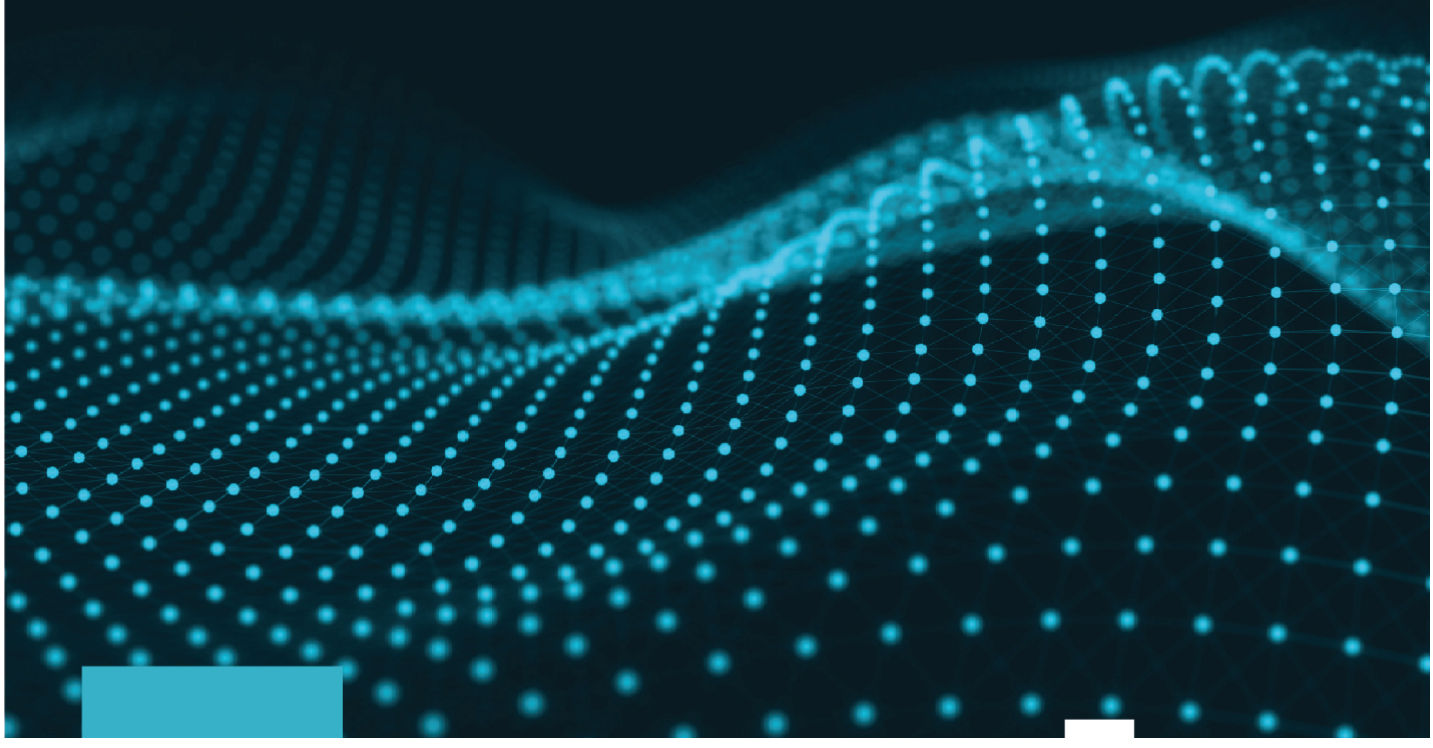
Sprint 02

Web Frontend SE

September 10, 2020



Contents


ucode

Engage

DESCRIPTION

Congratulations on hitting a new milestone!

You have learned the important aspects and got a good foundation of HTML and CSS and can now create basic web pages. But why stop there? It's time to get your hands on some real programming tasks.

Of course, there are many programming languages, but JavaScript (JS) has established itself as one of the most common programming languages of frontend development, so it's a logical starting point in our case. JavaScript will show you that you can do so much more with web pages than just text and pretty pictures! JavaScript makes a website dynamic. A dynamic web page can contain client-side and/or server-side scripts to create mutable content.

In this Sprint, you will gain a general understanding of JavaScript and get acquainted with the basics of programming. It will expand your understanding and mindset. You will learn how to add functionality to a web page with a few additional lines of code.

Let's get started!

BIG IDEA

Introduction to JavaScript.

ESSENTIAL QUESTION

What is the role of JavaScript in web frontend development?

CHALLENGE

Learn the programming basics with JavaScript.


ucode
Sprint 02 | Web Frontend SE > 2

Investigate

GUIDING QUESTIONS

We invite you to find answers to the following questions. By researching and answering them, you will gain the knowledge necessary to complete the challenge. To find answers, ask the students around you and search the internet. We encourage you to ask as many questions as possible. Note down your findings and discuss them with your peers.

- What is `JavaScript`?
- Why is JS used for web frontend development?
- How to connect a JS script to an HTML file?
- What is a `variable` in programming?
- What is a `loop`?
- What is the difference between the loops: `while`, `do while`, and `for`?
- What is a `function`?
- What `methods` exist to display something in JS?
- What is a `data type`? Which types exist in JS?
- What is a `string` in programming?
- What are `type conversions`?
- Is there `Infinity` in JS? What role does it play?
- Do `logical operators` simplify development?
- What is an `Array`? What operations can be done with them?

