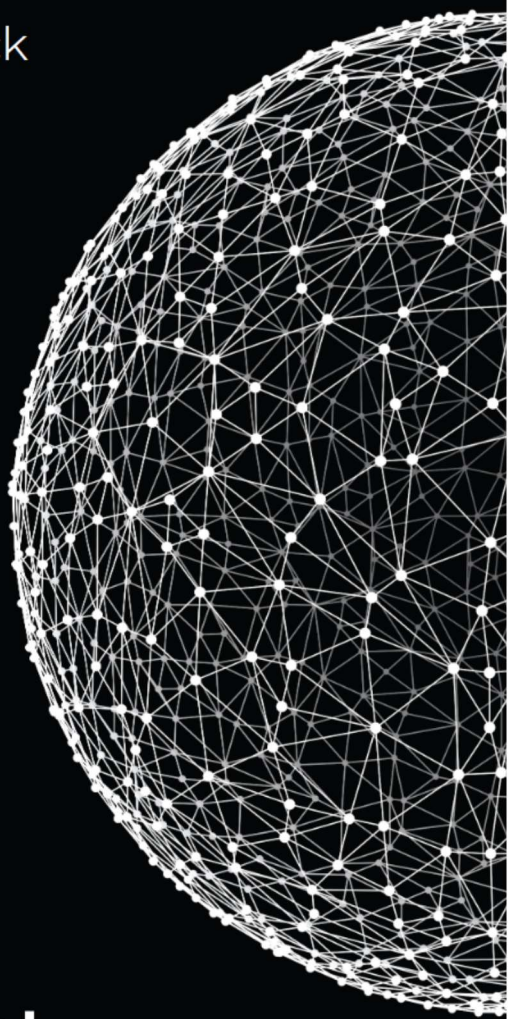
Sprint 06

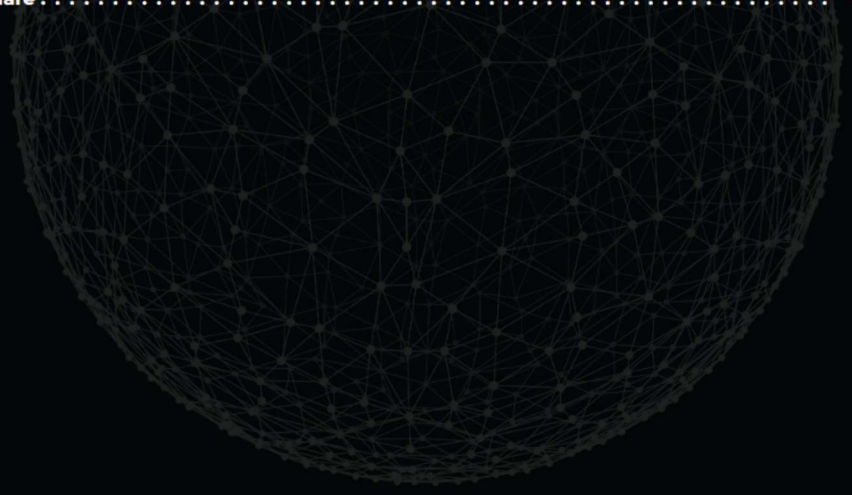
Half Marathon Full Stack

November 15, 2021



ucode connect

Contents



ucode connect

Engage

DESCRIPTION

Hopefully, you've enjoyed Node.js so far. Let's continue learning new topics in this Sprint.

Have you ever thought about how web sites can ask users for data? When you register on web sites, or buy tickets, you fill so-called webforms. They are elements on a web page that allow a user to enter data that is sent to a server for processing.
This time, you will learn more about them. And, since user interaction is quite error-prone, in this Sprint you will learn how to handle these errors.

Finally, it's time for you to get closer to the server side of things. You will get to practice starting a server for your pages.

Good luck!

BIG IDEA

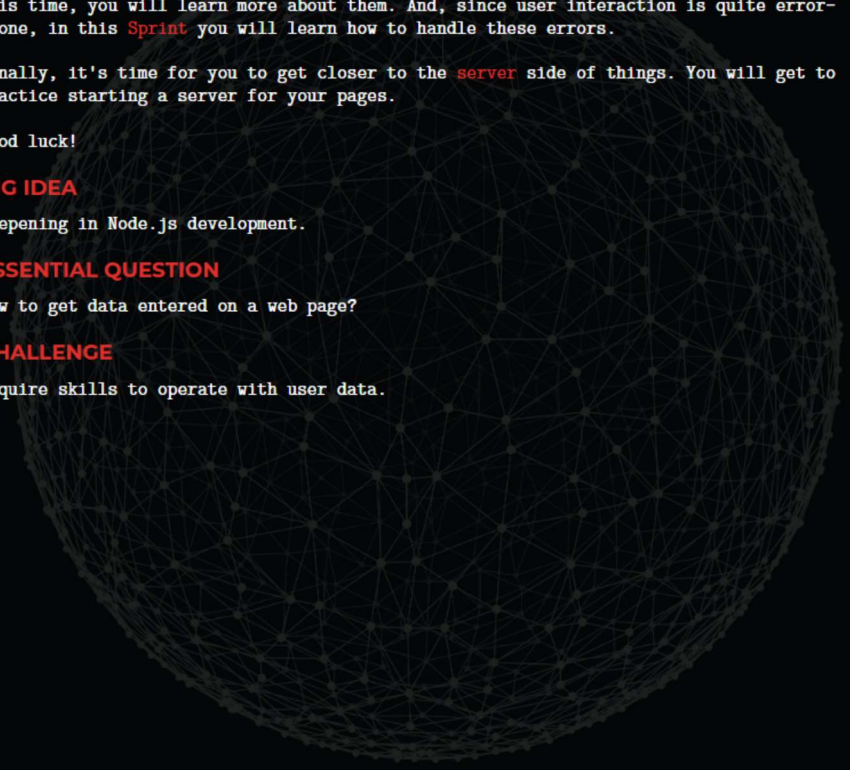
Deepening in Node.js development.

