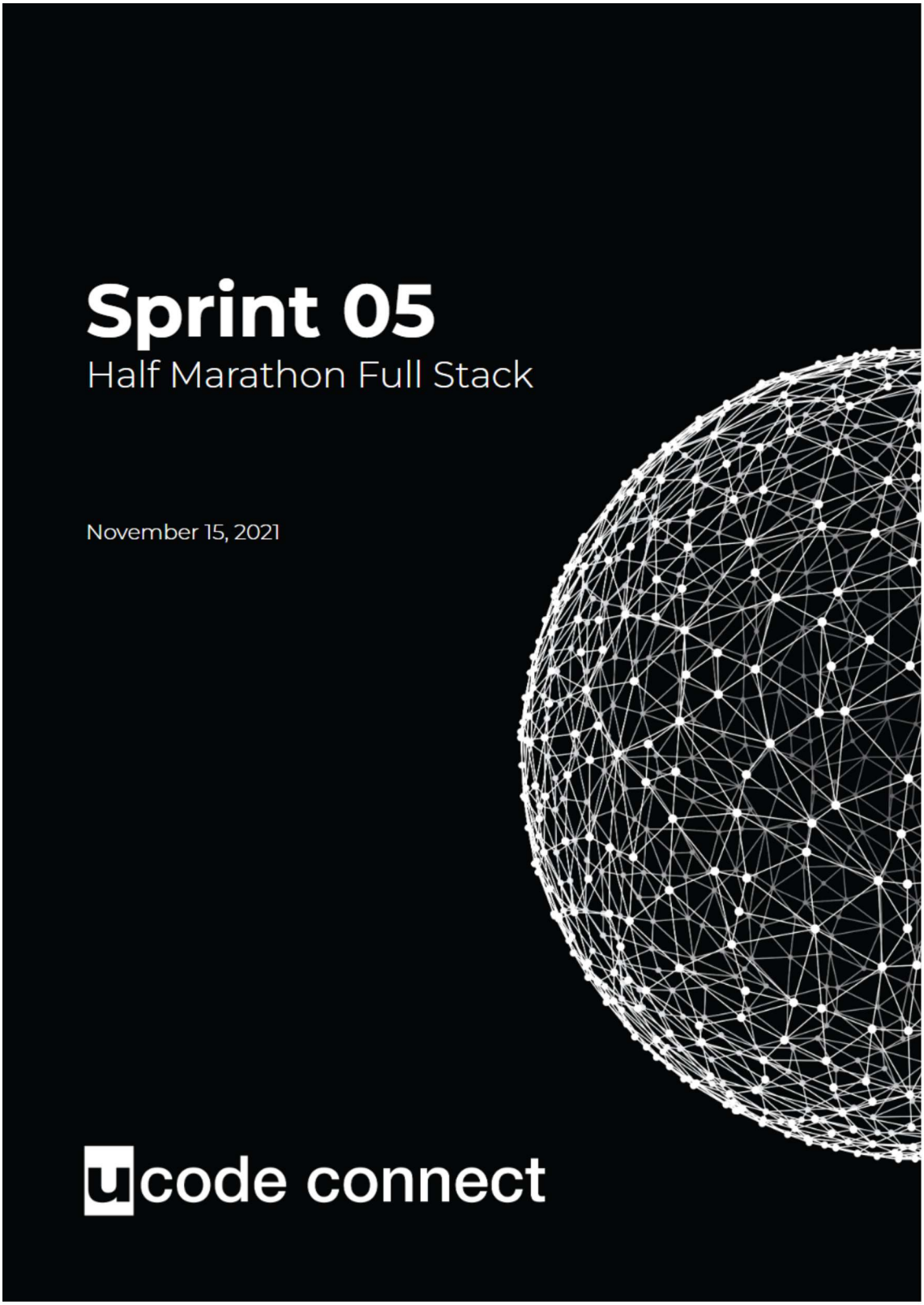

Sprint 05
Half Marathon Full Stack
November 15, 2021
ucode connect

Contents

ucode connect

Engage

DESCRIPTION

Hello again!

Get ready to meet Node.js, one of the top 5 server-side programming languages. Well, actually, Node.js is a runtime environment for running Javascript, but nevertheless... So far, you have only worked with the frontend, but there's much more to web than that! A web service is like a restaurant: frontend is the dining area, and backend is the kitchen. The time has come for you to look *under the hood*.

You are already familiar with the basics of JavaScript, so Node.js (which runs the V8 JavaScript engine) offers a particularly smooth transition to the server side.

New day, new knowledge and skills!

BIG IDEA

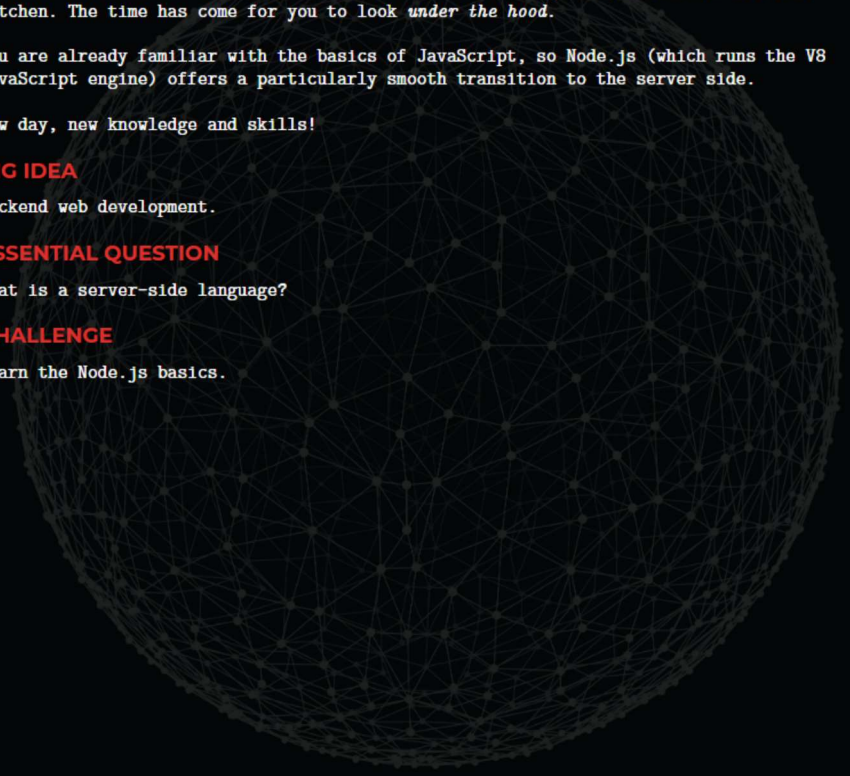
Backend web development.