GUIDING ACTIVITIES

Complete the following activities. Don't forget that you have a limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Read the story tasks.
- Find some informative resources with JavaScript documentation.
- Create an HTML file and add a JS script to it.
- Research the syntax and code structure of JavaScript.
- Read about debugging inside a browser (e.g Chrome).
- Learn recommendations on good code style for JS.
- Clone your git repository that is issued on the challenge page in the LMS.
- Start to develop the solution. Offer improvements. Test your code.
- Employ the full power of P2P by brainstorming with other students.


ucode
Sprint 02 | Web Frontend SE > 3

ANALYSIS

Analyze your findings. What conclusions have you made after completing guiding questions and activities? In addition to your thoughts and conclusions, here are some more analysis results.

- Be attentive to all statements of the story. Examine the given examples carefully. They may contain details that are not mentioned in the task.
- Analyze all information you have collected during the preparation stages.
- Complete the `Document` section while developing a challenge. It is described after the `Act` section.
- Perform only those tasks that are given in this document.
- Submit only the specified files in the required directory and nothing else. Garbage shall not pass.
- Pay attention to what is allowed. Use of forbidden stuff is considered a cheat and your challenge will be failed.
- The web page in the browser must open through `index.html`.
- The scripts must be written outside the HTML file - in a separate JS file (script.js) (except for Task00 in this Sprint).
- You can always use the `Console` panel to test and catching errors.
- Complete tasks according to the rules specified in the following style guides:
  - HTML and CSS: Google HTML/CSS Style Guide. As per section 3.1.7 Optional Tags, it doesn't apply. Do not omit optional tags, such as `<head>` or `<body>`
  - JavaScript:
    * JavaScript Style Guide and Coding Conventions
    * JavaScript Best Practices
- The solution will be checked and graded by students like you. Peer-to-Peer learning.
- If you have any questions or don't understand something, ask other students or just Google it.


ucode
Sprint 02 | Web Frontend SE > 4

Act: Task 00

NAME

Inside HTML

DIRECTORY

`t00/`

SUBMIT

`index.html`

ALLOWED FUNCTIONS

`alert()`

DESCRIPTION

Create a web page that runs a JS (JavaScript) script. The script must:

- be written inside an HTML file
- use the `alert()` method to show the message `Hello JavaScript!`
- contain a 1-row comment with a description of the `alert()` method

Keep in mind that the Google HTML/CSS Style Guide advises against the practice of mixing HTML and CSS or JS in one document. Implement it here as an exercise in order to know how it can be done, but avoid doing it in the future.

SEE ALSO

JavaScript Guide
Window alert() Method


ucode
Sprint 02 | Web Frontend SE > 5

Act: Task 01

NAME

Outside HTML

DIRECTORY

`t01/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert()`

DESCRIPTION

Copy the HTML web page you have created for the Task00. In this task, you need to add to it one more JS script, and this time it will be in a separate file. The script must:

- be written outside the HTML file, in a separate JS file called `script.js`
- have a variable with the value `Hello JavaScript from outside!`
- use the `alert()` method to show the message with the value of the variable
- contain a 2-row comment describing how to declare a variable in JS

SEE ALSO

JavaScript Variables


ucode
Sprint 02 | Web Frontend SE > 6

Act: Task 02

NAME

Words of wisdom

DIRECTORY

`t02/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

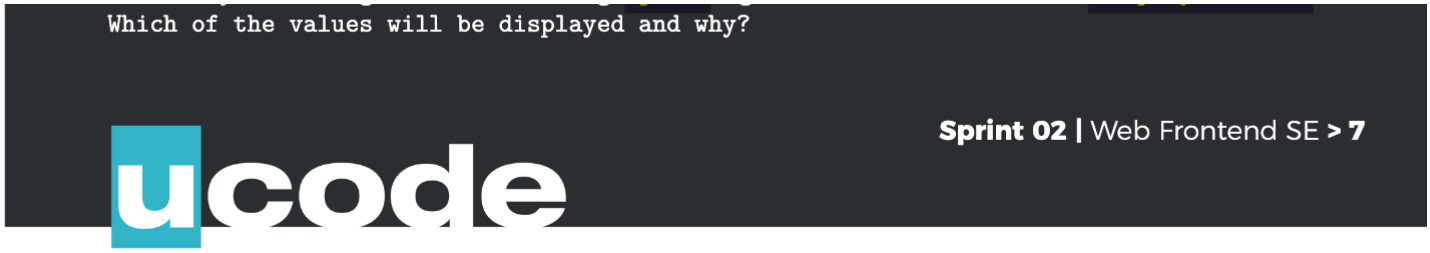
`alert()`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS. Within that JS file, complete the following steps. Make sure to follow the order of the steps perfectly. Here are the steps:

- declare a variable `quote` using the `var` keyword
- use `alert()` to display `quote`
- to the variable `quote`, assign the value
  `With great power, comes great responsibility.`
- again, use `alert()` to display `quote`
- create a function `displayAuthor()`
- inside the function:
  - declare and initialize (in one line) the variable `author` with the value
    `- Spiderman`
  - use `alert()` to display `author`
  - reassign the value of `quote` to `...`
- once again, use `alert()` to display `quote`

Separately from the submitted solution, play around with this code to understand the concept of scope better. For example, try to change or display the value of `author` outside of the function `displayAuthor()`. Open the console in browser developer tools to see what errors appear.
Also, try declaring and initializing `quote` again inside the function `displayAuthor()`.