ESSENTIAL QUESTION

How to get data entered on a web page?

CHALLENGE

Acquire skills to operate with user data.



ucode connect

Sprint 06 | Half Marathon Full Stack **> 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 solutions, 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.

- Explain the purpose of object cloning.
- What is an exception?
- How to handle exceptions? What are they used for?
- How can an HTML form and a JS script interact?
- How to run a server in Node.js?

GUIDING ACTIVITIES

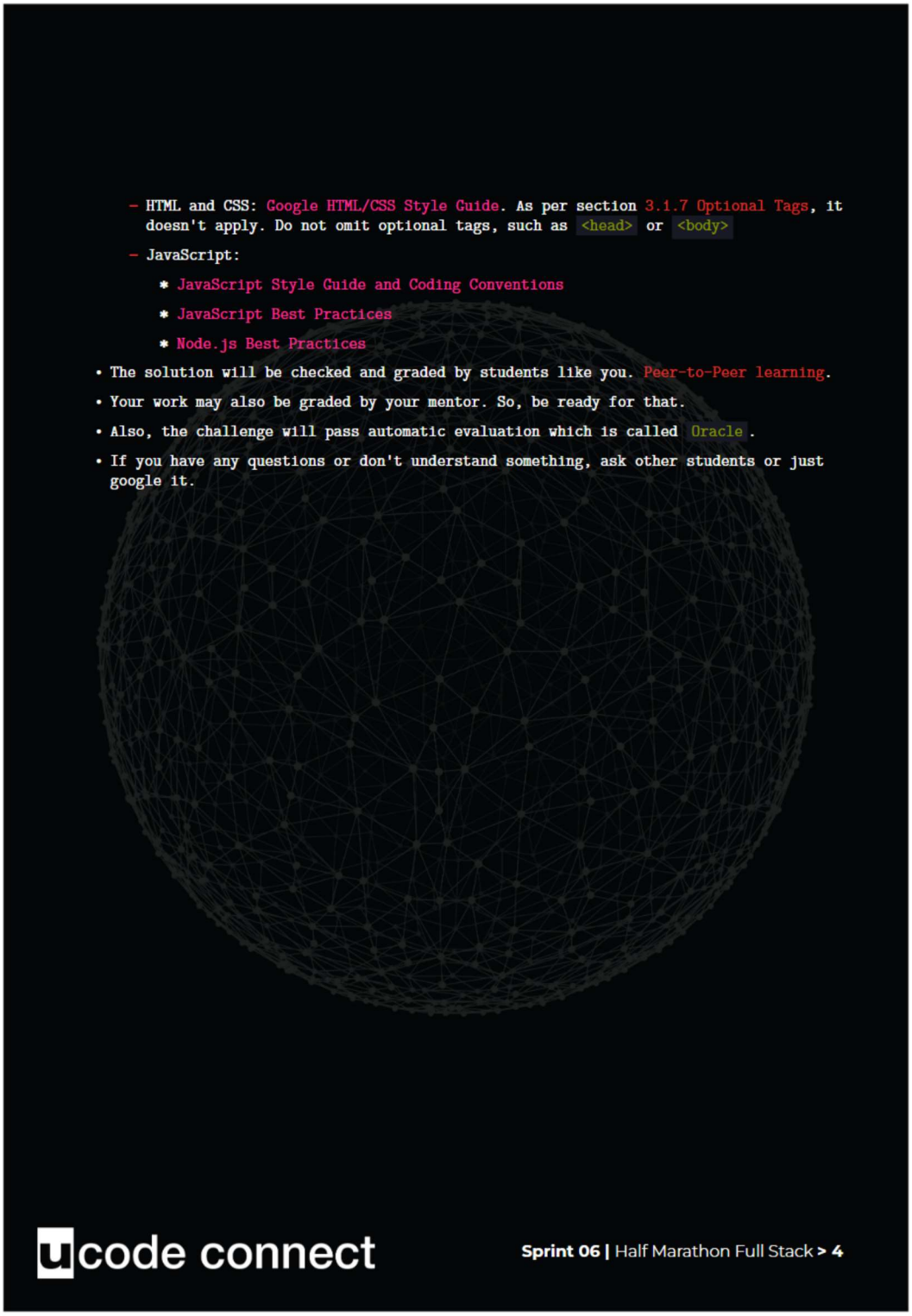
Complete the following activities. Don't forget that you have limited time to overcome the challenge. Use it wisely. Distribute tasks correctly.

