Web Dev Bootcamp 2024/25

JavaScript - Lab 09

07.12.2024

Sponsored by:

Agenda

1. Diskusi tentang tugas 2 minggu yang lalu
2. Object-oriented programing
3. Functional programming
4. Sorting Algorithms
5. Prep & To-Do untuk next session
6. Organisasi dan feedback

Diskusi tentang tugas 2 minggu yang lalu

- **Build a Telephone Number Validator** - Sari's solution

Main function

```
39  // Run the validation and handle the UI updates
40  const handleCheck = (inputElement, resultsContainer) => {
41      const regex = createPhoneNumberRegex(); // Calling the Regex function
42      const inputValue = inputElement.value;
43      const isValid = validatePhoneNumber(inputValue, regex); // Calling the phone number validation function
44
45      if (isValid !== null) {
46          const resultElement = createResultElement(isValid, inputValue);
47          appendResult(resultsContainer, resultElement); // If valid -> update UI
48      } else {
49          alert('Please provide a phone number'); // If not valid -> alert popup
50      }
51
52      inputElement.value = '';
53  };
```

Diskusi tentang tugas 2 minggu yang lalu

- **Build a Telephone Number Validator** - Sari's solution

Regex generator function

```
7   // Regexes untuk validasi US phone numbers, dikumpulin dalam satu function
8   const createPhoneNumberRegex = () => {
9       const countryCode = '^(1\\s?)?';
10      const areaCode = '(\\([0-9]{3}\\)|[0-9]{3})';
11      const spacesDashes = '[\\s\\-]?';
12      const phoneNumber = '[0-9]{3}[\\s\\-]?[0-9]{4}$';
13
14      return new RegExp(`${countryCode}${areaCode}${spacesDashes}${phoneNumber}`);
15  };
```

Input validation, Regex test method

```
17  // Phone number validation function
18  // to validate the input against the regex
19  const validatePhoneNumber = (input, regex) => {
20      if (input === '') return null; // Input validation kalau inputnya empty string
21
22      return regex.test(input);
23  };
```

Object-oriented programming

Sesuai namanya, OOP ini fokus ke '**Objects**' dalam suatu code/sistem.

- Objects ini bisa digrup menjadi **Classes** atau **Entities**
  - **Inheritance**, parent-child relationship antara objects dalam suatu Class.

```
...

class Bug {
    constructor (name, phrase, power) {
        this.name = name
        this.phrase = phrase
        this.power = power
        this.species = "bug"
    }
    hide = () => console.log("You can't catch me now!")
    sayPhrase = () => console.log(this.phrase)
    attack = () => console.log(`I'm attacking with a power of ${this.power}!`)
}

class Robot {
    constructor (name, phrase, power) {
        this.name = name
        this.phrase = phrase
        this.power = power
        this.species = "robot"
    }
    transform = () => console.log("Optimus prime!")
    sayPhrase = () => console.log(this.phrase)
    attack = () => console.log(`I'm attacking with a power of ${this.power}!`)
}

const bug1 = new Bug("Buggy", "Your debugger doesn't work with me!", 10)
const Robot1 = new Robot("Tito", "I can cook, swim and dance!", 15)

console.log(bug1.power) //output: 10
Robot1.attack() // output: "I'm attacking with a power of 15!"
```

Source: https://www.freecodecamp.org/news/object-oriented-javascript-for-beginners/

Object-oriented programming

Sesuai namanya, OOP ini fokus ke '**Objects**' dalam suatu code/sistem.

- Objects ini bisa digrup menjadi **Classes** atau **Entities**
  - **Polymorphism**,
    yaitu suatu method bisa menghasilkan (return) value yang berbeda, tergantung dari kondisi yang sudah diberikan.

```
const alien2 = new Alien("Lien", "Run for your lives!", 15, 60)
const bug1 = new Bug("Buggy", "Your debugger doesn't work with me!", 25, 100)

alien2.sayPhrase() // output: "Run for your lives!"
bug1.sayPhrase() // output: "Your debugger doesn't work with me!"
```

Source: https://www.freecodecamp.org/news/object-oriented-javascript-for-beginners/

Object-oriented programming

Sesuai namanya, OOP ini fokus ke '**Objects**' dalam suatu code/sistem.