Which of the values will be displayed and why?
ucode
Sprint 02 | Web Frontend SE > 7

SYNOPSIS

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Words of wisdom</title>
  <meta name="description" content="t02. Words of wisdom">
</head>

<body>
  <h1>Words of wisdom</h1>

   <script src="js/script.js"></script>
</body>

</html>
```

SEE ALSO

JavaScript Functions
Understanding Scope in JavaScript


ucode
Sprint 02 | Web Frontend SE > 8

Act: Task 03

NAME

Fair and square

DIRECTORY

`t03/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert()`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must:

- contain a function `myFunc()` that calls the `alert()` method with the value `Freedom is power`
- call the function `myFunc()`
- contain a function `countMySquare(number)` that calculates the square of the incoming number
- show a message with the result, for example, `The square of 2 = 4`.
  Note: you must not hardcode this value, you must output numbers that are passed directly to the function. Test it with different numbers to check!

SYNOPSIS

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Fair and square</title>
  <meta name="description" content="t03. Fair and square">
</head>

<body>
  <h1>Fair and square</h1>
```


ucode
Sprint 02 | Web Frontend SE > 9

```
  <script src="js/script.js"></script>
</body>

</html>
```

SEE ALSO

Functions in JavaScript


ucode
Sprint 02 | Web Frontend SE > 10

Act: Task 04

NAME

Solver

DIRECTORY

`t04/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert(), Number.*, Math.*`

DESCRIPTION

Create a function inside a JS file that will be included into the HTML page written in the SYNOPSIS.
The function takes values of a, b, c, d, e, and solves the following equation:

$$x = a^2 - 5bc + \frac{d}{3} + e$$

The `solver()` function returns the value of x, or "Wrong input" if one of the variables is not a number or is missing.

You can test your function using the test.js file written in the EXAMPLE. Add more test cases of your own.

SYNOPSIS

```
  solver(a, b, c, d, e)
```

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Solver</title>
  <meta name="description" content="t04. Solver">
</head>
```


ucode
Sprint 02 | Web Frontend SE > 11

```
<body>
  <h1>Solver</h1>

  <script src="js/script.js"></script>
  <!-- <script src="js/test.js"></script> uncomment this line when testing -->
</body>

</html>
```

EXAMPLE

```
/*
    Equation:
    x = a ^ 2 - 5 * b * c + d / 3 + e
    Solve for x.
*/

alert(solver(40, -9, 0.5, 7, 100));
// displays 1724.83

alert(solver(40, "doesn't look like a number", 0.5, 7, 100));
// displays "Wrong input"

alert(solver(40, -9, 0.5, 7));
// displays "Wrong input"
```

SEE ALSO

How To Do Math in JavaScript with Operators
Number.prototype.toFixed()


ucode
Sprint 02 | Web Frontend SE > 12

Act: Task 05

NAME

Sooo..., a string

DIRECTORY

`t05/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert(), String.*`

DESCRIPTION

In this task, you will get to know strings and how you can work with them. The EXAMPLE below shows the output that you must be able to get, but you must do this with the following manipulations.
Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must contain a part that:

- contains a `str1` string with value You're catnip to a girl like me. Handsome, dazed, and to die for... and count its length
- returns the character located at the specified index of the `str1`
- converts all characters of the `str1` string to uppercase
- concatenation `str1` with - Catwoman
- contains a `str2` string with value Batman: "I tried to save you."
- contains a `str3` string with value Selina Kyle: catwoman"Mmm seemsCatwomanlike everyCatWomanwoman you try to save windsCatWOMANup dead... or deeply resentful." and replaces all "catwoman" to " ", not case-sensitive
- calls the `alert()` method to display information from the EXAMPLE

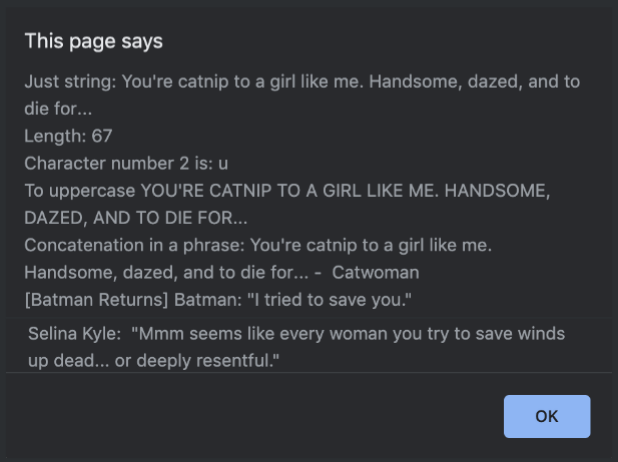
If you want, you can create your own functions for each of the manipulations.

SYNOPSIS


ucode
Sprint 02 | Web Frontend SE > 13

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Sooo..., a string</title>
  <meta name="description" content="t05. Sooo..., a string">
</head>

<body>
  <h1>Sooo..., a string</h1>

  <script src="js/script.js"></script>
</body>

</html>
```

EXAMPLE



SEE ALSO

JavaScript Strings
JavaScript String Methods
15 JavaScript String Functions


ucode
Sprint 02 | Web Frontend SE > 14

Act: Task 06

NAME

What type of data?