- Research how to structure an HTML form.
- Find on the internet some HTML registration form templates that you like.
- Discuss with your peers why magic methods are called magic.
- Maybe, it is time to create your personal cheat sheet with all the basic concepts that you have learned so far.
- Try to run your first Node.js server using official example.
- Clone your git repository, issued on the challenge page in the LMS.
- Try to implement your thoughts in code.

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.
- All tasks are divided into Act Basic and Act Advanced. This means that the complexity of the tasks increases gradually. Try to complete all tasks to get maximum points and more knowledge.
- Analyze all the information you have collected during the preparation stages.
- 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 a reason to fail the challenge.
- 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
* Node.js Best Practices
• The solution will be checked and graded by students like you. Peer-to-Peer learning.
• Your work may also be graded by your mentor. So, be ready for that.
• Also, the challenge will pass automatic evaluation which is called Oracle.
• If you have any questions or don't understand something, ask other students or just google it.
ucode connect
Sprint 06 | Half Marathon Full Stack > 4

Act Basic: Task 00

NAME

LinkedList

DIRECTORY

`t00_linkedlist/`

SUBMIT

`LLData.js, LList.js`

ALLOWED

JS

DESCRIPTION

Let's refresh our memory!
Create a linked list item class called `LLData`, and a linked list class called `LList`.

Properties of `LLData` include:

- data
- next - reference to the next element or null

Methods of `LList` (`the iterable protocol`) include:

- `getFirst()`
- `getLast()`
- `add(value)` - creates a new LLData with given value and appends it to the list
- `addFromArray(arrayOfData)` - appends all values (as LLData items) of the given array to the list
- `remove(value)` - deletes the first element that equals the given value
- `removeAll(value)` - deletes all elements that are equal to the given value
- `contains(value)` - checks whether a value is present in the LList
- `clear()` - clears LList
- `count()` - gets number of items in the LList
- `toString()` - prints all element values, separated by a `,` (comma)
- `getIterator()` - returns an iterator for element values (implement `Iterator`)
- `filter(callback)` - (works similarly to an array's `filter()` method) returns a new LList, all elements of which would return `true` given the provided callback function

The script in the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

ucode connect

Sprint 06 | Half Marathon Full Stack **> 5**

SYNOPSIS

```
/*
  Task name: LinkedList
*/

const {LList} = require("./LList");

const list = new LList();

list.addFromArray([100, 1, 2, 3, 100, 4, 5, 100]);

list.add(10);

const onlySmallList = list.filter((data) => {
  return data < 18;
});

let sumOfAll = 0;

for (const data of list) {
  sumOfAll += data;
}

console.log([...list]); // [ 100, 1,   2,  3, 100, 4, 5, 100, 10 ]
console.log([...onlySmallList]); // [ 1, 2, 3, 4, 5, 10 ]
console.log(sumOfAll); // 325
console.log(list.contains(10)); // true
console.log(list.contains(22)); // false

list.clear()

console.log([...list]); // []
```

CONSOLE OUTPUT

```
>node t00/test.js
[
  100, 1,   2,  3, 100,
    4, 5, 100, 10
]
[ 1, 2, 3, 4, 5, 10 ]
325
true
false
[]
```

SEE ALSO

Linked list

ucode connect

Sprint 06 | Half Marathon Full Stack **> 6**

Act Basic: Task 01

NAME

Object to string

DIRECTORY

`t01_object_to_string/`

SUBMIT

`Avenger.js`

ALLOWED

JS

LEGEND

"There was an idea to bring together a group of remarkable people,
to see if we could become something more."

DESCRIPTION

So, let's create some remarkable people.

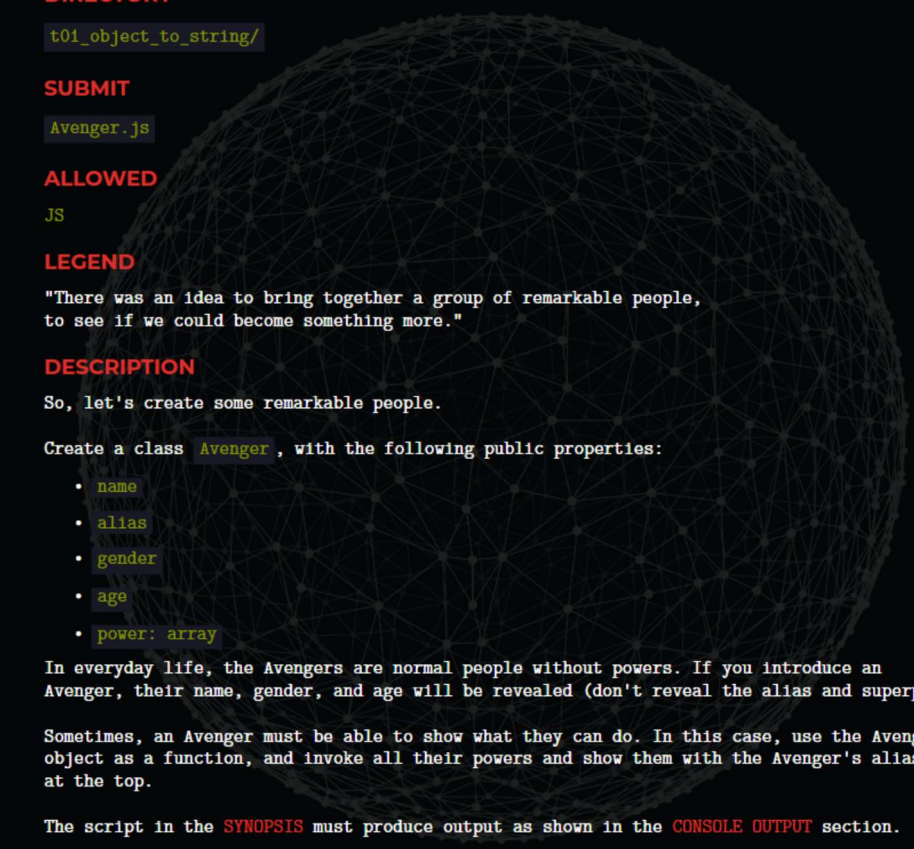
Create a class `Avenger`, with the following public properties:

