Web Dev Bootcamp 2024/25

JavaScript - Lab 07

16.11.2024

Sponsored by:

Agenda

1. Form validation (continued)
2. JS Objects
3. JS Arrays
4. JS Functions dan Recursions
5. Prep & To-Do untuk next session

Built-in Form Validation: JavaScript

- Menggunakan conditions (`if … else …`)
- Menggunakan Constraint Validation API
  - Bisa digunakan untuk beberapa HTML elements (e.g. `<buttons>`, `<input>`, `<textarea>`, dll.)
  - Ekspansi dari built-in HTML form validations (langsung dipanggil di element HTML nya)
  - Customize warning messages/toasts

HTML

```
<form>
  <label for="mail">
    I would like you to provide me with an email address:
  </label>
  <input type="email" id="mail" name="mail" />
  <button>Submit</button>
</form>
```

Source: https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation#validating_forms_using_javascript

JS

```
const email = document.getElementById("mail");

email.addEventListener("input", (event) => {
  if (email.validity.typeMismatch) {
    email.setCustomValidity("I am expecting an email address!");
  } else {
    email.setCustomValidity("");
  }
});
```

Regex Form Validation di JavaScript

Regular Expression adalah format pattern yang bisa digunakan sebagai constraint untuk suatu input teks/String. Coba-coba Regex bisa di **regex101.com**

HTML

```
<p>
  Enter your phone number (with area code) and then click "Check".
  <br />
  The expected format is like ###-###-####.
</p>
<form id="form">
  <input id="phone" />
  <button type="submit">Check</button>
</form>
<p id="output"></p>
```

Source: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions

JS

```
const form = document.querySelector("#form");
const input = document.querySelector("#phone");
const output = document.querySelector("#output");

const re = /^(?:\d{3}|\(\d{3}\))([-/.])\d{3}\1\d{4}$/;

function testInfo(phoneInput) {
  const ok = re.exec(phoneInput.value);

  output.textContent = ok
    ? `Thanks, your phone number is ${ok[0]}`
    : `${phoneInput.value} isn't a phone number with area code!`;
}

form.addEventListener("submit", (event) => {
  event.preventDefault();
  testInfo(input);
});
```

JavaScript Objects

```
// Object yang berisikan properties
let firstCat = {
       name = “Nanuk”,
       breed = “European short hair”,
       age = 8
};


console.log(firstCat.name);             // Hasilnya adalah “Nanuk”
console.log(firstCat.breed);            // Hasilnya adalah “European short hair”

firstCat.favoriteFood = “Everything”;   // Menambahkan property baru (beserta valuenya) dalam object firstCat
```

- Objek di JavaScript, sama seperti Variabel, bisa tidak memiliki isi atau ‘empty’.
- Sama seperti di HTML, dan beberapa programming language lain yang memiliki sistem **‘inheritance’** atau ‘parent-child’, JavaScript menggunakan Objek sebagai implementasi ‘inheritance’.
- Setiap Objek di JavaScript memiliki native/built-in property, ini disebut sebagai **‘Prototype’**.
- More on this:
  - JavaScript Object
  - JavaScript Object Prototypes

JavaScript Arrays

```
// Array
let myCats = [
      “Balu”,
      “Nanuk”
];

console.log(firstCat[breed]);          // Hasilnya adalah “European short hair”
console.log(myCats[0]);                // Hasilnya adalah “Balu”

myCats.push(“Pia”);                    // Hasilnya adalah myCats = [“Balu”, “Nanuk”, “Pia”]
```

- Array adalah tipe data yang merepresentasikan sebuah list atau tabel.
- Kalau urutan/sortiran suatu data merupakan hal yang penting bagi suatu sistem, maka sebaiknya data tersebut dideklarasikan sebagai Array.
- Array bisa dideklarasikan sebagai kosong/empty, dan bisa juga memiliki satu atau beberapa **items**.
- Items dalam suatu Array memiliki satu tipe data, contohnya satu array yang berisikan string items, atau satu array yang berisikan object items.
- More on this: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Arrays