DIRECTORY

`t06/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert()`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must:

- create variables for the following data types in JavaScript and assign the appropriate values:
  - Number
  - BigInt
  - String
  - Boolean
  - Null
  - Undefined
  - Object
  - Symbol
  - Function
- display, at once, all the variable names and their data types in the following format: `variable_name is data_type\n` with `alert()` method

Note: in this task `typeof` will help you. One of the outputs may surprise you, because there is a known error in the JS language. Don't be scared and read SEE ALSO.

SYNOPSIS


ucode
Sprint 02 | Web Frontend SE > 15

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>What type of data?</title>
  <meta name="description" content="t06. What type of data?">
</head>

<body>
  <h1>What type of data?</h1>

  <script src="js/script.js"></script>
</body>

</html>
```

SEE ALSO

JavaScript Fundamentals Data types
String Interpolation in JavaScript


ucode
Sprint 02 | Web Frontend SE > 16

Act: Task 07

NAME

Not my type

DIRECTORY

`t07/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert()`

DESCRIPTION

Since JavaScript is a weakly typed language, it allows for type coercion. It's both a blessing and a curse. While allowing for more freedom, it's prone to unexpected errors.

Create a JS file that will be included into the HTML page in the SYNOPSIS.
Copy the code from the EXAMPLE into your JS file. Your goal is for the output to match what is written in the comment.

You're only allowed to edit the third parameter in each function call to `getAnswer()`.
Nothing else. Edit the expressions in the third parameter in such a way that the page output matches what is written in the comment section. You can only use explicit conversion to achieve the result. Don't change the order of operations, values, variables, etc.

SYNOPSIS

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Not my type</title>
  <meta name="description" content="t07. Not my type">
</head>

<body>
  <h1>Not my type</h1>
```


ucode
Sprint 02 | Web Frontend SE > 17

```
  <script src="js/script.js"></script>
</body>

</html>
```

EXAMPLE

```
function getAnswer(question, wrongAnswer, correctAnswer){
  return(`${question}\nwrong: ${wrongAnswer}\ncorrect: ${correctAnswer}\n\n`);
}

const a = '15';
const b = 4;
const c = true;
const d = 'a';
const e = 'B';
const f = '';

let answer = 'Not my type\n\n';

answer += getAnswer('a+b=?', a + b, a + b);
answer += getAnswer('a+c=?', a + c, a + c);
answer += getAnswer('e+d+e+d=?', e + d + e + d, e + d + e + d);
answer += getAnswer('c+f=?', c + f, c + f);

alert(answer);

/* The web page must display:
Not my type

  a+b=?
  wrong: 154
  correct: 19

  a+c=?
  wrong: 15true
  correct: 2

  e+d+e+d=?
  wrong: BaBa
  correct: BaNaNa

  c+f=?
  wrong: true
  correct: 10
```


*/
ucode
Sprint 02 | Web Frontend SE > 18

SEE ALSO

Type Conversions


ucode
Sprint 02 | Web Frontend SE > 19

Act: Task 08

NAME

Special numeric values

DIRECTORY

`t08/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert(), prompt()`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script takes a number from 1 to 4 as an input value, and returns the operation result as an alert. The result must depend on the input value in the following way:

- 1 - multiply the input value by 2 and divide by 'a'
- 2 - subtract the input value from the input value and then divide by zero
- 3 - multiply zero by Infinity
- 4 - multiply the input value by 40000000 and compare with Infinity

If the input value not from 1 to 4, the `alert()` method displays Wrong input.
Learn more about NaN and Infinity and in what cases they are used.

SYNOPSIS

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Special numeric values</title>
  <meta name="description" content="t08. Special numeric values">
</head>

<body>
```


<h1>Special numeric values</h1>
ucode
Sprint 02 | Web Frontend SE > 20

```
  <script src="js/script.js"></script>
</body>

</html>
```

SEE ALSO

JavaScript if else and else if
JavaScript NaN Property
JavaScript Infinity Property


ucode
Sprint 02 | Web Frontend SE > 21

Act: Task 09

NAME

Display heroes

DIRECTORY

`t09/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert()`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must contain a function that displays heroes in accordance with the following rules:

- This is Black Widow! if:
  - team: "Avengers" or "S.H.I.E.L.D."
  - universe: "Marvel"
  - race: "human"
  - eyeColor: "green"
  - hairColor: "lightBrown/green"
- This is a Superman or Robin! if:
  - teams: "Justice League Of America" or "Teen Titans"
  - universe: "DC Comics"
  - race: "human" or "kryptonian"
  - eyeColor: "blue"
  - hairColor: "black"
- Didn't recognize! in other cases

You must use the `logical operators` for implementation.


ucode
Sprint 02 | Web Frontend SE > 22

SYNOPSIS

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Display heroes</title>
  <meta name="description" content="t09. Display heroes">
</head>

<body>
  <h1>Display heroes</h1>

  <script src="js/script.js"></script>
</body>

</html>
```