- `name`
- `alias`
- `gender`
- `age`
- `power: array`

In everyday life, the Avengers are normal people without powers. If you introduce an Avenger, their name, gender, and age will be revealed (don't reveal the alias and superpowers!).

Sometimes, an Avenger must be able to show what they can do. In this case, use the Avenger object as a function, and invoke all their powers and show them with the Avenger's alias at the top.

The script in the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.



ucode connect

Sprint 06 | Half Marathon Full Stack **> 7**

SYNOPSIS

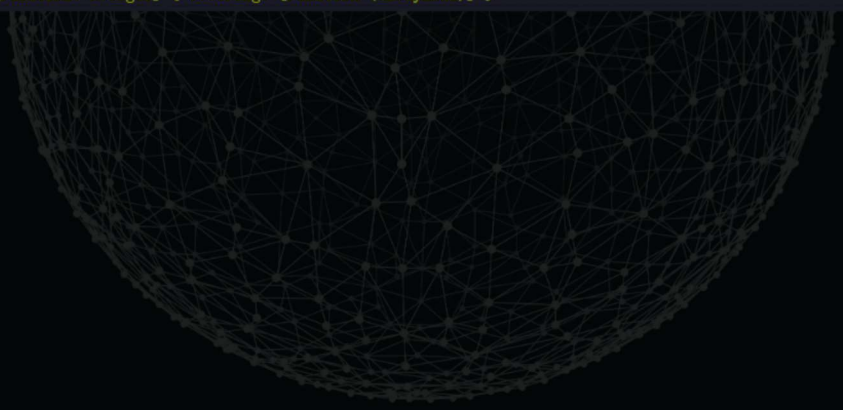
```
/*
  Task name: Object to string
*/

const {Avenger} = require("./Avenger");

const stark = new Avenger({
  name: 'Tony Stark',
  alias: 'Iron Man',
  gender: 'man',
  age: 38,
  powers: ["intelligence", "durability", "magnetism"]
})

const natasha = new Avenger({
  name: 'Natasha Romanoff',
  alias: 'Black Widow',
  gender: 'woman',
  age: 35,
  powers: ["agility", "martial arts"]
})

const examine = (avenger) => {
  console.count('Avenger');
  console.group('*** Avenger introduced ***');
  console.log(avenger.toString());
  console.groupEnd();
  console.group('*** Avenger called ***');
  console.log(avenger());
  console.groupEnd();
  console.group('*** Avenger\'s internals ***');
  console.log(avenger, '\n');
  console.groupEnd();
}

examine(stark);

examine(natasha);
```

ucode connect

Sprint 06 | Half Marathon Full Stack **> 8**

CONSOLE OUTPUT

```
>node t01/test.js
Avenger: 1
*** Avenger introduced ***
  name: Tony Stark
  gender: man
  age: 38
*** Avenger called ***
  IRON MAN
  intelligence
  durability
  magnetism
*** Avenger's internals ***
  [Function: Avenger] { toString: [Function (anonymous)] }

Avenger: 2
*** Avenger introduced ***
  name: Natasha Romanoff
  gender: woman
  age: 35
*** Avenger called ***
  BLACK WIDOW
  agility
  martial arts
*** Avenger's internals ***
  [Function: Avenger] { toString: [Function (anonymous)] }
```



ucode connect

Sprint 06 | Half Marathon Full Stack **> 9**

Act Basic: Task 02

NAME

Clone the Avengers

DIRECTORY

`t02_clone_the_avengers/`

SUBMIT

`Team.js, Avenger.js`

ALLOWED

JS

LEGEND

"I get emails from a raccoon, so nothing sounds crazy."

DESCRIPTION

There are so many villains in this universe who want to destroy the Avengers. It will be great to have an additional team or two.

Use the class `Avenger` from the previous task and add an `hp` property to it.