ESSENTIAL QUESTION

What is a server-side language?

CHALLENGE

Learn the Node.js basics.



ucode connect

Sprint 05 | 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.

- What is Node.js?
- What problems can Node.js solve?
- How to access non-static and static properties within class methods?
- What is the keyword `static` used for?
- What is function overriding in JavaScript?
- What is a module?
- What kinds of modules does Node.js support?

GUIDING ACTIVITIES

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

- Watch this video.
- Prepare the environment for local development. Install Node.js if you do not have it yet:
  - if you work at ucode iMac - install Node.js with ubrew:
    1) clone repository https://gitlab.ucode.world:8443/services/ubrew
    2) install ubrew according to the instructions in the README.md file
    3) install node with command `brew install node`
  - if you work from some other place - download the latest Node.js version from here and install it.
- Read about the advantages and disadvantages of Node.js among other server languages.
- Find the most informative resources with Node.js documentation.
  We advise you to visit nodejs.org.
- Clone your git repository issued on the challenge page in the LMS.
- Proceed with tasks.

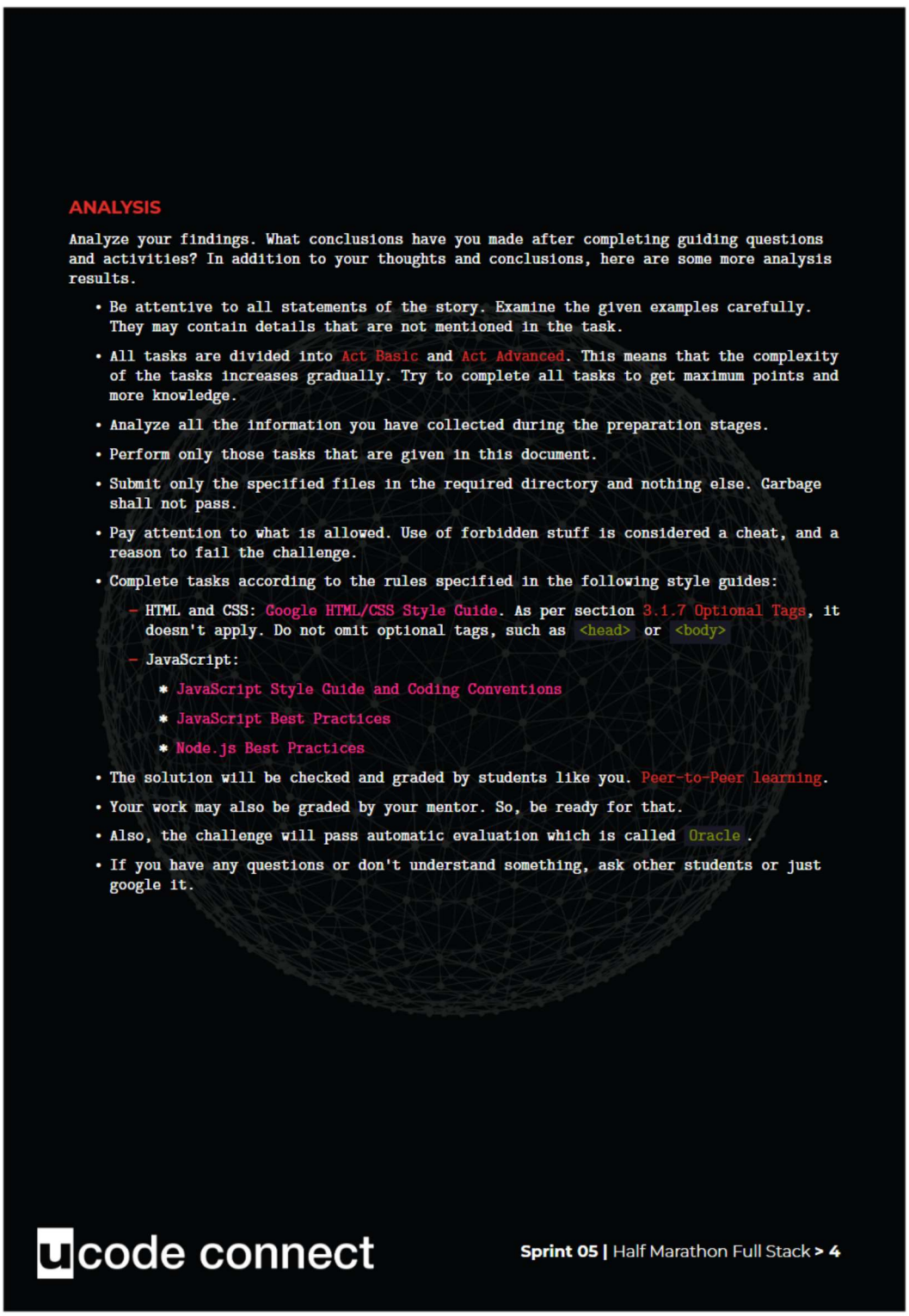
ucode connect

Sprint 05 | Half Marathon Full Stack **> 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.
- 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.