- Objects bisa berisikan **data**, dan juga **methods**
  - **Encapsulation**, di sini data dan functions digabung menjadi dalam satu Class.
    - Variable → **Attributes**
    - Functions → **Methods**
    - Private properties / **Abstraction**

```
// Here's our class
class Alien extends Enemy {
    constructor (name, phrase, power, speed) {
        super(name, phrase, power, speed)
        this.species = "alien"
    }
    fly = () => console.log("Zzzzzziiiiiinnnnngggggg!!")
}

// Here's our object
const alien1 = new Alien("Ali", "I'm Ali the alien!", 10, 50)

// Here we're accessing our public properties and methods
console.log(alien1.name) // output: Ali
alien1.sayPhrase() // output: "I'm Ali the alien!"
```

Source: https://www.freecodecamp.org/news/object-oriented-javascript-for-beginners/

Object-oriented programming

Sesuai namanya, OOP ini fokus ke '**Objects**' dalam suatu code/sistem.

- Dengan OOP, data dan methods ini bisa selalu berdekatan
  - Processing dan transformasi object
  - State and type of the data

More on OOP:

- https://www.freecodecamp.org/news/object-oriented-javascript-for-beginners
- https://www.freecodecamp.org/news/prototypes-and-inheritance-in-javascript/

Functional programming

Seperti namanya, functional programming (FP) ini fokus ke **functions**.

Tidak seperti OOP, di FP ini objects dan functions sengaja dipisahkan.

Jadi functions dideklarasikan di lokasi yang berbeda dari objects, dan akan di*call* setiap function itu diperlukan → **isolated functions**.

Ketika setiap *action* atau proses dibuat menjadi satu spesifik function, maka code kita akan jadi lebih rapi:

- Better traceability
- Easier to debug
- Immutable
  (tidak bisa diubah-ubah)

```
const ages = [12,32,32,53]
const newAges = ages.map(function (age){
    if (age == 12) { return 20; }
    else { return age; }
})
```

Source: https://www.freecodecamp.org/news/functional-programming-in-javascript/

Functional programming

- Reusable functions
- Cleaner code

More on functional programming:

- https://www.freecodecamp.org/news/functional-programming-in-javascript/
- https://hackernoon.com/functional-programming-with-javascript-a-deep-dive

```
/// So here's an example where we have to copy and paste it
function add50(num) {
    return num + 50;
}

// Ok. Now we need to add 30. But we still ALSO need elsewhere to add 50 still
// So we need a new function
function add30(num){
    return num + 30;
}

// Ugh, business change again
function add20(num){
    return num + 20;
}

// Everytime we need to change the function ever so slightly. We need a new functio

//Let's use composition

// Our small, reusable pure function
function add10(num){
    return num + 10;
}

function add50Composed(num){
    return add10(add10(add10(add10(addNum(num)))));
}

function add30Composed(num){
    return add10(add10(add10(num)));
}

function add20Composed(num){
    return add10(add10(num));
}
```

Source: https://www.freecodecamp.org/news/functional-programming-in-javascript/

OOP vs FP

Di JavaScript & Web Development, kebanyakan kita menggunakan blended programming concept antara OOP dan FP.

More on OOP vs FP:

- https://www.datacamp.com/tutorial/functional-programming-vs-object-oriented-programming
- https://circleci.com/blog/functional-vs-object-oriented-programming
- https://careerfoundry.com/en/blog/web-development/functional-programming-vs-oop/

Functional Programming (FP)	Object-Oriented Programming (OOP)
Key Concepts	
Pure functions	Encapsulation
Avoids shared state and mutable data	Abstraction
	Inheritance
	Polymorphism
Programming Model	
Declarative	Imperative
Popular Languages	
Haskell	Java
Clojure	C++
Erlang	Python
Pros	
Reliable results without side effects	Easy to read
Emphasis on efficiency and optimization	Easy to understand
Cons	
Harder to read	Can lead to unpredictable results
Best for	
If you have a fixed set of things and you need to add operations to them	When you have a fixed set of operations on things, and you need to add more things

Source: https://careerfoundry.com/en/blog/web-development/functional-programming-vs-oop/

10 mins. break :)
See you @ 20:00

Sorting algorithms

Biasanya ***sort*** ini dilakukan ke data bertipe **Array**.

