Name : Bertan Kerem / Surname : Cüce

Final Project of Introduction to Web Programming

Introduction Part: The aim of the concept is to introduce the original Marvel characters to the reader and to have the reader determine and choose their favorite character accordingly.

Key Sections of My Code:

```
function showDetails(character) {
const characterDetails = {
    ironman: {
        title: "Iron Man",
        description: "<i>The real name of Iron Man is <b>Tony Stark</b>, a genius, billionaire, and master of armor technology.<i>",
        equipment: ["Iron Man Suit", "Arc Reactor", "AI Assistant J.A.R.V.I.S."],
        image: "ironman.jpg",
        wikiLink: "https://en.wikipedia.org/wiki/Iron_Man",
        videoLink: "https://www.youtube.com/embed/nDmia5MjQ0I?si=S_cJ3Wsvg-RLafAy"
    },
    captainamerica: {
        title: "Captain America",
        description: "<i><b>Steve Rogers</b> became a strong and agile hero thanks to the super soldier serum.<i>",
        equipment: ["Vibranium Shield", "Motorcycle", "Tactical Gear"],
        image: "captainamerica.jpg",
        wikiLink: "https://en.wikipedia.org/wiki/Captain_America",
        videoLink: "https://www.youtube.com/embed/swHGr6hJd98?si=t0xLUUnQJkKRzsSs"
    },
    thor: {
        title: "Thor",
        description: "<i>The god of Asgard, <b>Thor son of Odin</b>, wields his hammer Mj□lnir to protect the world from evil.<i>",
        equipment: ["Mj□lnir", "Stormbreaker", "Asgardian Armor"],
        image: "thor.jpg",
        wikiLink: "https://en.wikipedia.org/wiki/Thor_(Marvel_Comics)",
        videoLink: "https://www.youtube.com/embed/0WqBCG_-udw?si=j3HXiG8L264ajfPf"
    },
    hulk: {
        title: "Hulk",
        description: "<i><b>Bruce Banner</b> turns into a green giant due to gamma radiation and his anger.<i>",
        equipment: ["Gamma Strength", "Lab Equipment", "Quinjet"],
        image: "hulk.jpg",
        wikiLink: "https://en.wikipedia.org/wiki/Hulk_(Marvel_Comics)",
        videoLink: "https://www.youtube.com/embed/igndOYw14fc?si=AxBZBzzm1MWQMOp2"
```

```
},
blackwidow: {
    title: "Black Widow",
    description: "<i><b>Natasha Romanoff</b> is a super spy and a master of martial arts.<i>",
    equipment: ["Black Widow's Bite (Electroshock Weapons)", "Grappling Hook", "Stealth Suit"],
    image: "blackwidow.jpg",
    wikiLink: "https://en.wikipedia.org/wiki/Black_Widow_(Natasha_Romanoff)",
    videoLink: "https://www.youtube.com/embed/JyyGJk51n-0?si=shGvwjMOzzHzi_m4"
},
hawkeye: {
    title: "Hawkeye",
    description: "<i><b>Clint Barton</b> is a SHIELD agent who has mastered archery.<i>",
    equipment: ["Bow and Arrows", "Explosive Arrows", "Quiver with Advanced Tech"],
    image: "hawkeye.jpg",
    wikiLink: "https://en.wikipedia.org/wiki/Hawkeye_(Clint_Barton)",
    videoLink: "https://www.youtube.com/embed/IZC9AhvuUko?si=UmPVd3dCrc4kKU8z"
}
```

The provided code defines a function named showDetails that takes a character argument. This function most likely populates a section of the webpage with details about a selected superhero from the Marvel Original Six.

Let's break down the function body:

1. The function defines a constant dictionary named characterDetails.

2. This dictionary stores information about each superhero as key-value pairs, where the key is the superhero's name in lowercase letters.
3. Each value in the dictionary is another dictionary containing details about that particular superhero.
4. The details include:
    - title: The superhero's name
    - description: A short description of the superhero in HTML format
    - equipment: A list of the superhero's equipment
    - image: The filename of the superhero's image
    - wikiLink: A link to the superhero's Wikipedia page
    - videoLink: A link to a YouTube video about the superhero
5. When the showDetails function is called with a superhero's name as an argument (e.g., showDetails('ironman')), it retrieves the corresponding superhero details dictionary from the characterDetails dictionary.
6. The function can then use this information to populate the details section of the webpage with the superhero's name, description, image, equipment list, and links.

```
const mainContent = document.getElementById("mainContent");
const details = document.getElementById("details");
const characterData = characterDetails[character];

if (characterData) {
    const equipmentList = characterData.equipment.map(item => `<li>${item}</li>`).join('');

    details.innerHTML = `
        <img src="${characterData.image}" alt="${characterData.title}" style="width: 100%; max-width: 300px; margin-bottom: 1rem;">
        <h2>${characterData.title}</h2>
        <p>${characterData.description}</p>
        <h3>Favorite Equipment:</h3>
        <ul style="text-align: left; list-style-type: disc; padding-left: 20px;">
            ${equipmentList}
        </ul>
        <h3>Watch the Hero in Action:</h3>
        <iframe width="560" height="315" src="${characterData.videoLink}" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>
        <br>
        <a href="${characterData.wikiLink}" target="_blank" style="color: #f00; text-decoration: none; font-weight: bold;">More Info on Wikipedia about Hero</a>
        <hr>
        <button id="backButton" onclick="goBack()">Go Back</button>
    `;
    mainContent.style.display = "none";
    details.style.display = "block";
```