Act Basic: Task 00

NAME

Good morning!

DIRECTORY

`t00_good_morning/`

SUBMIT

`index.js`

ALLOWED

JS

DESCRIPTION

Create a script that outputs a message. See the CONSOLE OUTPUT for an illustration of how your script must work.

CONSOLE OUTPUT

```
>node t00/index.js
Good morning!
It's 7 A.M.
The weather in Malibu is 72 degrees with scattered clouds.
The surf conditions are fair, high tide will be at 10:52 a.m.
```

SEE ALSO

Node.js

ucode connect

Sprint 05 | Half Marathon Full Stack **> 5**

Act Basic: Task 01

NAME

Iron Vars

DIRECTORY

`t01_iron_vars/`

SUBMIT

`index.js`

ALLOWED

JS

BEFORE YOU BEGIN

Get a basic understanding about Node.js modules.

LEGEND

138. 138 combat missions.
That's how many I've flown, Tony.
Every one of them could've been my last, but I flew 'em.
'Cause the fight needed to be fought.

DESCRIPTION

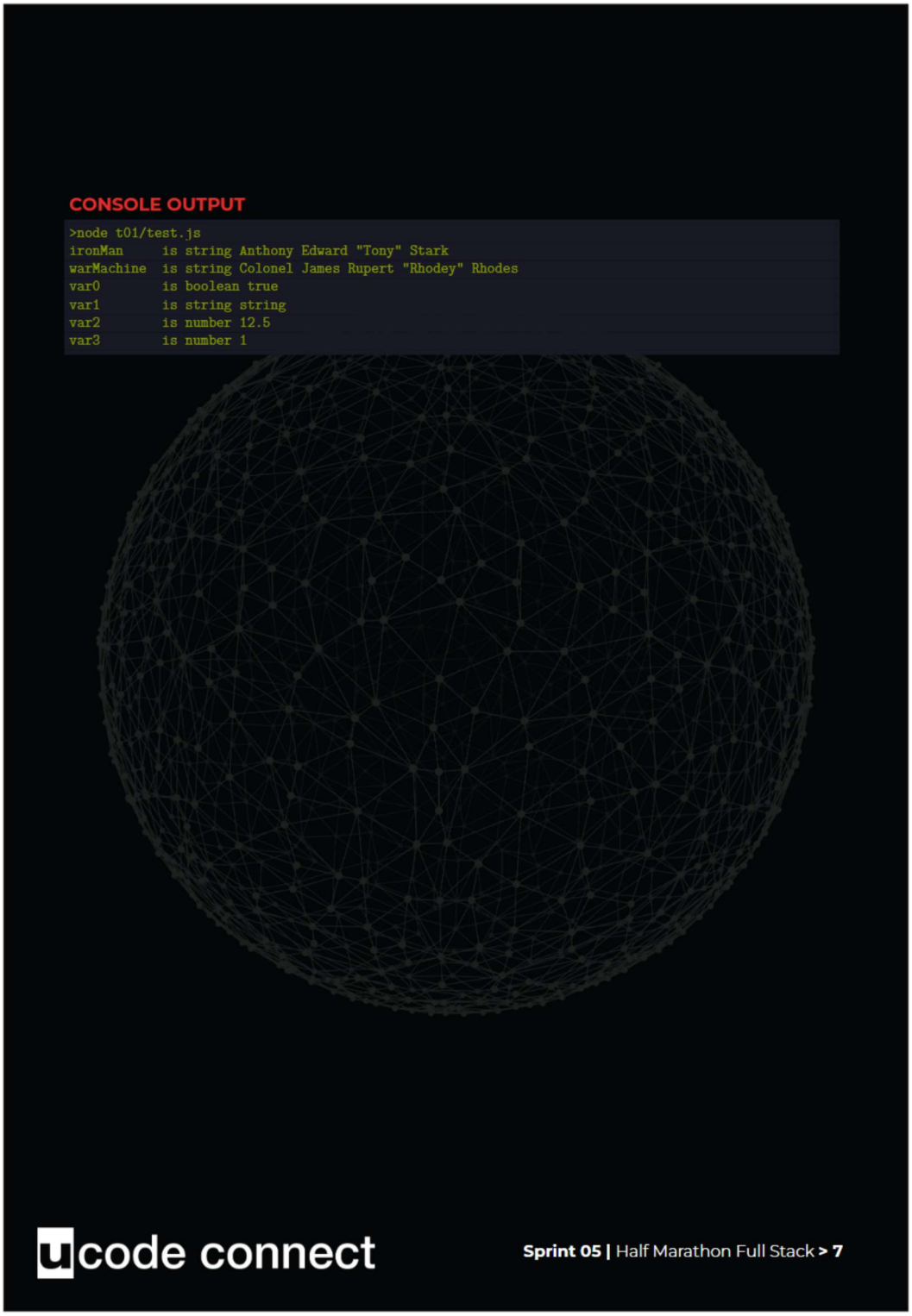
Create a script with declared and initialized variables of the available scalar types.
The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```
/*
  Task name: Iron Vars
*/

const i = require('./index');