Kurang lebih ada dua jenis sort berdasarkan jenis sortingnya:

- Alphabetical sorts
- Numerical sorts

Di JavaScript, untuk numerical sort ini, bisa menggunakan native JS methods seperti `sort()` dan `map()`.

More on sort in JS:

- https://www.w3schools.com/js/js_array_sort.asp
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort
- https://www.freecodecamp.org/news/how-does-the-javascript-sort-function-work/

```
const stringArray = ["Blue", "Humpback", "Beluga"];
const numberArray = [40, 1, 5, 200];
const numericStringArray = ["80", "9", "700"];
const mixedNumericArray = ["80", "9", "700", 40, 1, 5, 200];

function compareNumbers(a, b) {
  return a - b;
}

stringArray.join(); // 'Blue,Humpback,Beluga'
stringArray.sort(); // ['Beluga', 'Blue', 'Humpback']

numberArray.join(); // '40,1,5,200'
numberArray.sort(); // [1, 200, 40, 5]
numberArray.sort(compareNumbers); // [1, 5, 40, 200]

numericStringArray.join(); // '80,9,700'
numericStringArray.sort(); // ['700', '80', '9']
numericStringArray.sort(compareNumbers); // ['9', '80', '700']

mixedNumericArray.join(); // '80,9,700,40,1,5,200'
mixedNumericArray.sort(); // [1, 200, 40, 5, '700', '80', '9']
mixedNumericArray.sort(compareNumbers); // [1, 5, '9', 40, '80', 200, '700']
```

Source: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

Sorting algorithms

Kemudian, ada beberapa *sorting algorithms* yang sering dipakai, contohnya:

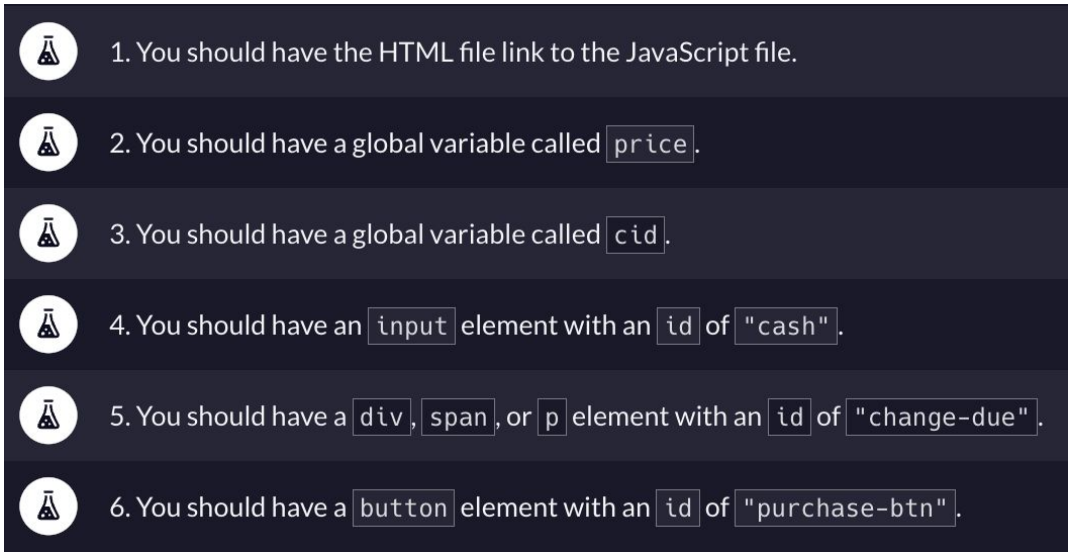
- Quick sort
- Bucket sort
- Counting sort
- Heap sort

More on sorting algorithms:

- https://www.freecodecamp.org/news/sorting-algorithms-explained-with-examples-in-python-java-and-c/
- https://medium.com/free-code-camp/an-intro-to-advanced-sorting-algorithms-merge-quick-radix-sort-in-javascript-b65842194597
- https://youtu.be/IAeLoGzU4RE?si=iln52x53ugHI-nbG