SEE ALSO

Logical operators


ucode
Sprint 02 | Web Frontend SE > 23

Act: Task 10

NAME

Superhero name maker

DIRECTORY

`t10/`

SUBMIT

`index.html, js/script.js`

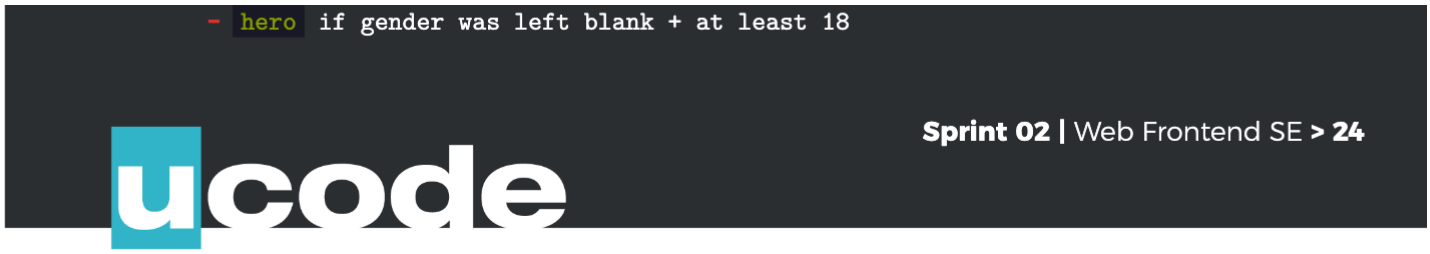
ALLOWED FUNCTIONS

`alert(), prompt(), RegExp.*`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
Make a script that generates superhero names based on input.
The script must:

- prompts the user to enter input three times:
  1. to enter an animal name: `What animal is the superhero most similar to?`
     - Input requirements: length <= 20, only one word that contains only letters
  2. to enter gender: `Is the superhero male or female? Leave blank if unknown or other.`
     - Input requirements: accepts only `male`, `female`, or blank (not case sensitive)
  3. to enter age: `How old is the superhero?`
     - Input requirements: length <= 5, only digits, cannot start with a zero
- checks input for validity using regular expression (also known as regex)
- if the input is not valid, displays an error message using `alert` and stops executing
- generates a description for the superhero depending on the entered gender and age:
  - `boy` if `male` + younger than 18
  - `man` if `male` + at least 18
  - `girl` if `female` + younger than 18
  - `woman` if `female` + at least 18
  - `kid` if gender was left blank + younger than 18


- hero if gender was left blank + at least 18
ucode
Sprint 02 | Web Frontend SE > 24

- displays `The superhero name is: [enteredAnimal]-[description]!`

So, for example, if the user entered: "bat", "Male", "25", the message will be:
`The superhero name is: bat-man!`

SYNOPSIS

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Superhero name maker</title>
  <meta name="description" content="t10. Superhero name maker">
</head>

<body>
  <h1>Superhero name maker</h1>

  <script src="js/script.js"></script>
</body>

</html>
```

SEE ALSO

Making decisions in your code - conditionals
Regular expressions


ucode
Sprint 02 | Web Frontend SE > 25

Act: Task 11

NAME

Once in a while

DIRECTORY

`t11/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert()`

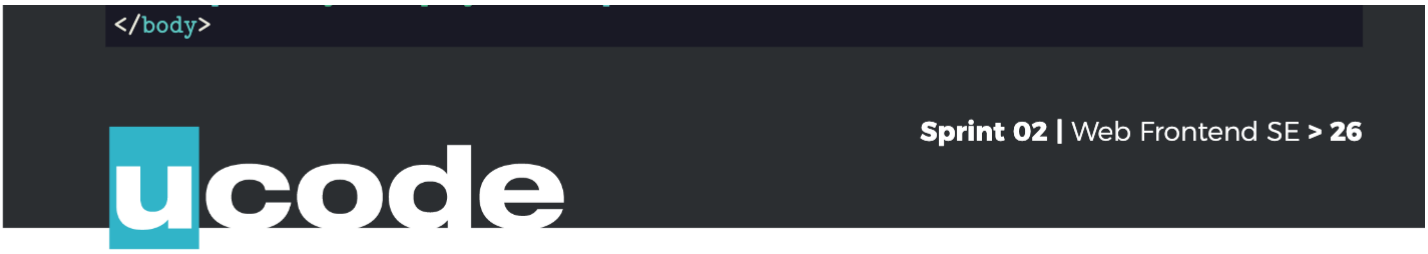
DESCRIPTION

Now your task is to understand what loops in programming are and their differences.

1. Look at the EXAMPLE. Two loops are shown there. Find out and understand how they work. Copy and test this example. In what case is nothing displayed? Try to change this situation. Also, you can change the values and replace the increment/decrement operation to change the behavior of the loops. Do not forget to look in SEE ALSO.
2. Create a JS file that will be included into the HTML page written in the SYNOPSIS. The script must display the multiplication table for the number of your choice and contain `while` or `do...while` loop for implementation.

SYNOPSIS

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>While/Do While</title>
  <meta name="description" content="t11. While/Do While">
</head>

<body>
  <h1>While/Do While</h1>

  <script src="js/script.js"></script>
```


</body>
ucode
Sprint 02 | Web Frontend SE > 26

```
</html>
```

EXAMPLE

```
// the first loop
var i = 6;
var res1 = ' ';

do {
        res1 += i + ' ';
        i++;
} while (i < 5);

// the second loop
i = 6;
var res2 = ' ';
while (i < 5)
{
        res2 += i + ' ';
        i++;
}

alert('do while: ' + res1 + 'while: ' + res2);
```