console.log(`ironMan     is ${typeof i.ironMan} ${i.ironMan}`);
console.log(`warMachine  is ${typeof i.warMachine} ${i.warMachine}`);
console.log(`var0        is ${typeof i.var0} ${i.var0}`);
console.log(`var1        is ${typeof i.var1} ${i.var1}`);
console.log(`var2        is ${typeof i.var2} ${i.var2}`);
console.log(`var3        is ${typeof i.var3} ${i.var3}`);
```

ucode connect

Sprint 05 | Half Marathon Full Stack **> 6**


CONSOLE OUTPUT
>node t01/test.js
ironMan is string Anthony Edward "Tony" Stark
warMachine is string Colonel James Rupert "Rhodey" Rhodes
var0 is boolean true
var1 is string string
var2 is number 12.5
var3 is number 1
ucode connect
Sprint 05 | Half Marathon Full Stack > 7

Act Basic: Task 02

NAME

Range

DIRECTORY

`t02_range/`

SUBMIT

`index.js`

ALLOWED

JS

BEFORE YOU BEGIN

Figure out how to work with default parameters.

DESCRIPTION

Create a function that takes two numbers as parameters. These two numbers are the start and end of an inclusive range. The function prints information for all the numbers in that range to the console.

Information includes whether a number is

- divisible by 2
- divisible by 3
- divisible by 10

The default function range is `1 - 60`.

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

SYNOPSIS

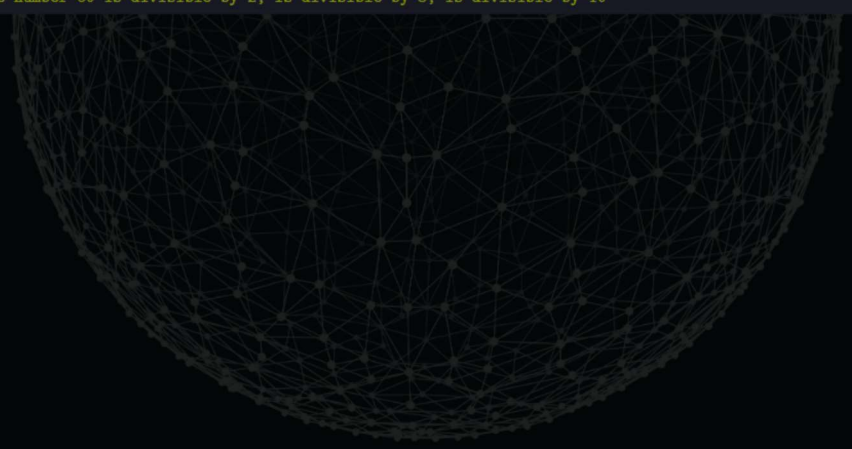
```
/*
  Task name: Range
*/

const i = require('./index');

console.log('*** Range is 3 - 7 checkDivision(3,7) ***');
i.checkDivision(3, 7);

console.log('\n*** Range is 58 - ... checkDivision(58) ***');
i.checkDivision(58);

console.log('\n*** Range is ... - ... checkDivision() ***');
i.checkDivision();
```

ucode connect

Sprint 05 | Half Marathon Full Stack **> 8**

CONSOLE OUTPUT

```
>node t02/test.js
*** Range is 3 - 7 checkDivision(3,7) ***
The number 3 is divisible by 3
The number 4 is divisible by 2
The number 5 -
The number 6 is divisible by 2, is divisible by 3
The number 7 -

*** Range is 58 - ... checkDivision(58) ***
The number 58 is divisible by 2
The number 59 -
The number 60 is divisible by 2, is divisible by 3, is divisible by 10

*** Range is ... - ... checkDivision() ***
The number 1 -
The number 2 is divisible by 2
The number 3 is divisible by 3
The number 4 is divisible by 2
The number 5 -
..
The number 60 is divisible by 2, is divisible by 3, is divisible by 10
```



ucode connect

Sprint 05 | Half Marathon Full Stack **> 9**

Act Basic: Task 03

NAME

First upper

DIRECTORY

`t03_first_upper/`

SUBMIT

`index.js`

ALLOWED

JS

BEFORE YOU BEGIN

Find out:

- What `trim()` is
- What `slice()` is
- What `love()` is

DESCRIPTION

Create a function `firstUpper()` that:

- takes a string as input
- returns that the string with only the first letter capitalized, or an empty string

Remove all whitespaces at the beginning and at the end of the string.

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

SYNOPSIS

```
firstUpper (string) : string
```

```
/*
  Task name: First upper
*/

const i = require('./index');

console.log(`"testing String": ${i.firstUpper("testing String")}`);
console.log(`"   testing   String": ${i.firstUpper("   testing   String")}`);
console.log(`"07": ${i.firstUpper("07")}`);
console.log(`"": ${i.firstUpper("")}`);
console.log(`null: ${i.firstUpper(null)}`);
```

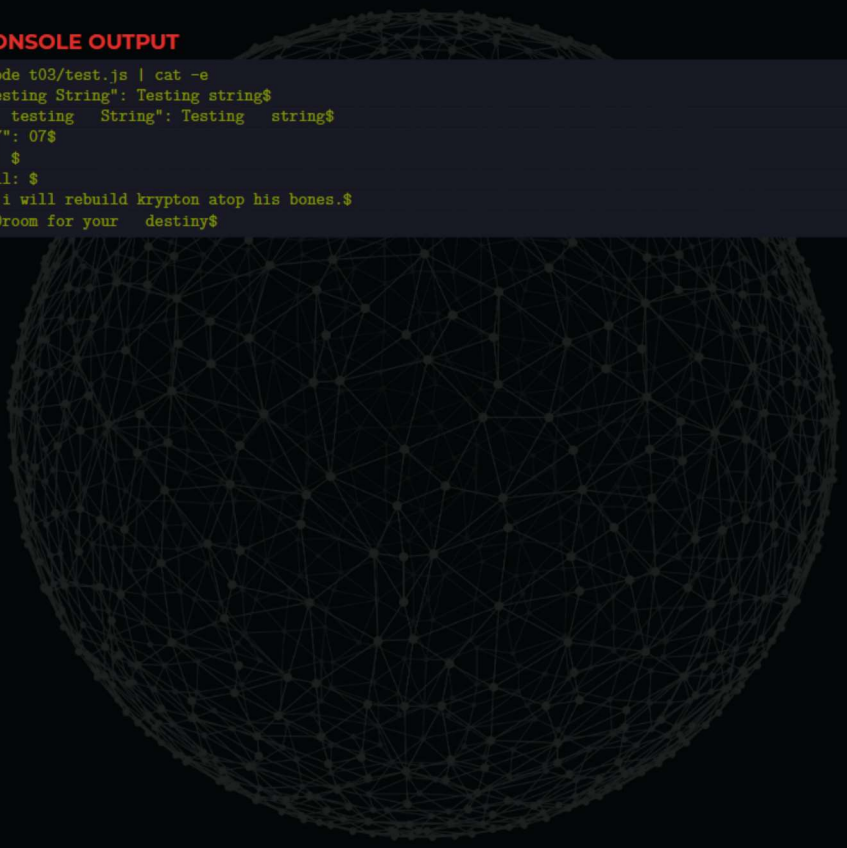
ucode connect

Sprint 05 | Half Marathon Full Stack **> 10**

