**, push the current card names in the **history div** in the following format:

[{top side card name} vs {bottom side card name}]

Create a functionality which checks all cards, shows which one is greater and keeps history of all hands.



9. * Distance Converter

Your task is to convert from **one** distance unit to **another** by adding a **click** event listener to a button. When it is clicked, **read** the value from the input field and **get** the **selected** option from the **input** and **output** units drop downs. Then **calculate** and **display** the converted value in the **disabled** output field.

Example

Distance Converter

From:

Kilometers

▼

CONVERT

To:

Meters

▼

Hints

- Multiply the incoming distance by the following conversion rates to convert to meter
- Divide to convert from meters to the required output unit
- To see which option is selected, read the properties of its parent: **value** gives you the value of the selected option (as displayed in the HTML), **selectedIndex** gives you the 0-based index of the selected option. For example, if miles are selected, **inputUnits.value** is "mi", **inputUnits.selectedIndex** is 4. Option text is irrelevant
- Use the following table information to do that:

1 km	1000 m
1 m	1 m
1 cm	0.01 m
1 mm	0.001 m
1 mi	1609.34 m
1 yrd	0.9144 m
1 ft	0.3048 m
1 in	0.0254 m

10. * Sudomu

Write a function that implements **SUDOMU** (Sudoku inside the DOM).

SUDOMU

The rules are simple and they are **the same** as the **typical sudoku game** (for more information, click [here](#))

If the table is filled with the **right numbers**, and the **"Quick Check"** button is **clicked**, the expected result should be:

SUDOMU

1	2	3
3	1	2
2	3	1

You solve it! Congratulations!

The table borer should be changed to: **"2px solid green"**. The **text content** of the **paragraph** inside the **div** with an **id "check"** must be **"You solve it! Congratulations!"**

The text color of that div must be **green**.

Otherwise, when the filled table **does not solve the sudomu**, the result should be:

SUDOMU

1	2	3
3	1	3
2	3	1

Quick Check

Clear

NOP! You are not done yet...

The table border should be changed to: "2px solid red".

The **text content** of the **paragraph** inside the **div** with an id "**check**" must be:

"NOP! You are not done yet..."

The text color of that div must be **red**!

The "**Clear**" button **clears the whole SUDOMU (removes all numbers)** and the **paragraph which contains the messages. It also removes the table border.**

SUDOMU

Quick Check

Clear