JavaScript Functions & Recursions

```
let catAge = 0;

function catBirthday(x) {
      return x++;
};

catBirthday(catAge);
console.log(catAge);                    // Hasilnya adalah 1
```

```
const catBirthday = x => x++;           // Sama seperti di atas, tapi di sini menggunakan arrow function
```

- Recursion dalam suatu Function artinya adalah Function itu bisa memanggil/call dirinya sendiri. Bayangkan the infinity symbol: ∞
- Jadi Function ini akan terus-menerus jalan (looping), sampai akhirnya memenuhi condition atau iteration yang diberikan.
- More on this:
  - JavaScript Function & Recursion
  - Recursion

JS

```
const factorial = (n) => {
  if (n === 0) {
    return 1;
  } else {
    return n * factorial(n - 1);
  }
};
console.log(factorial(10));
// 3628800
```

To-Do untuk next session

4. When you click on the `#convert-btn` element without entering a value into the `#number` element, the `#output` element should contain the text `"Please enter a valid number"`.

5. When the `#number` element contains the number `-1` and the `#convert-btn` element is clicked, the `#output` element should contain the text `"Please enter a number greater than or equal to 1"`.

6. When the `#number` element contains the number `4000` or greater and the `#convert-btn` element is clicked, the `#output` element should contain the text `"Please enter a number less than or equal to 3999"`.

12. When the `#number` element contains a random negative number and the `#convert-btn` element is clicked, the `#output` element should contain the text `"Please enter a number greater than or equal to 1"`.

13. When the `#number` element contains a number greater than 4000 and the `#convert-btn` element is clicked, the `#output` element should contain the text `"Please enter a number less than or equal to 3999"`.

Untuk menyelesaikan tasks no. 4, 5, 6, 12, dan 13, bisa menggunakan:

- Built-in HTML & JS form validation, e.g. `validity.rangeUnderflow`, `patternMismatch`
- Membandingkan input (integer) secara langsung, e.g. `if (numberInput.value < 1) { … }` (Link)
- Membandingkan input (string) menggunakan regex, e.g. `if (stringInput.match(/[e.]/g) { … }` (Link)
- Customize alert message dengan Constraint Validation API, e.g. setCustomValidity

To-Do untuk next session

7. When the `#number` element contains the number `9` and the `#convert-btn` element is clicked, the `#output` element should contain the text `"IX"`.

8. When the `#number` element contains the number `16` and the `#convert-btn` element is clicked, the `#output` element should contain the text `"XVI"`.

9. When the `#number` element contains the number `649` and the `#convert-btn` element is clicked, the `#output` element should contain the text `"DCXLIX"`.

10. When the `#number` element contains the number `1023` and the `#convert-btn` element is clicked, the `#output` element should contain the text `"MXXIII"`.

11. When the `#number` element contains the number `3999` and the `#convert-btn` element is clicked, the `#output` element should contain the text `"MMMCMXCIX"`.

Untuk menyelesaikan tasks no. 7, 8, 9, 10, dan 11, bisa menggunakan:

- JavaScript Object + `if` statements
- JavaScript Array Map (Array dengan key:value pair) + recursive function/loop function, e.g. `forEach()` method (Link, Link)

To-Do untuk next session

08	**JavaScript Algo. & Data Structures**	• JavaScript Objects • Arrays • Functions and Recursions	Sabtu, **16 Nov. 2024** Recording only	**Pre-read**: 1. JavaScript for Beginners (Section 5 dan 6) 2. Handling Strings in JavaScripts 3. JavaScript Arrays 4. JavaScript Functions 5. (Optional) Useful String methods **Homework:** **1. freeCodeCamp: Build A Roman Numeral Converter**
09	**JavaScript Algo. & Data Structures**	• JavaScript localStorage	Sabtu, **23 Nov. 2024** 19:00 - 21:00 WIB	**Pre-read:** • **localStorage Complete Guide** • **Using localStorage in JavaScript** **Homework**: 1. freeCodeCamp: Build a Telephone Number Validator