```
console.log(1.firstUpper('   ...I Will Rebuild Krypton Atop His Bones.'));
console.log(1.firstUpper(' 300room FOR yoUr   DESTiny   '));
```

CONSOLE OUTPUT

```
>node t03/test.js | cat -e
"testing String": Testing string$
"   testing   String": Testing   string$
"07": 07$
"": $
null: $
...i will rebuild krypton atop his bones.$
300room for your   destiny$
```



ucode connect

Sprint 05 | Half Marathon Full Stack > **11**

Act Basic: Task 04

NAME

Anonymous

DIRECTORY

`t04_anonymous/`

SUBMIT

`index.js`

ALLOWED

JS

LEGEND

"Some people call me a terrorist. I consider myself a teacher."

BEFORE YOU BEGIN

Get familiar with the idea of anonymous classes.

DESCRIPTION

Create a function that returns an instance of an anonymous class with private fields and methods to access them.

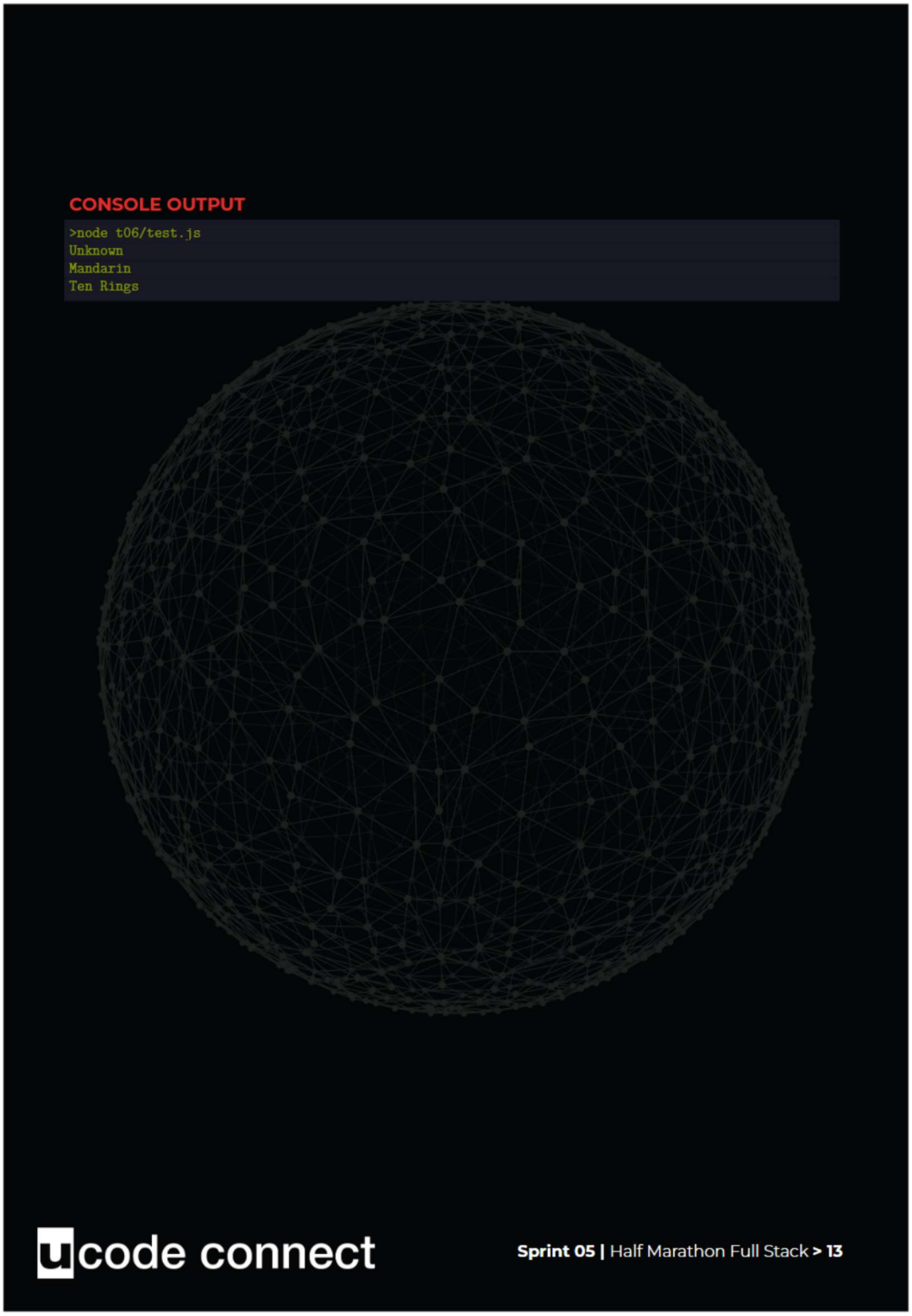
The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```
/*
  Task name: Anonymous
*/

const i = require('./index');

const mandarin = i.getAnonymous('Unknown', 'Mandarin', 'Ten Rings');

console.log([
    mandarin.name,
    mandarin.alias,
    mandarin.affiliation,
].join('\n'));
```

ucode connect

Sprint 05 | Half Marathon Full Stack > **12**


CONSOLE OUTPUT
>node t06/test.js
Unknown
Mandarin
Ten Rings
ucode connect
Sprint 05 | Half Marathon Full Stack > 13

Act Advanced: Task 05

NAME

String frequency

DIRECTORY

`t05_string_frequency/`

SUBMIT

`str-frequency.js`

ALLOWED

JS

DESCRIPTION

Create a class `StrFrequency` that:

- initializes the value of the string using the constructor
- has `letterFrequencies()` method that counts the frequency of each letter in the string
- has `wordFrequencies()` method that counts the frequency of each word in the string
- has `reverseString()` method that inverts the order of letters in the string

Ignore the letter case in all operations. The first two methods must ignore non-letter characters.