Create a class `Team` with the following properties and methods:

- `id`
- `avengers` - an array of `Avenger` objects
- `battle(damage): void` - method that subtracts `damage` from `hp` of each team member
- `calculateLosses(clonedTeam): void` - compares two teams and prints the number of losses in the current team

Create a function `battle(damage)` in which the value of the `hp` variable will be decreased by the given `damage` amount. If the `hp` of an Avenger is 0 or less, then they should be deleted from the Team. In the test code, we clone the team before a battle in order to calculate the losses afterwards. Implement the function `calculateLosses(clonedTeam)`, which prints out information about how many Avengers we have lost in the last battle.

The script in the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

ucode connect

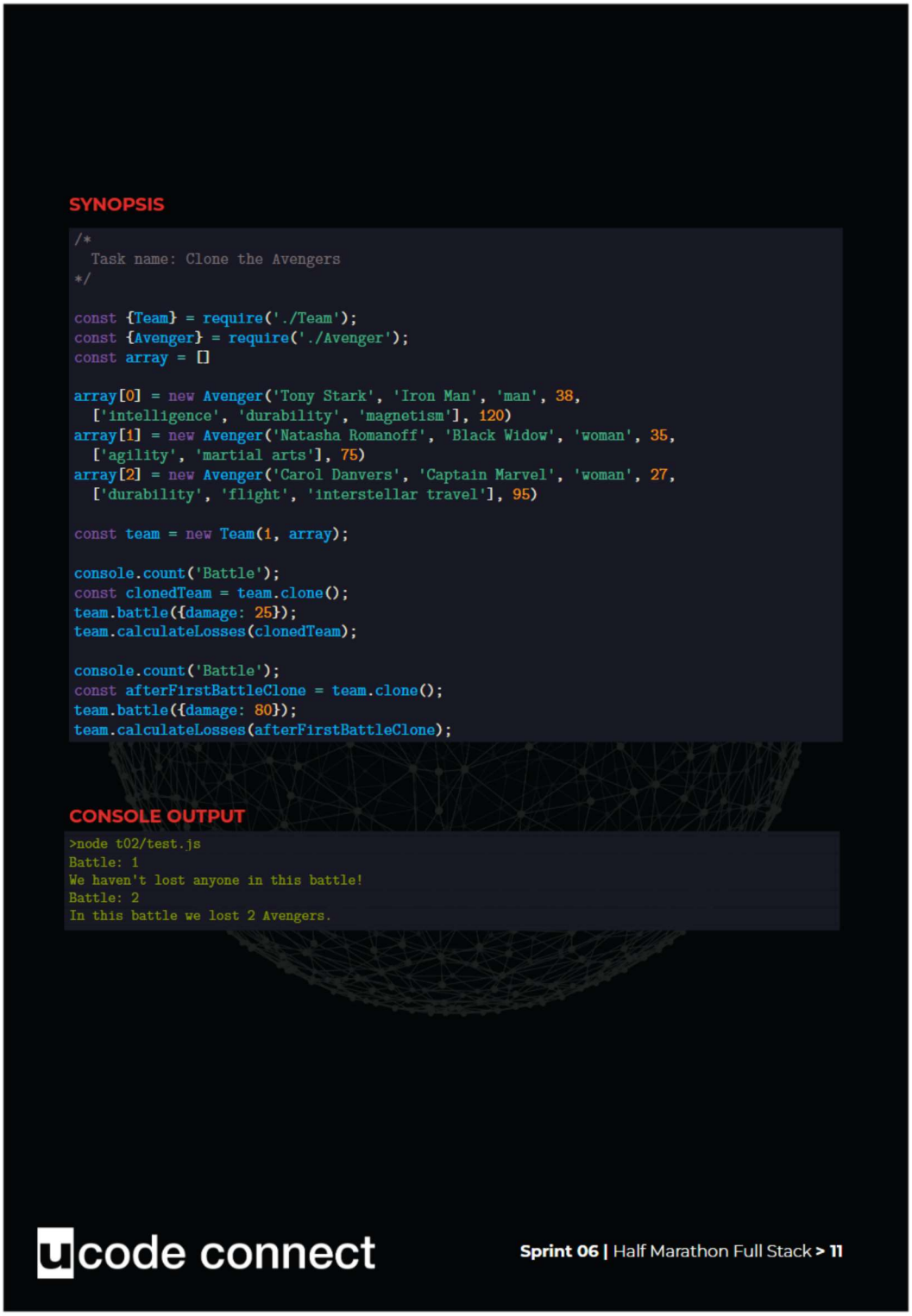
Sprint 06 | Half Marathon Full Stack **> 10**

SYNOPSIS

```
/*
  Task name: Clone the Avengers
*/

const {Team} = require('./Team');
const {Avenger} = require('./Avenger');
const array = []

array[0] = new Avenger('Tony Stark', 'Iron Man', 'man', 38,
  ['intelligence', 'durability', 'magnetism'], 120)
array[1] = new Avenger('Natasha Romanoff', 'Black Widow', 'woman', 35,
  ['agility', 'martial arts'], 75)
array[2] = new Avenger('Carol Danvers', 'Captain Marvel', 'woman', 27,
  ['durability', 'flight', 'interstellar travel'], 95)

const team = new Team(1, array);

console.count('Battle');
const clonedTeam = team.clone();
team.battle({damage: 25});
team.calculateLosses(clonedTeam);

console.count('Battle');
const afterFirstBattleClone = team.clone();
team.battle({damage: 80});
team.calculateLosses(afterFirstBattleClone);
```

CONSOLE OUTPUT

```
>node t02/test.js
Battle: 1
We haven't lost anyone in this battle!
Battle: 2
In this battle we lost 2 Avengers.
```



Act Basic: Task 03

NAME

Try, catch

DIRECTORY

`t03_try_catch/`

SUBMIT

`eat-exception.js, product.js, ingestion.js`

ALLOWED

JS

BEFORE YOU BEGIN

Get a basic understanding of an exceptions concept.

