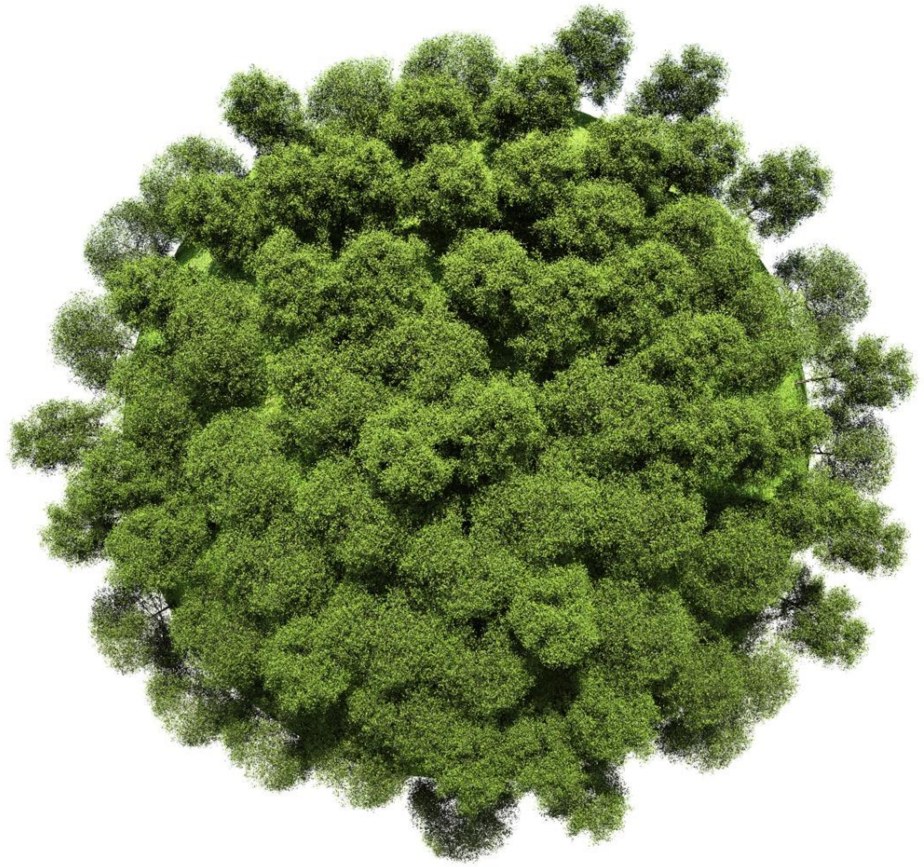
Deloitte.
Digital

OCTOBER 23, 2023

Lightning Development Basics



📅 [LWC Week3] – Day 1

⬇️Download the git folder with the challenges Week 3 - Challenges

1. JavaScript mini challenges (30mins)
2. Astronaut challenge (1h)
3. Set up the environment (30mins)
    a. trailhead: https://trailhead.salesforce.com/content/learn/projects/quick-start-lightning-web-components (parts 1 and 2)
    b. https://developer.salesforce.com/blogs/2023/05/developer-tooling-from-scratch-part-1-of-2
4. Dice Game challenge (1h)
5. Challenge solutions and discussion (1h)

JavaScript mini challenges (1/5)

Exercise 1: hoisting

1. assign in a variable ‘myString’ a value without initializing the variable and log the result
2. repeat the step 1 using 'use strict' and with 'use strict' commented out
3. initialize a variable with var keyword and console.log the variable before its initialization
4. in a line before logging the variable try to assign a value to it
5. repeat the steps 3 and 4 but declare the variable with let and const
6. What do you observe?