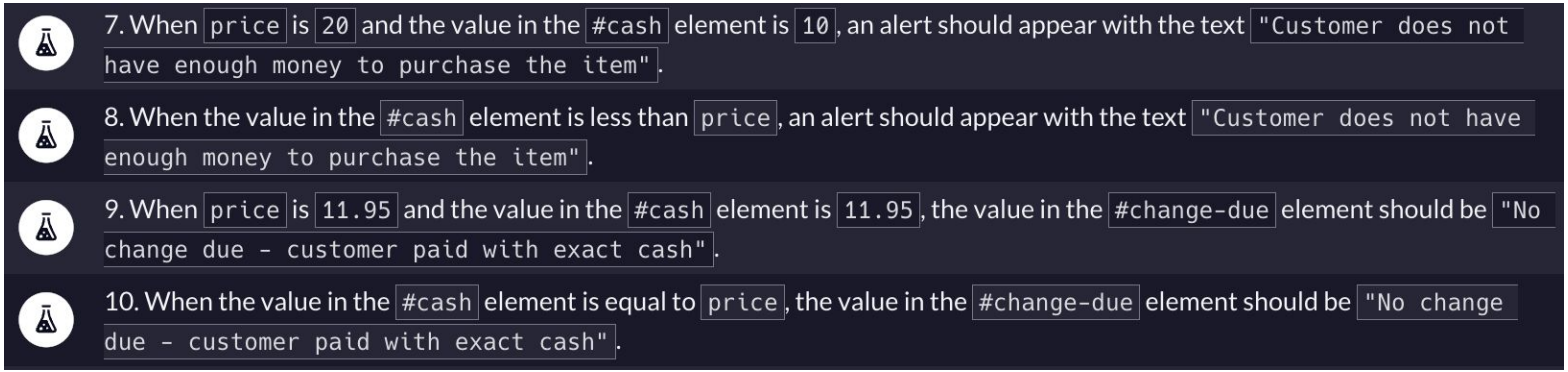
To-do untuk next session

- HTML & CSS



To-do untuk next session

- Simple if conditions



To-do untuk next session

- Recursion (loop), math rounding, if conditions

11. When `price` is `19.5`, the value in the `#cash` element is `20`, `cid` is `[["PENNY", 1.01], ["NICKEL", 2.05], ["DIME", 3.1], ["QUARTER", 4.25], ["ONE", 90], ["FIVE", 55], ["TEN", 20], ["TWENTY", 60], ["ONE HUNDRED", 100]]`, and the `#purchase-btn` element is clicked, the value in the `#change-due` element should be `"Status: OPEN QUARTER: $0.5"`.

12. When `price` is `3.26`, the value in the `#cash` element is `100`, `cid` is `[["PENNY", 1.01], ["NICKEL", 2.05], ["DIME", 3.1], ["QUARTER", 4.25], ["ONE", 90], ["FIVE", 55], ["TEN", 20], ["TWENTY", 60], ["ONE HUNDRED", 100]]`, and the `#purchase-btn` element is clicked, the value in the `#change-due` element should be `"Status: OPEN TWENTY: $60 TEN: $20 FIVE: $15 ONE: $1 QUARTER: $0.5 DIME: $0.2 PENNY: $0.04"`.

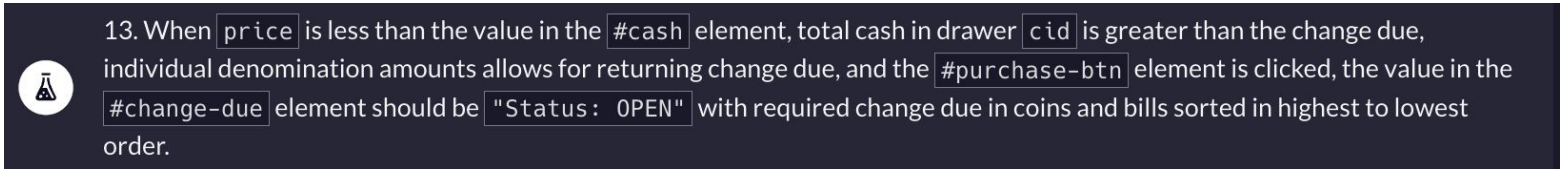
14. When `price` is `19.5`, the value in the `#cash` element is `20`, `cid` is `[["PENNY", 0.01], ["NICKEL", 0], ["DIME", 0], ["QUARTER", 0], ["ONE", 0], ["FIVE", 0], ["TEN", 0], ["TWENTY", 0], ["ONE HUNDRED", 0]]`, and the `#purchase-btn` element is clicked, the value in the `#change-due` element should be `"Status: INSUFFICIENT_FUNDS"`