LEGEND

"[arguing over which Avenger is strong enough to wield the Infinity Gauntlet]
Thor: Do you know what is coursing through my veins right now?
James Rhodes: Cheez Whiz?"

DESCRIPTION

Since we are helping the Avengers, there is another character who needs help.
Let's help Thor lose weight!

Create an exception class named `EatException` with a message `No more junk food, dumpling`.
Create a class `Product` with:

- `name`
- `kcal_per_portion`

You should determine whether it is junk food or not by the amount of calories per portion.
If the amount is more than 200 - it's junk food. Otherwise - healthy food.

Create a class `Ingestion` with:

- `id`
- `meal_type: array (breakfast, lunch, dinner)`
- `day_of_diet`
- `products: array`
- `getFromFridge(product): void - if it's junk food - throw the exception`

Test your code with the script given in the SYNOPSIS. The CONSOLE OUTPUT will differ for every call, because the calories amount is a random number, but the logic must work the same. Test with more cases.

ucode connect

Sprint 06 | Half Marathon Full Stack > **12**

SYNOPSIS

```
/*
  Task name: Try, catch
*/

const {Product} = require("./product");
const {Ingestion} = require("./ingestion");
const productNames = [
  'Nutella',
  'Chicken',
  'Coca-Cola',
  'Biscuit',
  'Brocolli',
  'Tomatoes',
  'Apple',
  'Potato',
  'Pizza',
  'Beer'
];

const randomInt = (min, max) => {
  return min + Math.floor((max - min) * Math.random());
}

const stock = new Ingestion('breakfast', 1);

productNames.forEach(name => {
  stock.setProduct(new Product(name, randomInt(40, 500)))
})

productNames.forEach(productName => {
  console.log(`***\nGetting ${productName} that has`,
  `${stock.getProductInfo(productName).kcal} calories.`)
  try {
    stock.getFromFridge(productName);
    console.log(`You're doing great, ${productName} is good!`)
  } catch (error) {
    console.log(`Caught exception: ${error.message}!`,
    `Throw ${productName} away!`)
  }
})
```

ucode connect

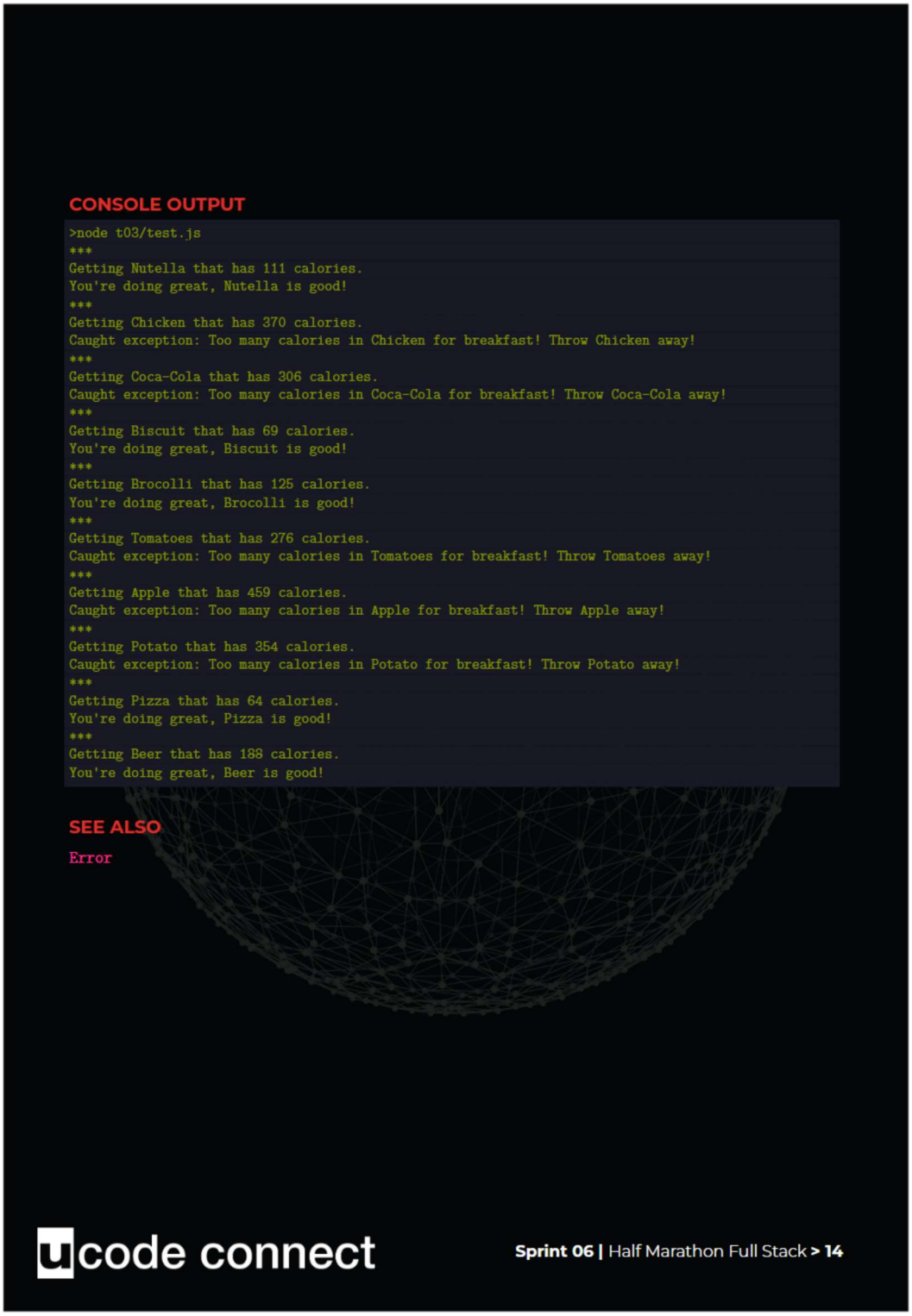
Sprint 06 | Half Marathon Full Stack **> 13**