SEE ALSO

JavaScript while Statement
JavaScript do/while Statement


ucode
Sprint 02 | Web Frontend SE > 27

Act: Task 12

NAME

Triangle print

DIRECTORY

`t12/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert()`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must:

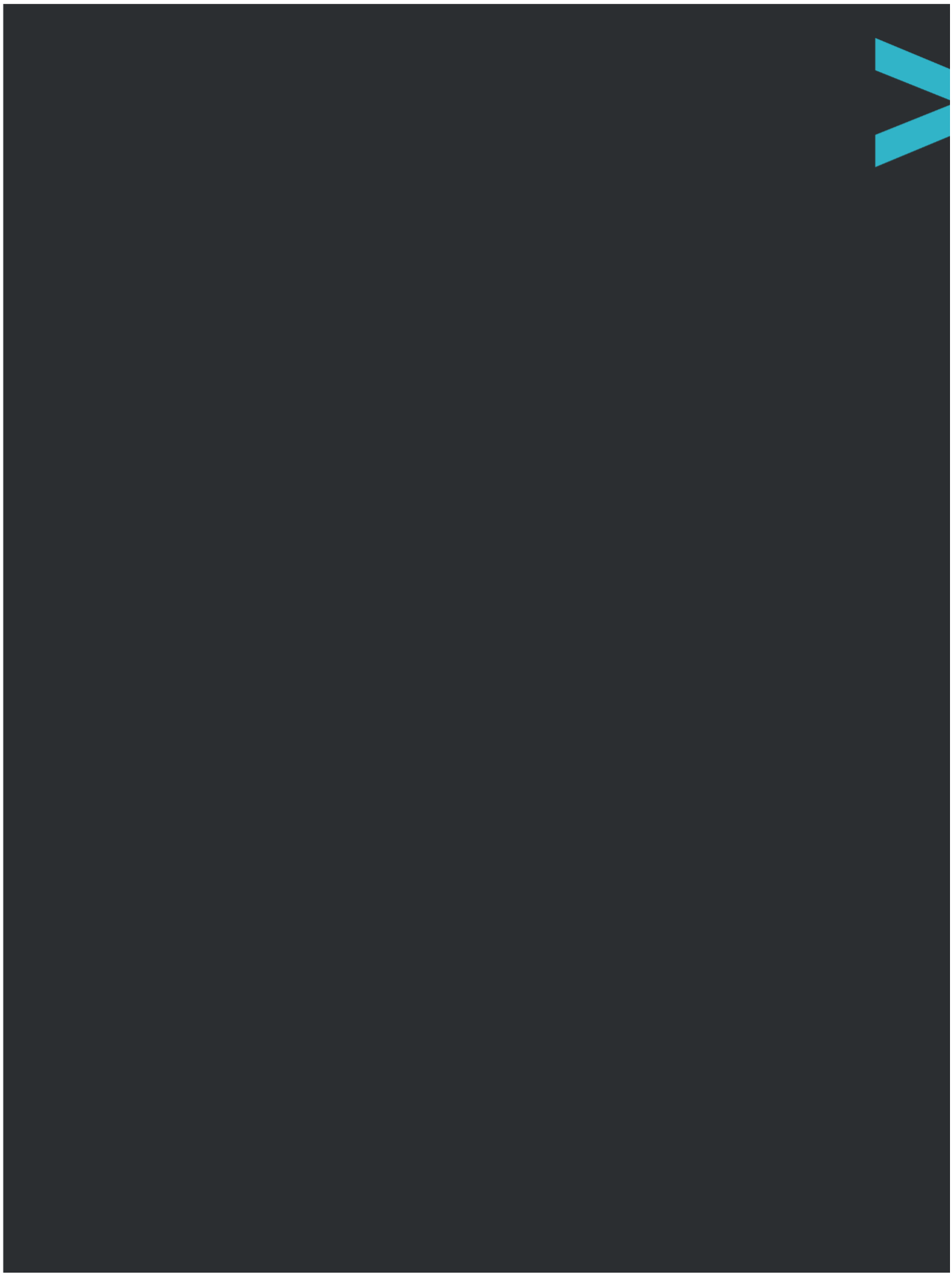
- contain a function that displays a triangle of input length. Use a `for` loop for implementation
- show the result with `alert()` method

You can see an example with a length of "6" in the EXAMPLE section.

SYNOPSIS

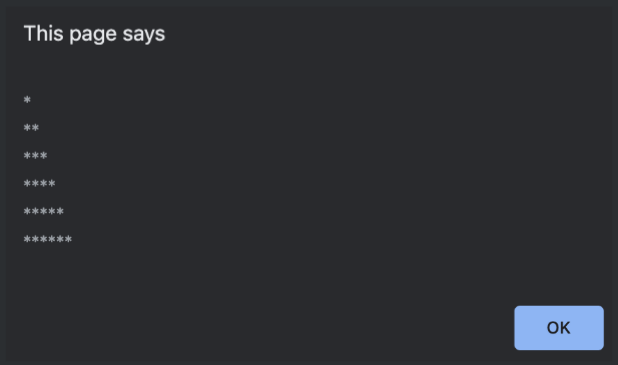
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Triangle print</title>
  <meta name="description" content="t12. Triangle print">
</head>