JavaScript mini challenges (2/5)

Exercise 2: Variable mutation and comparison

1. Declare two variables num with value 7 and str with value '3'
2. The point of this exercise is to understand the type coercion of the variables:
   a. use the typeof keyword to log the type (log also the value using console.log(value, typeof)) of
      i. the variables num , str
      ii. the variable str + num
      iii. the variable str * num
      iv. the variable str*1
      v. the variable +str
      vi. the variable ''+num
      vii. the variable String(num)
      viii. the variable Boolean(num)

   b. What are the Falsy and Truthy values?
      i. try to log with the boolean constructor the values: 0, '', null, undefined, NaN
      ii. try to log with the boolean constructor the values: 1, ' ', {}, [], Infinity

JavaScript mini challenges (3/5)

2.c. In the following code try first to write as a comment in the end of each line the expected outcome (True or False) then run the code to your js file and think about the errors in the expected and actual values

```
console.log('comparison')
// https://developer.mozilla.org/en-
US/docs/Web/JavaScript/Equality_comparisons_and_sameness
console.log('0 == false: ', 0 == false)
console.log('1 == true: ', 1 == true)
console.log('1 === true: ', 1 === true)
console.log('1 == "1": ', 1 == '1')
console.log('1 === "1": ', 1 === '1')
console.log('null == undefined: ', null == undefined)
console.log('null == false: ', null == false)
console.log('NaN === NaN: ', NaN === NaN)
console.log('!null: ', !null)
console.log('!!null: ', !!null)
console.log('!NaN: ', !NaN)
console.log('!!NaN: ', !!NaN)
```

JavaScript mini challenges (4/5)

Exercise 3: Objects and this keyword

1. Create a person object with attributes firstName, lastName, birthYear and a printInfo function (leave blank for now)
2. log person’s firstName
3. log person’s lastName but using something like person['last’ + 'Name']
4. get the date today and store to a variable the current year
5. in the print info function try to return a string like that
   'Full Name: yourFullName, Age: yourAge'
6. log the result of the printInfo function
7. Use template literal to return the info (step 5)
8. copy the person object code and create a person2 object with only difference instead of a named function use an arrow function
9. log the function results, what do you observe?

JavaScript mini challenges (5/5)

Exercise 4: Arrays

Given the array [4, 5, -1, 6, 0, 10, 3]

1. calculate the min of the array using for loop or Array.prototype.forEach
2. calculate the max of the array using for loop or Array.prototype.forEach
3. calculate the sum of the elements using for loop, Array.prototype.forEach or Array.prototype.reduce

Astronaut instructions – part 1 (1/2)

🗁index.html file

1. link the other two style sheets existing in the astronaut challenge folder
2. Add a h1 element with the title "How Many people are in space 🌌?"
3. Add a button element with the "btn" class and label "Calculate"
4. Link the 'scriptWithAsync.js' file using <script> tag

🗁 style.css file

5. add a class name hidden which will modify the display attribute to none

🗁 scriptWithAsync.js file

6. Add a variable to store the button element added on the html,
    you can use getElementById method of the document element to access the button
7. Call init function
8. Use addEventListener method and a callback function in the button element
   created in step 6.
    a. The callback function must me asynchronous (async keyword must be used)

Astronaut instructions – part 1 (2/4)

```
    8.b. the callback function will call and await the fetchData function
    8.c. The callback function must remove the hidden class added to the outcome element in
         the init function.
9.  a. Use the fetch API as shown in the documentation to fetch the data from apiUrl.
    b. store the respose in a variable called response (as shown in the step 9.a link)
    c. store the response json in a variable called data (as shown in the step 9.a link)
    d. console.log the data variable and see the results of the API call
    Hint: the data object should be like this:
       data = { people: [array], number: num, message: 'success’ }
10. Store in the numberOfPeople variable the data attribute that has the number of people
11. a. For each item of the peopleArray create a temp line like this:
              '<p>&nbsp👦name, 🚀spacecraftName</p>'
            b. Add the line to moreInfoOutput string
            c. print the moreInfoOutput in the '.outcome__people__info' element
12. Print the moreInfoOutput in the '.outcome__people__info' element
```