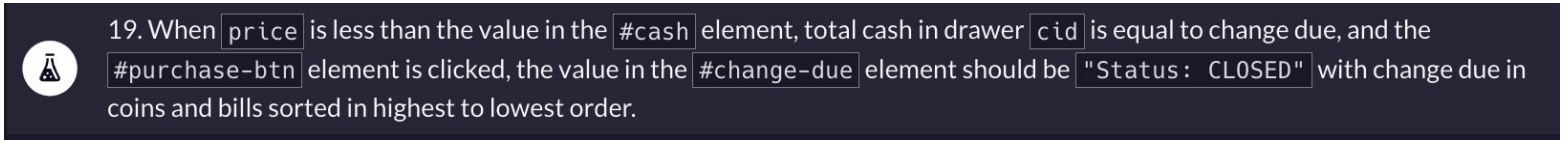
16. When `price` is `19.5`, the value in the `#cash` element is `20`, `cid` is `[["PENNY", 0.01], ["NICKEL", 0], ["DIME", 0], ["QUARTER", 0], ["ONE", 1], ["FIVE", 0], ["TEN", 0], ["TWENTY", 0], ["ONE HUNDRED", 0]]`, and the `#purchase-btn` element is clicked, the value in the `#change-due` element should be `"Status: INSUFFICIENT_FUNDS"`.

18. When `price` is `19.5`, the value in the `#cash` element is `20`, `cid` is `[["PENNY", 0.5], ["NICKEL", 0], ["DIME", 0], ["QUARTER", 0], ["ONE", 0], ["FIVE", 0], ["TEN", 0], ["TWENTY", 0], ["ONE HUNDRED", 0]]`, and the `#purchase-btn` element is clicked, the value in the `#change-due` element should be `"Status: CLOSED PENNY: $0.5"`.

To-do untuk next session

- If conditions, math rounding, sorting





To-do untuk next session

- If conditions

15. When the `price` is less than the value in the `#cash` element and the total cash in the drawer (`cid`) is insufficient to cover the change due, the purchase should not proceed. When the `#purchase-btn` is clicked under these conditions, the `#change-due` element should display `"Status: INSUFFICIENT_FUNDS"`.

17. When `price` is less than the value in the `#cash` element, total cash in drawer `cid` is greater than change due, but the individual denomination amounts make it impossible to return needed change, when the `#purchase-btn` element is clicked, the value in the `#change-due` element should be `"Status: INSUFFICIENT_FUNDS"`

To-do untuk next session

10	**JavaScript Algo. & Data Structures**	• Sorting Algorithms • Object-Oriented vs Functional Programming	Sabtu, **7 Des. 2024** 19:00 - 21:00 WIB	**Pre-read**: • Sorting Algorithms Explained • What is Functional Programming? • OOP in JavaScripts for Beginners • (Optional) JavaScript OOP Tutorial **Homework**: 1. **freeCodeCamp: Build a Cash Register**
11	**JavaScript Algo. & Data Structures**	• Asynchronous Programming • Fetch and Promises	Sabtu, **14 Des. 2024** 19:00 - 21:00 WIB	**Pre-read**: 1. Async vs Synchronous JavaScript 2. Async JavaScript Tutorial **Homework**: 1. freeCodeCamp: Build a Pokemon Search App

Organisasi dan feedback

- Feedback: **https://forms.gle/bGTYp3QaWp9Sqcqg8**
- 21 Desember:
  - Showcase updated portfolio + Foto bareng untuk CloudKilat 🤗
  - Kasih testimoni tentang bootcamp ini ke CloudKilat
  - Siapa aja yang mau portfolionya dipamerin di social medianya CloudKilat & LinkedIn?
- 28 Desember:
  - Skip dulu (liburan tahun baru) atau zoom seperti biasa?
- Setelah JavaScript:
  - Lanjut ke backend :)
    https://www.freecodecamp.org/learn/back-end-development-and-apis/