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

SYNOPSIS

```
/*
  Task name: String frequency
*/

const StrFrequency = require('./str-frequency');

function test(str) {
  const sf = new StrFrequency(str);
  console.log(`Letters in ${str}`);
  for (const [k, v] of Object.entries(sf.letterFrequencies())) {
      console.log(`Letter ${k} is repeated ${v} times`);
  }

  console.log(`Words in ${str}`);
  for (const [k, v] of Object.entries(sf.wordFrequencies())) {
      console.log(`Word ${k} is repeated ${v} times`);
  }
```

ucode connect

Sprint 05 | Half Marathon Full Stack **> 14**

```
  console.log(`Reverse of the string: ${str}`);
  console.log(`${sf.reverseString()}`);
}

test("Face it, Harley-- you and your Puddin' are kaput!");
console.log('*************');
test('  Test test 123 45 !0 f   HeL1O wOrLd  ');
console.log('*************');
test('');
```

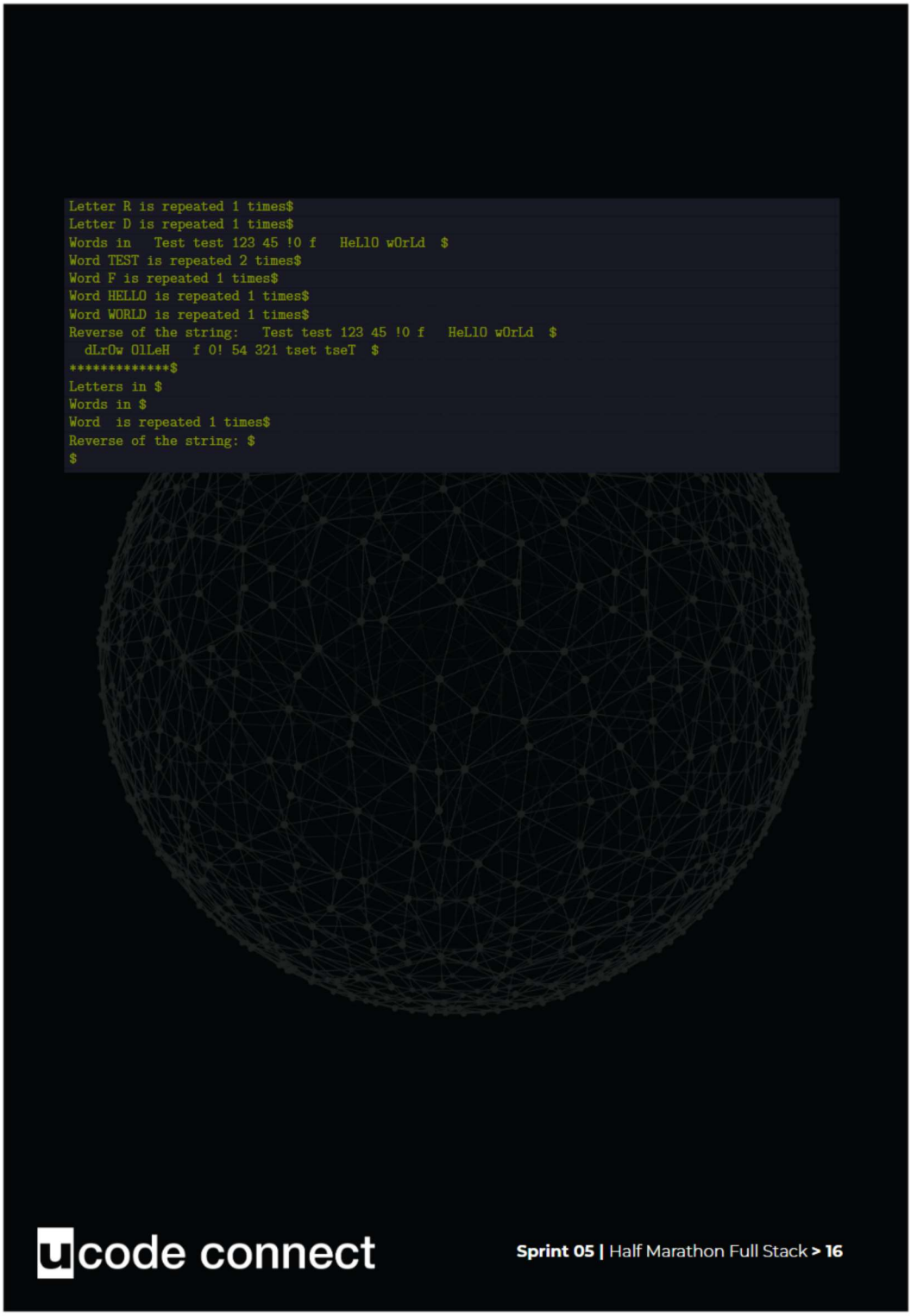
CONSOLE OUTPUT

```
>node t04/test.js | cat -e
Letters in Face it, Harley-- you and your Puddin' are kaput!$
Letter F is repeated 1 times$
Letter A is repeated 5 times$
Letter C is repeated 1 times$
Letter E is repeated 3 times$
Letter I is repeated 2 times$
Letter T is repeated 2 times$
Letter H is repeated 1 times$
Letter R is repeated 3 times$
Letter L is repeated 1 times$
Letter Y is repeated 3 times$
Letter O is repeated 2 times$
Letter U is repeated 4 times$
Letter N is repeated 2 times$
Letter D is repeated 3 times$
Letter P is repeated 2 times$
Letter K is repeated 1 times$
Words in Face it, Harley-- you and your Puddin' are kaput!$
Word FACE is repeated 1 times$
Word IT is repeated 1 times$
Word HARLEY is repeated 1 times$
Word YOU is repeated 1 times$
Word AND is repeated 1 times$
Word YOUR is repeated 1 times$
Word PUDDIN is repeated 1 times$
Word ARE is repeated 1 times$
Word KAPUT is repeated 1 times$
Reverse of the string: Face it, Harley-- you and your Puddin' are kaput!$
!tupak era 'nidduP ruoy dna uoy --yelraH ,ti ecaF$
*************$
Letters in   Test test 123 45 !0 f   HeL1O wOrLd  $
Letter T is repeated 4 times$
Letter E is repeated 3 times$
Letter S is repeated 2 times$
Letter F is repeated 1 times$
Letter H is repeated 1 times$
Letter L is repeated 3 times$
Letter O is repeated 2 times$
Letter W is repeated 1 times$
```