<body>
  <h1>Triangle print</h1>

  <script src="js/script.js"></script>
</body>
```


</html>
ucode
Sprint 02 | Web Frontend SE > 28




ucode
Sprint 02 | Web Frontend SE > 29

EXAMPLE



SEE ALSO

JavaScript For Loop


ucode
Sprint 02 | Web Frontend SE > 30

Act: Task 13

NAME

What kind of idiom?

DIRECTORY

`t13/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert(), prompt(), Number.*`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must:

- call a `prompt()` method and take a number from 1 to 10 as an input value
- check that the input value is a number, and exactly from 1 to 10
- show an idiom with `alert()` method

The idiom must depend on the input value in the following way:

1 - `Back to square 1`
2 - `Goody 2-shoes`
3 or 6 - `Two's company, three's a crowd`
4 or 9 - `Counting sheep`
5 - `Take five`
7 - `Seventh heaven`
8 - `Behind the eight-ball`
10 - `Cheaper by the dozen`

Note: You must use a `switch` statement for implementation. The conditional operator `if` is FORBIDDEN for this task.

SYNOPSIS


ucode
Sprint 02 | Web Frontend SE > 31

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>What kind of idiom?</title>
  <meta name="description" content="t13. What kind of idiom?">
</head>

<body>
  <h1>What kind of idiom?</h1>

  <script src="js/script.js"></script>
</body>

</html>
```

SEE ALSO

JavaScript Switch Statement
Window prompt() Method
JavaScript Number isFinite() Method


ucode
Sprint 02 | Web Frontend SE > 32

Act: Task 14

NAME

Total price

DIRECTORY

`t14/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`Number.*`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS. Imagine that you're shopping online, and every time you add something to your cart, this function is called. The script must contain a function that:

- takes three parameters:
  - Number of items
  - The price per item
  - The current total of the price
- returns the total order sum

Display and track the result in the Console panel.

You can test your function using the test.js file written in the EXAMPLE. It is appropriate to use default parameter in this task. Add more test cases of your own.

SYNOPSIS

```
total(addCount, addPrice, currentTotal) : number
```

```
<!DOCTYPE html>
<html lang="en">

<head>
```


<meta charset="UTF-8">
ucode
Sprint 02 | Web Frontend SE > 33

```
  <title>Total price</title>
  <meta name="description" content="t14. Total price">
</head>

<body>
  <h1>Total price</h1>

  <script src="js/script.js"></script>
  <!-- <script src="js/test.js"></script> uncomment this line when testing -->
</body>

</html>
```

EXAMPLE

```
let sum = total(1, 0.1);
sum = total(1, 0.2, sum);
sum = total(1, 0.78, sum);
console.log(sum); // will return 1.08
```

SEE ALSO

HTML DOM console.log() Method
Number.prototype.toFixed()


ucode
Sprint 02 | Web Frontend SE > 34

Act: Task 15

NAME

Numbers

DIRECTORY

`t15/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`prompt(), console.log(), String.*`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must:

- call `prompt()` and take the numbers for the beginning and end of a range
- contain a function that:
  - takes two number variables (inclusive range), and prints suitable descriptions for all numbers in the range to the Console panel. Descriptions:
    * 'number' is even
    * 'number' is a multiple of 3
    * 'number' is a multiple of 10
  - has default range of `1 - 100`

Look at the EXAMPLE of how the result may look like.

SYNOPSIS

```
checkDivision(beginRange, endRange)
```

```
<!DOCTYPE html>
<html lang="en">

<head>
```


<meta charset="UTF-8">
ucode
Sprint 02 | Web Frontend SE > 35

```
  <title>Numbers</title>
  <meta name="description" content="t15. Numbers">
</head>

<body>
  <h1>Numbers</h1>

  <script src="js/script.js"></script>
</body>

</html>
```

EXAMPLE

```
1 -
2 is even
3 is a multiple of 3
4 is even
5 -
...
60 is even, a multiple of 3, a multiple of 10
```


ucode
Sprint 02 | Web Frontend SE > 36

Act: Task 16

NAME

Heroes

DIRECTORY

`t16/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`Array.*`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must contain a function that:

- accepts an `array` and `n` parameters. `n` stands for the number of elements to take from the beginning of the array
- finds first `n` elements of the array
- returns a new array, containing first `n` elements of the original array

You can test your function using the test.js file written in the EXAMPLE. Add more test cases of your own.

SYNOPSIS

```
function firstElements(arr, n)
```

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Heroes</title>
  <meta name="description" content="t16. Heroes">
</head>
```


<body>
ucode
Sprint 02 | Web Frontend SE > 37

```
  <h1>Heroes</h1>

  <script src="js/script.js"></script>
  <!-- <script src="js/test.js"></script> uncomment this line when testing -->
</body>