Astronaut instructions – part 2

13. Copy the scriptWithAsync.js and paste it in the same folder but rename it 'scriptWithThen.js'
14. Remove the async...await of the functions along with the result and data variable assignment inside the fetchData
15. Use the

```
    fetchPromise.then(response => get response json)
            .then(data => {
                    move all the logic from console.log(data)
                    until the end of try block
            })
```

    help: https://www.developerway.com/posts/fetching-in-react-lost-promises

    You can keep the try...catch or replace it with .catch((err) => {handle error})
16. In the html replace the src of the script tag to point to 'scriptWithThen.js'
17. The results are rendering twice (once with the initial values and once with the correct values) can you explain why?

Astronaut instructions – part 3

18. Switch back to 'scriptWithAsync.js'
19. Replace the Calculate text inside the button block with this:

```
<span class="btn__label">calculate</span>
<span class="loader">
    <span class="loader__element"></span>
    <span class="loader__element"></span>
    <span class="loader__element"></span>
</span>
```

20. In the js file add in the beggining two more variables,
    one to select the '.loader' and one to select the '.btn__label' elements.
21. In the init the style.display attribute to hide the '.loader' element
    (it's the same logic as the hidden class)
22. In the calculate callback function we should have the logic

```
hide btn label (add hidden class)
show loading dots (style.display = 'flex')
***** fetching data ******
hide loading dots (you did the same in step4)
show button label (remove hidden)
```

Astronaut instructions – part 4

23. (html) Inside the outcome div add after the last <p> element

```
    <div>
        <a href="" class="showLess" target="">>> Hide Results</a>
    </div>
```

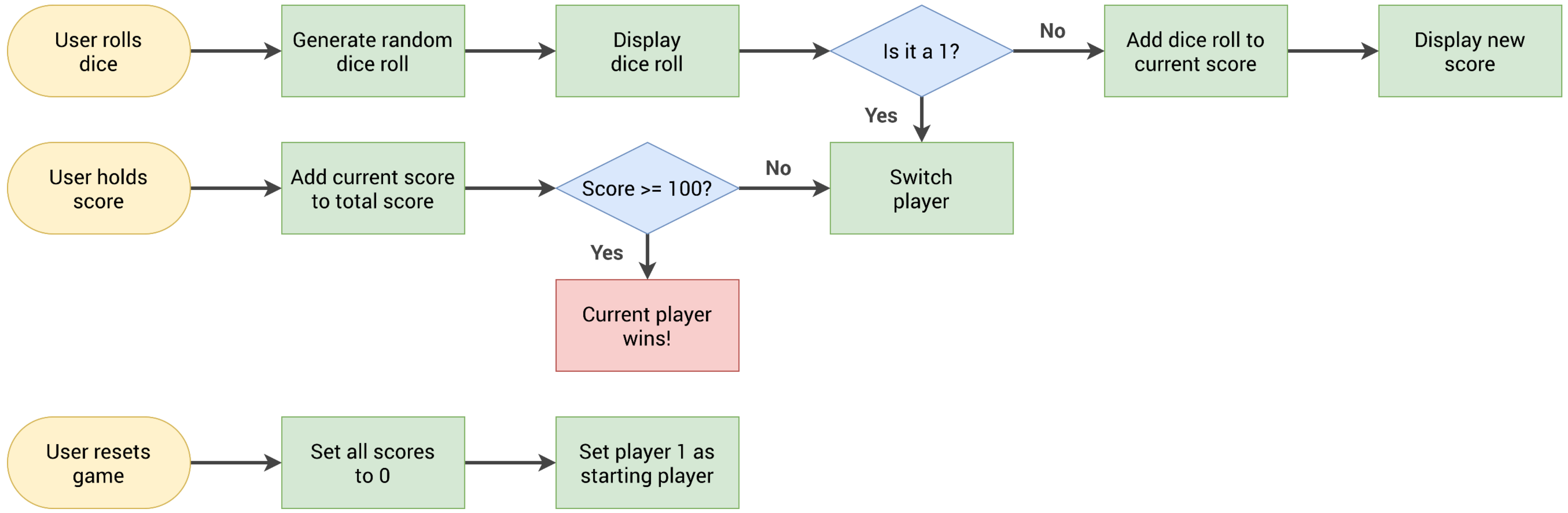
24. (.js) Add a variable and save the '.showLess' element