ucode connect

Sprint 05 | Half Marathon Full Stack **> 15**

```
Letter R is repeated 1 times$
Letter D is repeated 1 times$
Words in   Test test 123 45 !0 f   HeLlO wOrLd  $
Word TEST is repeated 2 times$
Word F is repeated 1 times$
Word HELLO is repeated 1 times$
Word WORLD is repeated 1 times$
Reverse of the string:   Test test 123 45 !0 f   HeLlO wOrLd  $
  dLrOw OlLeH   f 0! 54 321 tset tseT  $
************$
Letters in $
Words in $
Word  is repeated 1 times$
Reverse of the string: $
$
```



Act Advanced: Task 06

NAME

Access

DIRECTORY

`t06_access/`

SUBMIT

`access.js`

ALLOWED

JS

LEGEND

Things are different now. I have to protect the one thing that I can't live without. That's you.

DESCRIPTION

Create a class `Access` with the following behavior:

- when the property `mark_LXXXV` hasn't been set, return `'undefined'`
- when set to `null`, return `'null'`
- otherwise, return the value

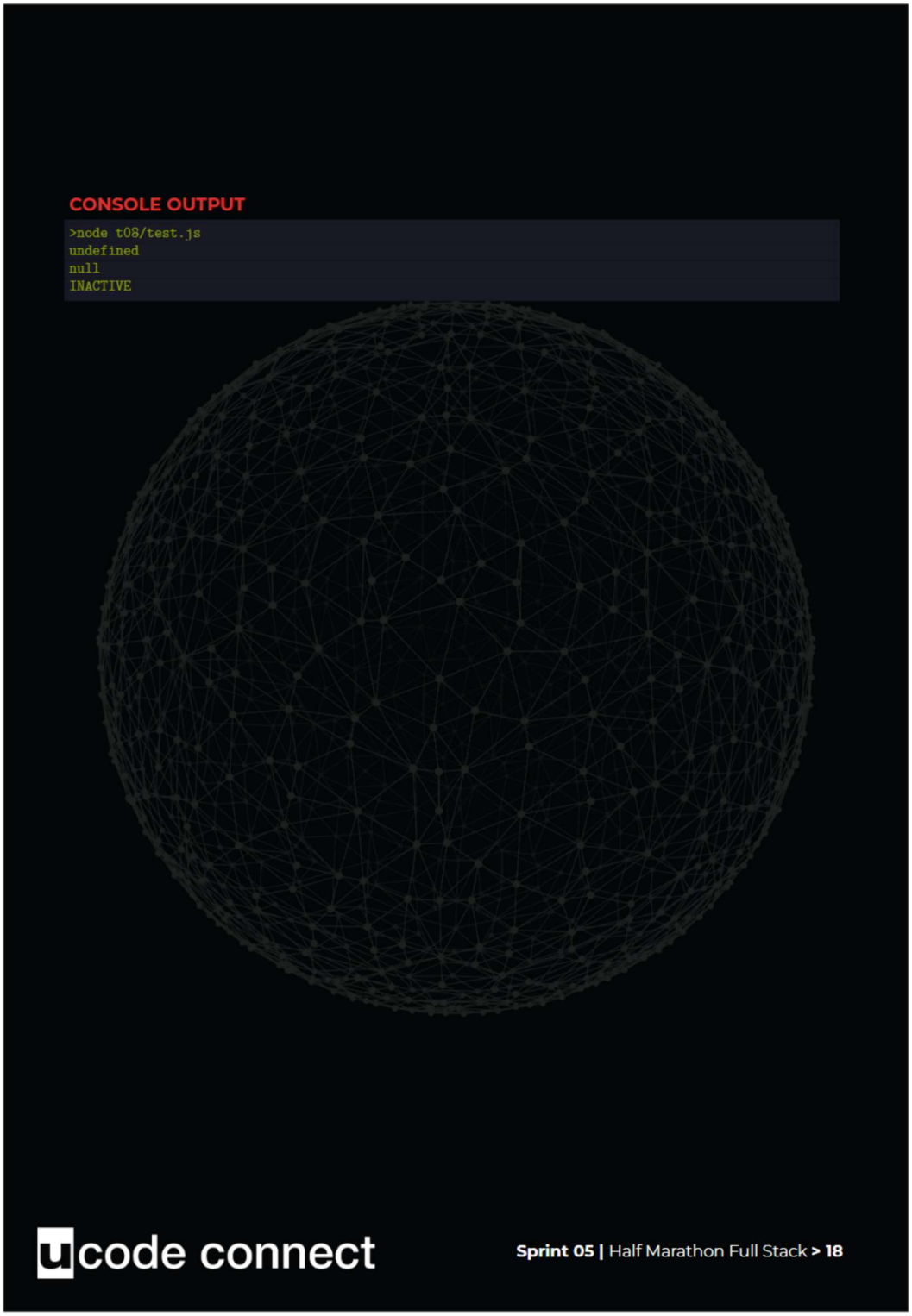
The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```
/*
  Task name: Access
*/