Sure, this part of the code fills in the details section of the webpage with information about the selected superhero and hides the character grid section.

Let's break it down step by step:

1. **Retrieving elements:**
    - The code first retrieves two HTML elements from the webpage using their IDs:
        - mainContent: This likely refers to the element containing the character grid section.
        - details: This likely refers to the element containing the details section.

- It also retrieves the character data from the characterDetails dictionary using the character argument passed to the showDetails function.

2. **Checking for character data:**
    - An if statement checks if the characterData is truthy (not null or undefined). This ensures the code only proceeds if a valid superhero was selected.
3. **Building the equipment list:**
    - If characterData exists, the code uses the map function to iterate over the equipment list within characterData.
    - Inside the map function, each equipment item is wrapped in an <li> (list item) HTML tag.
    - The join('') method combines all the <li> items into a single string separated by empty strings.
4. **Populating the details section:**
    - The innerHTML property of the details element is set to a large string containing HTML code. This code defines the structure and content of the details section.
    - The string uses template literals with embedded expressions to dynamically insert the superhero's details from characterData:
        - An image with source, alt text, and styles.
        - The superhero's title in an <h2> tag.
        - The superhero's description in a <p> tag.
        - An <h3> tag for "Favorite Equipment".
        - An unordered list (<ul>) to display the equipment list generated earlier using the map function.
        - An <h3> tag for "Watch the Hero in Action".
        - An <iframe> element to embed a YouTube video from the videoLink.
        - A link to the superhero's Wikipedia page with styles.
        - A horizontal line (<hr>) for separation.
        - A "Go Back" button with an onclick event handler that calls the goBack function (likely to hide the details and show the character grid again).
5. **Showing/hiding sections:**
    - After populating the details section, the code sets the display style of the mainContent element (character grid) to "none" (hidden).
    - It then sets the display style of the details element (details section) to "block" (visible).

In summary, this code snippet dynamically generates the HTML content for the details section based on the selected superhero's information and displays it, while hiding the character grid section.

```
function goBack() {
    const mainContent = document.getElementById("mainContent");
    const details = document.getElementById("details");

    details.style.display = "none";
    mainContent.style.display = "grid";
```

Certainly, the provided code defines a function named goBack that likely hides the superhero details section and shows the character grid section again.

Here's a breakdown of the code:

1. **Retrieving elements:**
    - Similar to the showDetails function, this code retrieves references to the mainContent and details elements using their IDs.
2. **Hiding details section:**
    - The details.style.display property is set to "none". This hides the element containing the superhero details section.
3. **Showing character grid:**
    - The mainContent.style.display property is set to "grid". This likely shows the element containing the character grid layout.

In essence, this function serves as a way to navigate back from the superhero details section to the main character grid section of the webpage.

```
<main id="mainContent">
    <section class="character" onclick="showDetails('ironman')">
        <img src="ironman.jpg" alt="Iron Man">
        <h2>Iron Man</h2>
    </section>

    <section class="character" onclick="showDetails('captainamerica')">
        <img src="captainamerica.jpg" alt="Captain America">
        <h2>Captain America</h2>
    </section>

    <section class="character" onclick="showDetails('thor')">
        <img src="thor.jpg" alt="Thor">
        <h2>Thor</h2>
    </section>

    <section class="character" onclick="showDetails('hulk')">
        <img src="hulk.jpg" alt="Hulk">
        <h2>Hulk</h2>
    </section>

    <section class="character" onclick="showDetails('blackwidow')">
        <img src="blackwidow.jpg" alt="Black Widow">
        <h2>Black Widow</h2>
    </section>

    <section class="character" onclick="showDetails('hawkeye')">
        <img src="hawkeye.jpg" alt="Hawkeye">
        <h2>Hawkeye</h2>
    </section>
</main>
```

Certainly, let's break down this part of the HTML code:

1. Main Content Section (<main id="mainContent">):

- This tag defines the main content area of the webpage.
- The id="mainContent" attribute gives this section a unique identifier, which is used by JavaScript to access and manipulate it (like hiding it when the details section is shown).