25. Add an click event listener to the element and in the callback function add hidden to the outcome element

****

Congratulations you finished the challenge!!!!🎉🎉💫✨💥

Dice Game flow chart



Dice Game instructions (1/2)

- *talk about variables and onclick events in LWC framework*

diceGameChallenge.html file

#1. Set the score variable scoreP2 for P2. (html - l19)

#2. Set the current score variable currentP2 for P2. (html - l23)

#3. Connect the handleRollBtnClicked and handleHoldBtnClicked functions to the appropriates button elements

diceGameChallenge.js file

```
#4. Initialize the variables scoreP1, scoreP2, currentP1, currentP2 to  0.
#5. Remove the player--winner class from player1 and plaeyer2 elements
#6. Create the basic algorith of the user roll dice
    a. if diceNum is not equal to 1
       i. add to the current score (currentScore variable) the diceNum.
      ii. assign the currentScore variable to the current score of the active player
          Help: if this activePlayer is P1 assign the current score to currentP1 variable.
                if this activePlayer is P2 asign the current score to currentP2 variable.
     (Hint: currentP1 = current + P1 and activePlayer is either P1 or P2 strings 💣)
```

Dice Game instructions (2/3)

```
    6.b. if diceNum is equal to 1 according yo the flow chart we must switch the player so
         the switchPlayer function must be called.
#7. Complete the user holds score algorithm:
    a. Add the current score to the score of the active player.
Hint: For example if the active player is P1 add the currentScore to the scoreP1 variable
    b. If the score of the active player is greater or equal than 100
          i. set the playing variable to false
         ii. select the game element and add hidden class
        iii. select the active player element and add the
             'player--winner' class and remove the 'player--active' class.
         iv. select the name+ActivePlayer--winner element and remove the hidden class
    c. if the score of the active player is lower than 100,
       the switchPlayer function must be called.
```

Dice Game instructions (3/3)

#8. Complete the switch player logic:

a. set the current+Active player score to 0

b. set the currentScore to zero

c. we must switch the activePlayer!!

i. if the current activePlayer is 'P1' the activePlayer must be set to 'P2'

ii. if the current activePlayer is 'P2' the activePlayer must be set to 'P1'

- *Talk about how to improve the code using*
  - template literals
  - ternary operator

📅 [LWC Week3] – Day 2

1. Pokedex Challenge (~1.5h)
2. Final Exercise

Pokedex instructions – html (1/4)

```
#1. Create a lightning-card with narrow variant,
        title: Pokedex, icon: standard:contract_line_item
#2. In the actions slots of the card put a button with label: Reset Data
    and onclick bind a method with name "initializeApp". You can style your button red
    using the right variant!!
#3. In the javascript file make sure that the method initializeApp () {}
    exists if not add the method.
#4. After the button put a div with class="slds-m-around_small",
    inside the div will be the card-body
#5. Inside the card body put
    a. A div with inline style="display:flex;flex-direction:row;
       justify-content:flex-start;align-items:flex-end; margin-bottom: 1rem;"
       This will be the search div.
    b. A div with class="slds-grid slds-wrap slds-m-around_small slds-border_bottom"
       this will be the results header
```