const Access = require('./access');

const o = new Access;
console.log(o.mark_LXXXV);

o.mark_LXXXV = 'null';
console.log(o.mark_LXXXV);

o.mark_LXXXV = 'INACTIVE';
console.log(o.mark_LXXXV);
```

ucode connect

Sprint 05 | Half Marathon Full Stack **> 17**


CONSOLE OUTPUT
>node t08/test.js
undefined
null
INACTIVE
ucode connect
Sprint 05 | Half Marathon Full Stack > 18

Act Advanced: Task 07

NAME

Trait

DIRECTORY

`t07_trait/`

SUBMIT

`markii.js, printable.js`

ALLOWED

JS

LEGEND

"Tony, don't be jealous."
"No, it's subtle, all the bells and whistles."
"Yeah, it's called being a badass."

DESCRIPTION

Create a class `MarkII` and a mixin `Printable`.

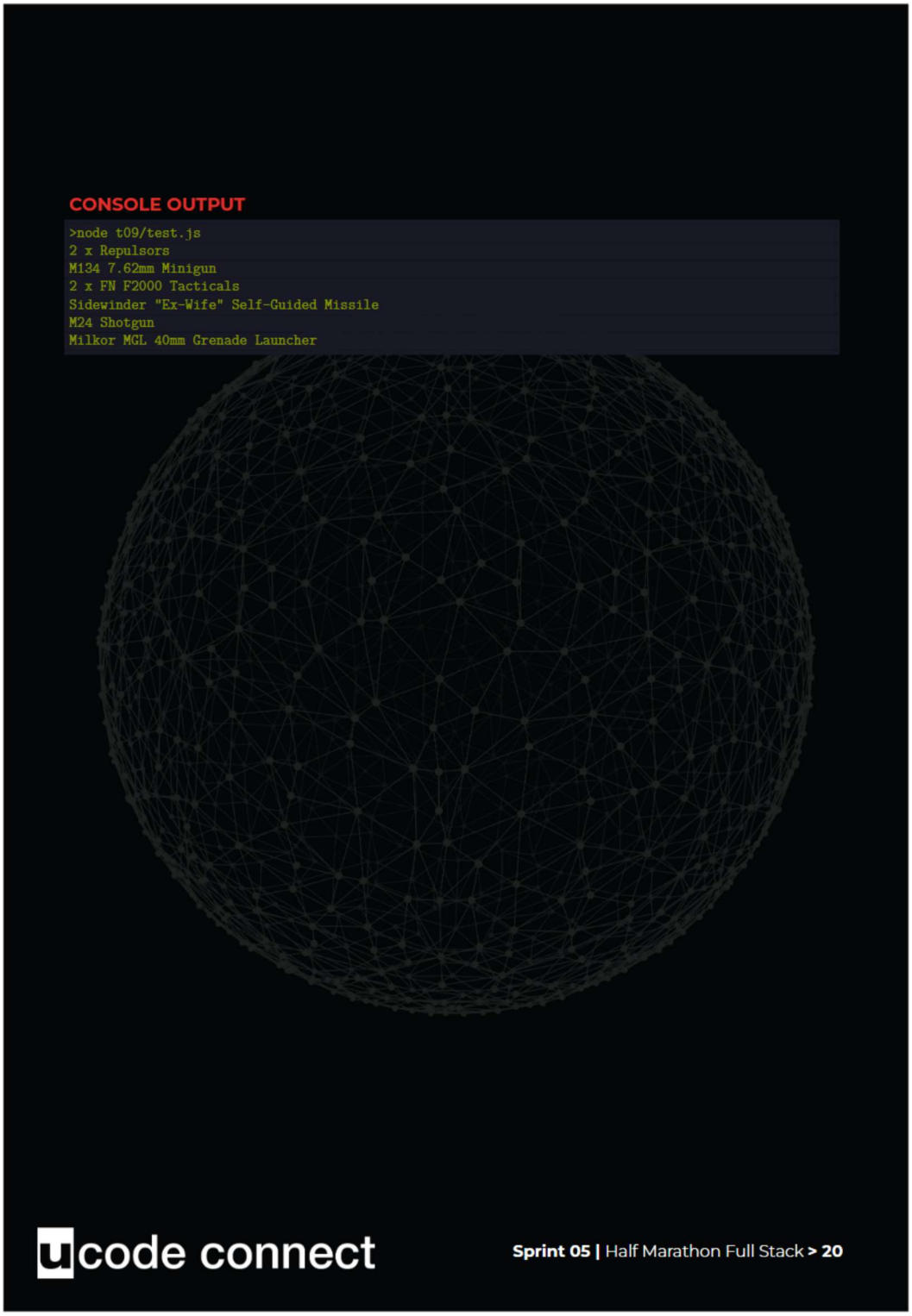
The script from the SYNOPSIS must produce output as shown in the CONSOLE OUTPUT section.

SYNOPSIS

```
/*
  Task name: Mixins
*/

const MarkII = require('./markii');
const Printable = require('./printable');

class WarMachine extends MarkII {}
Object.assign(WarMachine.prototype, Printable);

const wm = new WarMachine;
wm.print();
```

ucode connect

Sprint 05 | Half Marathon Full Stack **> 19**


CONSOLE OUTPUT
>node t09/test.js
2 x Repulsors
M134 7.62mm Minigun
2 x FN F2000 Tacticals
Sidewinder "Ex-Wife" Self-Guided Missile
M24 Shotgun
Milkor MGL 40mm Grenade Launcher
ucode connect
Sprint 05 | Half Marathon Full Stack > 20

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.