CONSOLE OUTPUT

```
>node t03/test.js
***
Getting Nutella that has 111 calories.
You're doing great, Nutella is good!
***
Getting Chicken that has 370 calories.
Caught exception: Too many calories in Chicken for breakfast! Throw Chicken away!
***
Getting Coca-Cola that has 306 calories.
Caught exception: Too many calories in Coca-Cola for breakfast! Throw Coca-Cola away!
***
Getting Biscuit that has 69 calories.
You're doing great, Biscuit is good!
***
Getting Brocolli that has 125 calories.
You're doing great, Brocolli is good!
***
Getting Tomatoes that has 276 calories.
Caught exception: Too many calories in Tomatoes for breakfast! Throw Tomatoes away!
***
Getting Apple that has 459 calories.
Caught exception: Too many calories in Apple for breakfast! Throw Apple away!
***
Getting Potato that has 354 calories.
Caught exception: Too many calories in Potato for breakfast! Throw Potato away!
***
Getting Pizza that has 64 calories.
You're doing great, Pizza is good!
***
Getting Beer that has 188 calories.
You're doing great, Beer is good!
```

SEE ALSO

Error



Act Basic: Task 04

NAME

What about forms?

DIRECTORY

`t04_what_about_forms/`

SUBMIT

`index.html, server.js, script.js`

ALLOWED

JS, HTML

LEGEND

"The hardest choices require the strongest wills."

DESCRIPTION

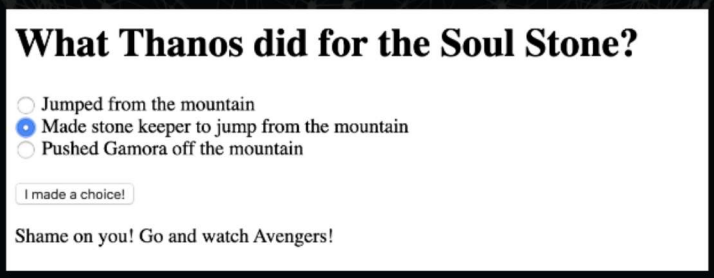
Do you remember what Thanos did to get the Soul Stone?

Let's create a little quiz. Ask any question on the web page which is related to the Avengers. The answers must be radio buttons.

There must a relevant reaction to submitting an answer (correct/incorrect).

Run a server for this quiz. Code for the server must be in the `server.js` file. Put your code for checking answers in the `script.js` file.

EXAMPLE



SEE ALSO

HTTP module

ucode connect

Sprint 06 | Half Marathon Full Stack **> 15**

Act Basic: Task 05

NAME

A new set

DIRECTORY

`t05_a_new_set/`

SUBMIT

`index.html, style.css, server.js, [optionally:] *.js`

ALLOWED

JS, HTML, CSS

LEGEND

"Natasha Romanoff: I used to have nothing. Then I got this. This job... this family. And I was... I was better because of it. And even though... they are gone... I'm still trying to be better."

DESCRIPTION

After Thanos took over the glove and snapped his fingers, there was a shortage of the Avengers. In this regard, the Avengers team is open to recruitment.

