

More Exercises: DOM

1. 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@kingslanduniversity.com	JS-WEB
Philip Anderson	philip@kingslanduniversity.com	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@kingslanduniversity.com	JS-WEB
Philip Anderson	philip@kingslanduniversity.com	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@kingslanduniversity.com	JS-WEB
Philip Anderson	philip@kingslanduniversity.com	FRONT-END
Sam Lima	sam@gmail.com	TECH-JS
Eva Longoria	eva@gmail.com	All possible courses

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

Student name	Student email	Student course
John Dan	john@john-dan.com	JS-CORE
Max Peterson	max@kingslanduniversity.com	JS-WEB
Philip Anderson	philip@kingslanduniversity.com	FRONT-END
Sam Lima	sam@gmail.com	TECH-JS
Eva Longoria	eva@gmail.com	All possible courses

What to submit?

Zip file containing the following:

- solution.js
- template.css
- template.html

File Name: TABLE-SEARCH-ENGINE.zip

2. Shopping Cart

You will be given some products that you should be able to add to your cart. Each product will have a **name**, **picture** and a **price**.

When the **"Add"** button is clicked, append the current product to the **textarea** in the following format: **"Added {name} for {money} to the cart.\n"**.




When the button **"Checkout"** is clicked, calculate the **total money** that you need to pay for the products that are currently in your cart. Append the result to the **textarea** in the following format:

"You bought {list} for {totalPrice}."

The list should contain only the **unique products**, separated by **" , "**. The total price should be rounded to the second decimal point.

Also, after clicking over **"Checkout"** and every from above is done you should **disable all buttons**. (You **can't** add products or checkout again, if once checkout button is clicked)

Examples

Shopping Cart			Price
	Bread Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Add	\$0.80
	Milk Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Add	\$1.09
	Tomatoes Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.	Add	\$0.99
<div>Added Tomatoes for 0.99 to the cart. Added Bread for 0.80 to the cart. Added Bread for 0.80 to the cart. You bought Tomatoes, Bread for 2.59.</div>			
			<div>Checkout</div>

What to submit?

Zip file containing the following:

- solution.js
- template.css
- template.html

File Name: SHOPPING-CART.zip

3. 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>
```

What to submit?

Zip file containing the following:

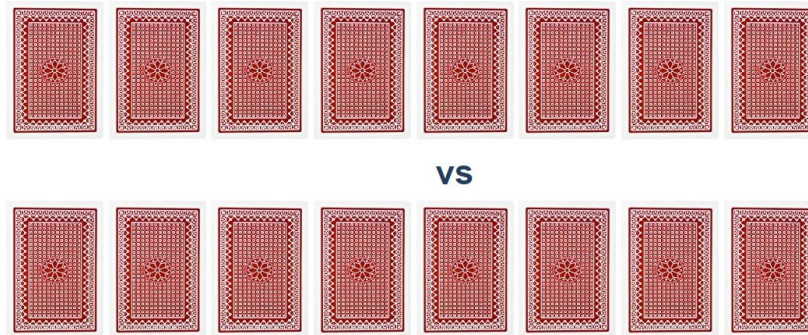
- solution.js
- template.css
- template.html

File Name: FURNITURE.zip

4. 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.

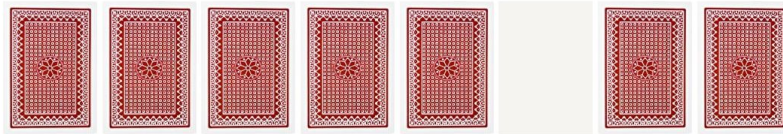


```
...<head>...</head> == $0
▼<body>
  ▼<section class="description">
    ▶<h2>...</h2>
  </section>
  ▼<section class="cards">
    ▼<div id="player1Div">
      
      
      
      
      
      
      
      
    </div>
    ▼<div id="result">
      <span></span>
      <span>vs</span>
      <span>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

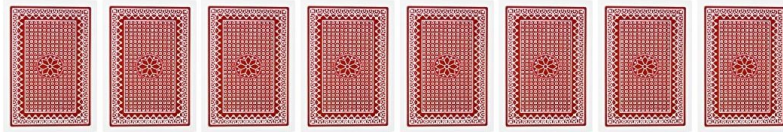
Firstly, add click events to all cards. When one of the cards is clicked, the current background card must be changed with the "whiteCard.jpg" picture (it is given in the skeleton) and the card name should be appended to one of the span elements in the div with id="result".