Pokedex instructions – html (2/4)

```
 5.c. A similar div as in 5.b without border and with margin around small this will be the
results data
        Inside the <template></template> your structure must be similar to the following
          <card__div>
           <reset__button></reset__button>
           <card-body__div>
             <search__div>
             </search__div>
             <results-header__div>
             </results-header__div>
             <results-data__div>
             </results-data__div>
           </card-body__div>
         </card div>
#6. [search__div] Inside the search__div put a lightning input with:
                  a. label 'Pokemon Name'
                  b. value: pokemonName (variable initialized to 'charizard')
                  c. onchange: handleChange (handler method, parameter event)
                  d. class="slds-m-right_xx-small"
```

Pokedex instructions – html (3/4)

```
#7. [search__div] Inside the search__div, after the input put a lightning button with:
    a. label 'Search Pokemon'
    b. variant: "brand"
    c. onclick: searchPokemon (handler method, parameter event)
    d. disabled: disableSearch (variable initialized to false)
#8. [.js] In your javascript file make sure that your variables and
    handler methods exist if not add them.
#9. [results-header__div] Inside the results-header__div add 4 columns with the same space
    using the instructions of the slds-grid each one of them must have additional
      • class slds-text-heading_small and
      • the labels of this elements should be: Name, Appearance, Abilities, Stats.
#10. [results-data__div] we want to show this div only if data exist:
    a. bind the div with an if directive and show it only if the hasData variable
       is true initialize the variable to false.
    b. put the following div after the results-data__div closing tab
            <div lwc:else class="slds-m-around_large">
               {errorMessage}
            </div>
```

Pokedex instructions – html (4/4)

#11. Inside the results data div put the following code:

```
 <div class="slds-size_1-of-4 slds-align-middle">
  print pokemon's name
</div>
<div class="slds-size_1-of-4">
  <img src={currentPokemon.imgUrl}>
</div>
<div class="slds-size_1-of-4 slds-text-color_weak slds-align-middle">
  print a list and each item will represent an ability
</div>
<div class="slds-size_1-of-4 slds-text-color_weak slds-align-middle">
  <ul>
    <template for:each={currentPokemon.stats} for:item="stat">
      <li key={stat}>{stat.name}: {stat.value}</li>
    </template>
  </ul>
</div>
```

#12. Your html is almost done.... Put a lightning spinner inside the card-body__div and before anything else.
The spinner will take a variable spinner initialized to false.
We will use it to inform the user that something is loading......

Final Exercise – Addressing the problem

The Service Department is responsible for handling defects on the products purchased by the customers. As of today, the only means of communication with their clients is through phone calls. Once they receive a call they have to find the account and search for the product he is referring to. The whole process is then tracked offline.

The head of the department has asked you for a solution that will allow tracking the clients' requests and efficiently track the process of the resolution. Additionally, he asked if it is possible to facilitate the searching of the products purchased by each customer, so that the service employees can be faster.

After an initial analysis, the proposed solution was to use the Case object to track each defect.

(Exercise preparation)

Create a lookup from the Case object to the OpportunityProduct__c object.