2. Character Sections (<section class="character">):

- This code creates six separate sections, each representing one of the Marvel Original Six superheroes.
- The class="character" attribute assigns a common class to all these sections, which can be used to style them similarly (e.g., background color, border).

3. Image and Title:

- Inside each section, there's:
    - An <img> tag: This displays an image of the superhero.
        - The src attribute specifies the path to the image file (e.g., "ironman.jpg").
        - The alt attribute provides alternative text for screen readers and browsers that cannot display images.
    - An <h2> tag: This displays the superhero's name as a heading.

4. onclick Event Handler:

- The most crucial part is the onclick="showDetails('ironman')" attribute (and similar attributes for other superheroes).
- This attribute defines an event handler that calls the showDetails function when the user clicks on the section.
- The argument passed to showDetails (e.g., 'ironman') is the lowercase name of the superhero, which is used by the function to retrieve the corresponding details from the characterDetails dictionary.

In essence, this part of the code:

- Creates a visual grid of the six superheroes.
- Displays an image and name for each superhero.
- Makes each superhero "clickable" by attaching the onclick event handler.
- When clicked, the showDetails function is triggered, leading to the display of detailed information about the selected superhero.

This structure effectively presents the superheroes in a visually appealing and interactive way, allowing users to easily explore each character and learn more about them.

```html
<form action="#" method="post">
  <h1>Choose Your Favorite Character:</h1>
  <table class="character-select-table">
    <thead>
      <tr>
        <th>Character</th>
        <th>Image</th>
        <th>Selection</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Iron Man</td>
        <td><img src="ironman.jpg" alt="Iron Man" style="width: 80px; height: auto;"></td>
        <td><button type="submit" name="Character" value="Iron Man" class="select-button">Select</button></td>
      </tr>
      <tr>
        <td>Captain America</td>
        <td><img src="captainamerica.jpg" alt="Captain America" style="width: 80px; height: auto;"></td>
        <td><button type="submit" name="Character" value="Captain America" class="select-button">Select</button></td>
      </tr>
      <tr>
        <td>Thor</td>
        <td><img src="thor.jpg" alt="Thor" style="width: 80px; height: auto;"></td>
        <td><button type="submit" name="Character" value="Thor" class="select-button">Select</button></td>
      </tr>
      <tr>
        <td>Hulk</td>
        <td><img src="hulk.jpg" alt="Hulk" style="width: 80px; height: auto;"></td>
        <td><button type="submit" name="Character" value="Hulk" class="select-button">Select</button></td>
      </tr>
      <tr>
        <td>Black Widow</td>
        <td><img src="blackwidow.jpg" alt="Black Widow" style="width: 80px; height: auto;"></td>
        <td><button type="submit" name="Character" value="Black Widow" class="select-button">Select</button></td>
      </tr>
      <tr>
        <td>Hawkeye</td>
        <td><img src="hawkeye.jpg" alt="Hawkeye" style="width: 80px; height: auto;"></td>
        <td><button type="submit" name="Character" value="Hawkeye" class="select-button">Select</button></td>
      </tr>
    </tbody>
  </table>
</form>
```

Certainly, let's break down this HTML code snippet:

1. Form Element (<form>)

- **action="#":** This attribute specifies the URL where the form data should be submitted. In this case, # means the data will be sent to the current page.
- **method="post":** This attribute specifies the HTTP method used to send the form data. "POST" is generally used for sending larger amounts of data or sensitive information.

2. Form Title (<h1>)

- This heading provides a clear label for the form: "Choose Your Favorite Character:".

3. Character Selection Table (<table>)

- **class="character-select-table":** This class is used to style the table with specific CSS rules.
- **Table Header (<thead>)**
    - Contains a single row (<tr>) with three table headers (<th>):
        - "Character"

        - "Image"
        - "Selection"

- **Table Body (<tbody>)**
    - Contains six rows, one for each superhero:
        - **Character Name:** Each row starts with a table data cell (<td>) displaying the superhero's name (e.g., "Iron Man").
        - **Image:** The next cell contains an <img> tag:
            - src: Specifies the path to the image file.
            - alt: Provides alternative text for screen readers.
            - style: Defines the image dimensions (width: 80px, height: auto).
        - **Selection:** The final cell contains a <button> element:
            - type="submit": This attribute indicates that clicking the button submits the form.
            - name="Character": This attribute gives the form data a name, which can be used to access the selected character on the server-side.
            - value="Iron Man": This attribute sets the value of the submitted data for this specific button.
            - class="select-button": This class can be used to style the button with CSS.

In summary:

This code creates an interactive form that allows users to select their favorite Marvel Original Six character. When a user clicks the "Select" button for a character, the form is submitted, and the server-side script can then process the selected character's name (e.g., "Iron Man") to perform actions like:

- Displaying more information about the selected character.
- Storing the user's preference in a database.
- Triggering other actions based on the user's choice.