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).

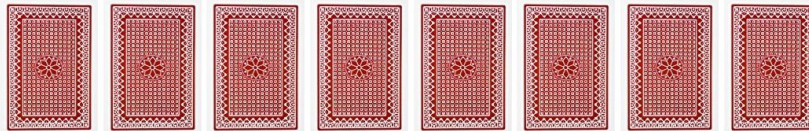
Create a functionality which checks all cards, shows which one is greater and keeps history of all hands.



10 vs



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



vs 7



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.

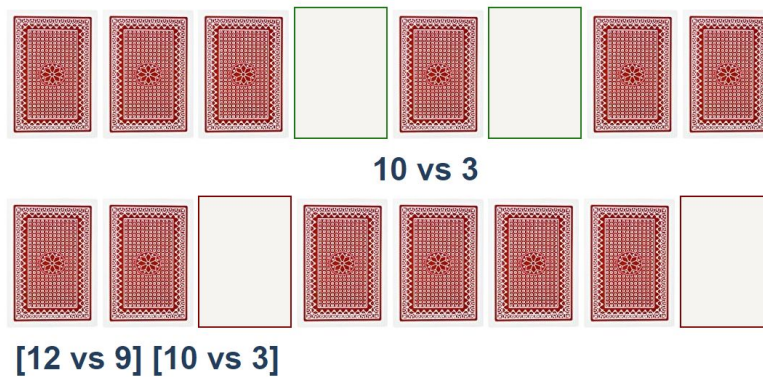


12 vs 9



[12 vs 9]

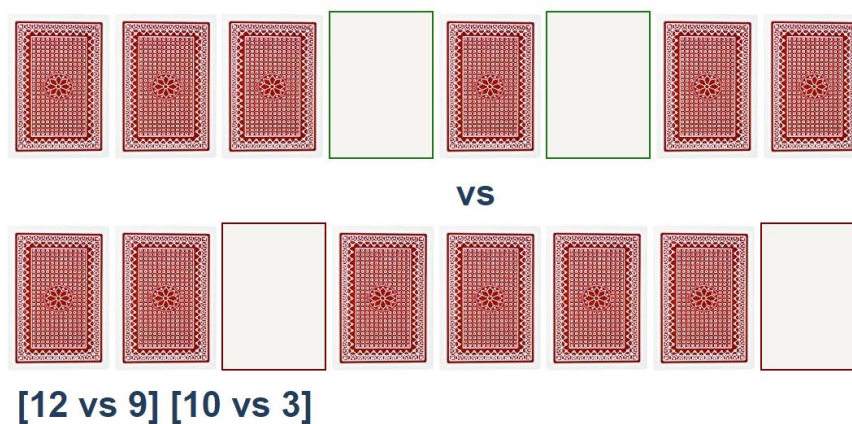
Create a functionality which checks all cards, shows which one is greater and keeps history of all hands.



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.



What to submit?

Zip file containing the following:

- solution.js
- template.css
- template.html

File Name: CARDS.zip

5. Sudomu

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

SUDOMU

<div>Quick Check Clear</div>		

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
<div>Quick Check Clear</div>		

You solve it! Congratulations!

The table border 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
<div>Quick Check Clear</div>		

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

<div>Quick Check Clear</div>		

What to submit?

Zip file containing the following:

- solution.js
- template.css
- Template.html
- 9 x 9 folder

File Name: SUDOMU.zip