</html>
```

EXAMPLE

```
var heroes = ["Captain America", "Hulk", "Thor", "Iron Man", "Spider-Man"];

console.log(firstElements(heroes, 5));
// ["Captain America", "Hulk", "Thor", "Iron Man", "Spider-Man"]

console.log(firstElements(heroes, 1));
// ["Captain America"]

console.log(firstElements(heroes, 3));
// ["Captain America", "Hulk", "Thor"]

console.log(firstElements(heroes, 6));
// ["Captain America", "Hulk", "Thor", "Iron Man", "Spider-Man"]

console.log(firstElements(heroes, -1));
// []
```

SEE ALSO

JavaScript Arrays
JavaScript Array Reference


ucode
Sprint 02 | Web Frontend SE > 38

Act: Task 17

NAME

Greeting

DIRECTORY

`t17/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`alert(), prompt(), console.log(), isNaN(), String.*`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must:

- prompt the user to enter their first name and last name
- check if the input is valid
- capitalize the first letter of the first and last name if it is not
- use `alert()` and the `Console` panel to greet the user using their full name
- display `Wrong input!` both to the `Console` panel and using `alert()` if a line contains a digit or other incorrect input

SYNOPSIS

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Greeting</title>
  <meta name="description" content="t17. Greeting">
</head>

<body>
  <h1>Greeting</h1>
```


ucode
Sprint 02 | Web Frontend SE > 39

```
  <script src="js/script.js"></script>
</body>

</html>
```


ucode
Sprint 02 | Web Frontend SE > 40

Act: Task 18

NAME

Place all

DIRECTORY

`t18/`

SUBMIT

`index.html, js/script.js`

ALLOWED FUNCTIONS

`Array.*`

DESCRIPTION

Create a JS file that will be included into the HTML page written in the SYNOPSIS.
The script must contain a function that:

- sorts a number array depending on whether the numbers are odd or even
- places all:
    - even numbers on the left
    - odd numbers on the right

See the EXAMPLE.
Note: Don't use a temporary array.

SYNOPSIS

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <title>Place all</title>
  <meta name="description" content="t18. Place all">
</head>

<body>
  <h1>Place all</h1>
```


ucode
Sprint 02 | Web Frontend SE > 41

```
  <script src="js/script.js"></script>
  <!-- <script src="js/test.js"></script> uncomment this line when testing -->
</body>

</html>
```

EXAMPLE

```
const arr = [6, 2, 15, 5, 1, 3, 8, 1, 8, 10, 7, 11];
sortEvenOdd(arr);
console.log(arr); // (12) [2, 6, 8, 8, 10, 1, 1, 3, 5, 7, 11, 15]
```


ucode
Sprint 02 | Web Frontend SE > 42

Act: Task 19

NAME

Brackets

DIRECTORY

`t19/`

SUBMIT

`index.html, js/script.js, js/test.js`

ALLOWED FUNCTIONS

`console.log(), String.*`

DESCRIPTION

Create a script with a function `checkBrackets(str)` that checks the validity of brackets' positions. A parameter of the function is considered valid if it contains at least one `(` or `)` bracket.

Check only `(` and `)` brackets. The function returns the minimum number of brackets that must be added to make the string correct.

If the parameter is invalid (not a string or doesn't contain `(` and `)` brackets), the function returns `-1`.

Also, create tests (Chai and Mocha(bdd)) to check:

- minimum 5 incorrect cases with different data types
- minimum 10 correct cases

EXAMPLE

```
console.log(checkBrackets('1)()(()2(()')); // 2
console.log(checkBrackets(NaN)); // -1
```

SEE ALSO

Chai
Mocha-JavaScript test framework


ucode
Sprint 02 | Web Frontend SE > 43

Document

DOCUMENTATION

One of the attributes of Challenge Based Learning is documentation of the learning experience from challenge to solution. Throughout the challenge, you document your work using text, audio, video, images, photographs, and reflect on the process. These artifacts are useful for ongoing reflection, informative assessment, evidence of learning, portfolios, and telling the story of challenge. The end of each phase (Engage, Investigate, Act) of the challenge offers an opportunity to document the process.

Much of the deepest learning takes place by considering the process, thinking about one's own learning, analyzing ongoing relationships with the content and between concepts, interacting with other people, and developing a solution. During learning, documentation of all processes will help you analyze your work, approaches, thoughts, implementation options, code, etc. In the future, this will help you understand your mistakes, improve your work, and read the code.

At the learning stage, it is important to understand and do this, as this is one of the skills that you will need in your future job. Naturally, the documentation should not be voluminous, it should be drawn up in an accessible, logical, and connected form.

Helpful tools:

- Google Tools - a good way to journal your phases and processes:
  - Google Docs
  - Google Sheets
- Dropbox Paper - a tool for internally developing and creating documentation
- Git Wiki - a section for hosting documentation on Git-repository
- Haroopad - a markdown enabled document processor for creating web-friendly documents
- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio
- code commenting - source code clarification method. The syntax of comments is determined by the programming language
- and others to your taste

Keep in mind that the implementation of this stage will be checked by peers at the assessment!


ucode
Sprint 02 | Web Frontend SE > 44

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. This allows you to share your challenges, solutions, and reflections with local and global audiences.

During this stage, you will discover ways of getting external evaluation and feedback on your work. As a result, you will get the most out of the challenge, and get a better understanding of both your achievements and missteps.

To share your work, you can create:

- a text post, as a summary of your reflection
- charts, infographics or other ways to visualize your information
- a video, either of your work, or a reflection video
- an audio podcast. Record a story about your experience
- a photo report with a small post

Helpful tools:

- Canva - a good way to visualize your data
- QuickTime - an easy way to capture your screen, record video or audio

Examples of ways to share your experience:

- Facebook - create and share a post that will inspire your friends
- YouTube - upload an exciting video
- GitHub - share and describe your solution
- Telegraph - create a post that you can easily share on Telegram
- Instagram - share photos and stories from ucode. Don't forget to tag us :)

Share what you've learned and accomplished with your local community and the world. Use #ucode and #CBLWorld on social media.


ucode
Sprint 02 | Web Frontend SE > 45