Final Exercise – Part 1

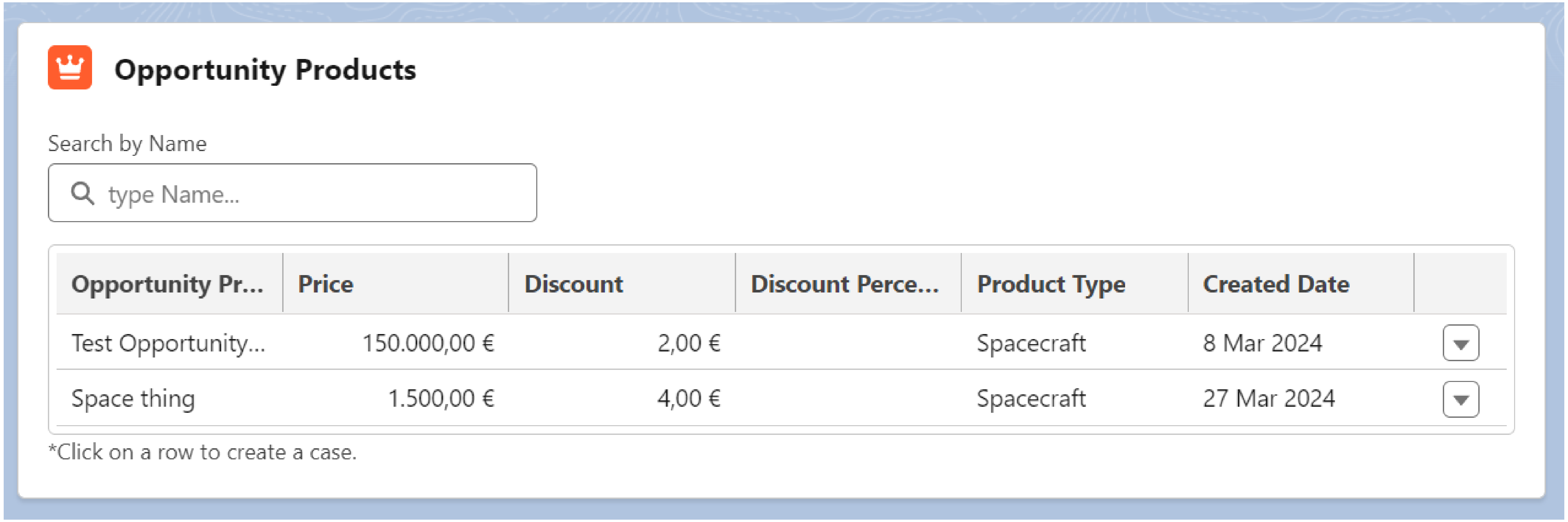
To facilitate the searching of the customers’ products a custom solution is necessary. Create a custom component that will be added to the Account record page to show all the account’s opportunity products.

The opportunity products must be shown in a table with the following fields:

1. Product name
2. Price (sortable)
3. Discount
4. Discount Percentage
5. Product Type (sortable)
6. Created Date

Above the table we need to add a text input that will be used to search in the table and filter the products based on the product name.

Final Exercise – Part 1 - mockup



Opportunity Products

Search by Name

type Name...

Opportunity Pr...	Price	Discount	Discount Perce...	Product Type	Created Date	
Test Opportunity...	150.000,00 €	2,00 €		Spacecraft	8 Mar 2024	▼
Space thing	1.500,00 €	4,00 €		Spacecraft	27 Mar 2024	▼

*Click on a row to create a case.

📅 [LWC Week3] – Day 3

```
Final Exercise
```

Final Exercise – Part 2

To facilitate the handling of the phone tickets, add the possibility to select a row on the products table. When a row is selected a modal window should open. In the modal window there should be the possibility to create a Case.

The fields necessary for the case creation are:

1. Subject (Required)
2. Description
3. AccountId (automatically populated and disabled)
4. OpportunityProduct__c (automatically populated and disabled)

Final Exercise – Part 3

The department head liked your proposal a lot and asked for a small enhancement on the account page. He would like to have a component that shows an icon based on the account score.

Below the icon, display the number of open cases on the account. When a case is opened from the table, reflect the change on this new component.

Hint

Communicate Between Unrelated Components | Salesforce Trailhead

📆 [LWC Week3] – Day 4

1. Hands on challenge to interact with all 3 (Aura & Visualforce) (2hr)
2. Hands on challenge to interact with all 3 (LWC part) (2hr)

Useful links

1. LWC Udemy course
1. LWC Recipes
2. Sample Gallery
3. Apex Recipes
4. Trailhead live sessions overview (github redirect)
5. Course: The Complete JavaScript Course 2024: From Zero to Expert! | Udemy Business : Section 8 and Section 10 | 138

Deloitte.
Digital

Thank you.

Deloitte.
Digital