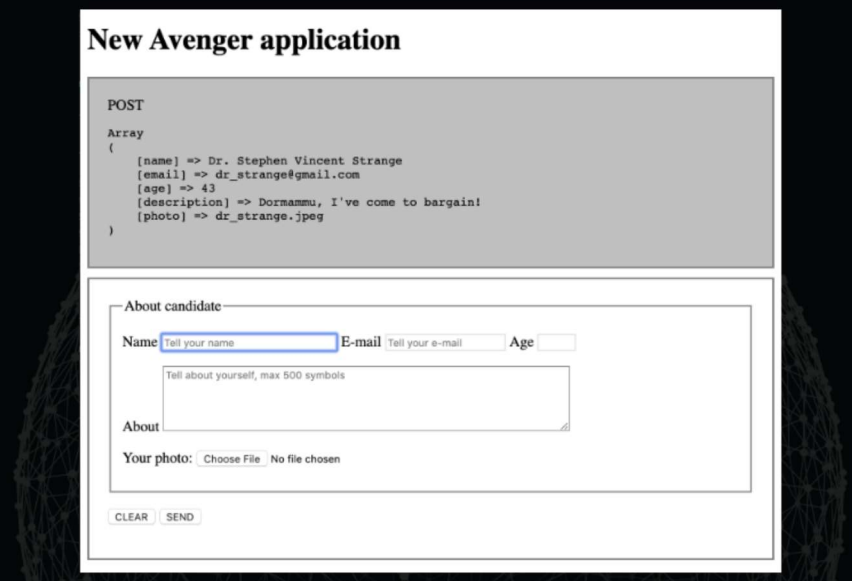
Create a form for an applying candidate. The candidate must enter name, e-mail, age, write something about themselves and upload a photo. The form in any way shouldn't be empty.
See the EXAMPLE.

Code for your server must be in server.js. If needed, put additional code for your page in a separate .js file.

ucode connect

Sprint 06 | Half Marathon Full Stack **> 16**

EXAMPLE



ucode connect

Sprint 06 | Half Marathon Full Stack > 17

Act Advanced: Task 06

NAME

Quantum

DIRECTORY

`t06_quantum/`

SUBMIT

`quantum.js, normal.js`

ALLOWED

JS

LEGEND

"Last night, we powered up the tunnel for the first time.
It was overloaded and it shut down. But for a split second, the doorway to the quantum realm was opened."

DESCRIPTION

Remember how Scott Lang got stuck in quantum space? And who can remember how much time he spends there? We know that there are no such concepts like time and space in the quantum dimension. Imagine that you fell into a quantum space on 1 of January in 1939 (we let you guess why exactly this year). Let's assume that 7 normal years are considered as 1 year in the quantum space.

So, create a function `calculateTime(): datetime` in the `normal` module, which returns how much time has passed since 1939 in normal space in format «years-months-days».
The function `calculateTime(): array` in the `quantum` module, which returns the same time by the standards of quantum space. We know that there are no such concepts as time and space in the quantum dimension. It's not as easy as it seems. Print your results to the console.

The script in the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

ucode connect

Sprint 06 | Half Marathon Full Stack **> 18**

SYNOPSIS

```
/*
  Task name: Quantum
*/

const normal = require('./normal');
const quantum = require('./quantum');

const time = normal.calculateTime();

console.log(`In real life you were absent for ${time.years()} years, ${time.months()}
↪  months, ${time.days()} days.`)

const quantumTime = quantum.calculateTime();

console.log(`In quantum space you were absent for ${quantumTime[0]} years,
↪  ${quantumTime[1]} months, ${quantumTime[2]} days.`)
```

CONSOLE OUTPUT

```
>node t06/test.js
In real life you were absent for 82 years, 0 months, 18 days.
In quantum space you were absent for 11 years, 5 months, 18 days.
```

SEE ALSO

Date

Act Advanced: Task 07

NAME

Go web!

DIRECTORY

`t07_go_web/`

SUBMIT

`index.js, normal-router.js, quantum-router.js, views/*.ejs`

ALLOWED

JS, HTML

LEGEND

"Hi...uhhh...is anyone home? This is Scott Lang.
We met a few years ago... at the airport...in Germany...I got really big."

DESCRIPTION

Were you able to cope with the previous task? Great, because we have another one.

Now you should display the same information on the web page. Your `index.js` must route you to either the normal or quantum information web page. On the page normal display how much time has passed since 1939 in normal space. And on the page quantum display the same time by the standards of quantum space.

SEE ALSO

EJS

ucode connect

Sprint 06 | Half Marathon Full Stack **> 20**

Act Advanced: Task 08

NAME

Info

DIRECTORY

`t08_info/`

SUBMIT

`index.js`

ALLOWED

JS

LEGEND

"Heilmdail: Be warned, I shall uphold my sacred oath to protect this realm as its gatekeeper. If your return threatens the safety of Asgard, my gate will remain shut and you will be left to perish on the cold waste of Jotunheim."

DESCRIPTION

Output to the console the following information:

- a name of file of the executed script
- arguments passed to the script
- IP address of the server
- a name of host that invokes the current script
- a name and a version of the information protocol
- a query method
- User-Agent information
- IP address of the client
- a list of parameters passed by URL

SEE ALSO

Process

ucode connect

Sprint 06 | Half Marathon Full Stack **> 21**

Share

PUBLISHING

Last but not least, the final stage of your work is to publish it. It 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 task and better understand 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 (macOS)
- ScreenToGif - screen, webcam, and sketchboard recorder with an integrated editor (Windows)

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
- 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.