Installing Node.js

Reference - ‣

There are a few ways you can install Node.js

1. Build it from Source (difficult, not recommended)
2. Use a package manager like `brew` , `snap` etc
3. Use `nvm` (Node Version Manager)

Node version manager

NVM is a `cli` (Command line interface) that lets you install/upgrade the version of Node.js you have locally.

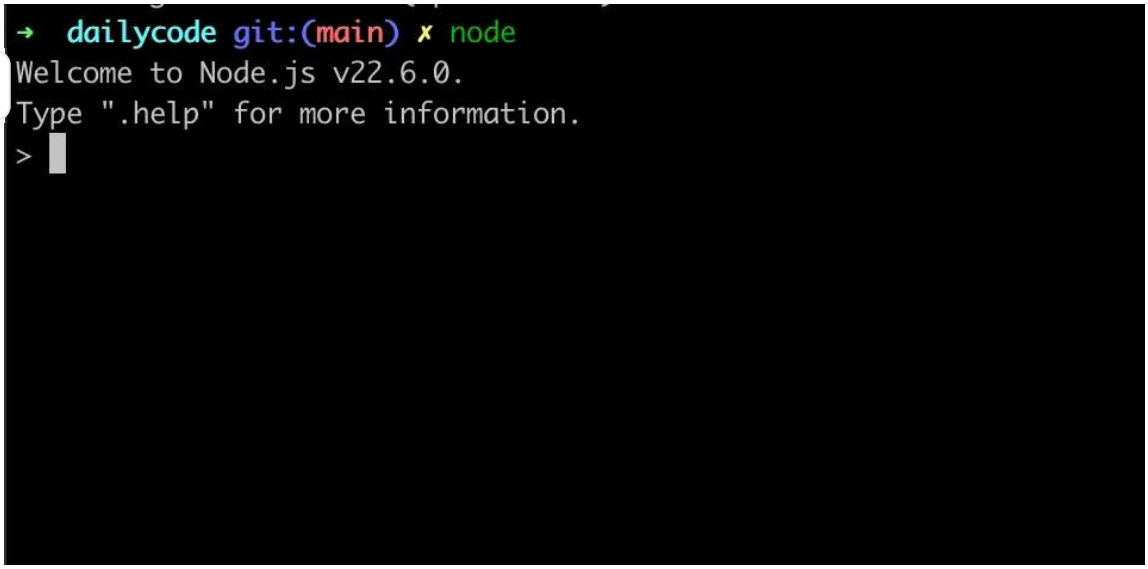
- Step 1 - Install `Node version manager`
  - Linux/Mac

    ```
    curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
    ```

  - Windows - First install WSL, then use the same steps as above
- Step 2 - Install Node

  ```
  nvm install v22
  ```

- Try running `node`



Using Node.js to do assignments

Our assignments are present here - https://github.com/100xdevs-cohort-3/assignments

- Step 1 - Clone the repo locally. You can click the Download Zip button and get the code locally
- Unzip the folder, and open it in visual studio code
- Go through the assignments README week by week and try to solve them.

Easy JS assignments for week 2

These include fairly simple algorithmic problems, each teaching you something new

Anagram

https://github.com/100xdevs-cohort-3/assignments/blob/main/week-2/week-2-js/easy/anagram.js

- Things to learn -
  - toLowerCase
  - split

```
function isAnagram(str1, str2) {
  if (str1.length !== str2.length) {
    return false;a
  }
  const sortedStr1 = str1.toLowerCase().split('').sort().join('');
  const sortedStr2 = str2.toLowerCase().split('').sort().join('');
  return sortedStr1 === sortedStr2;
}
```

Expenditure analysis

https://github.com/100xdevs-cohort-3/assignments/blob/main/week-2/week-2-js/easy/expenditure-analysis.js

- Things to learn -
  - Arrays to object conversion.
  - Object.keys

```
function calculateTotalSpentByCategory(transactions) {
  const categories = {};
  transactions.forEach((transaction) => {
    if (!categories[transaction.category]) {
      categories[transaction.category] = 0;
    }
    categories[transaction.category] += transaction.price;
  });
  return Object.keys(categories).map((category) => ({
    category,
    totalSpent: categories[category],
  }));
}
```

Find largest element

https://github.com/100xdevs-cohort-3/assignments/blob/main/week-2/week-2-js/easy/findLargestElement.js

Things to learn - nothing really

```
function findLargestElement(numbers) {
```

```
    let largestElement = numbers[0];
    for (let i = 1; i < numbers.length; i++) {
        if (numbers[i] > largestElement) {
            largestElement = numbers[i];
        }
    }
    return largestElement;
}
```

Medium JS Assignments for week 2

Count vowels

Things to learn -

- toLowerCase

```
function countVowels(str) {
    const vowels = ['a', 'e', 'i', 'o', 'u'];
    let count = 0;
    for (let i = 0; i < str.length; i++) {
        if (vowels.includes(str[i].toLowerCase())) {
            count++;
        }
    }
```

```
    return count;
```


Palindrome

Things to learn

1. `,` and `.` should be removed from the string
2. Using `filter`

```
function isPalindrome(str) {
  const lowercaseStr = str.toLowerCase();
  const filteredStr = lowercaseStr.split('').filter((char) => (char !== '?' && char !=
  const reversedStr = filteredStr.split('').reverse().join('');
  return filteredStr === reversedStr;
}
```

Times

Things to learn

1. How long does synchronous code take to run

```
function calculateTime(n) {
   const startTime = Date.now();
   let sum = 0;
   for (let i = 1; i <= n; i++) {
      sum += i;
   }
   const endTime = Date.now();
   console.log(endTime - startTime);
}

calculateTime(100);
```

Hard JS Assignments

These involve writing your own classes

Todo List

```
class Todo {
  constructor() {
    this.todos = [];
  }

  add(todo) {
    this.todos.push(todo);
  }

  remove(indexOfTodo) {
   this.todos.splice(indexOfTodo, 1);
  }

  update(index, updatedTodo) {
    if (index < 0 || index >= this.todos.length) {
      return;
    }
    this.todos[index] = updatedTodo;
  }

  getAll() {
    return this.todos;
  }

  get(indexOfTodo) {
    if (indexOfTodo < 0 || indexOfTodo >= this.todos.length) {
      return null;
    }
    return this.todos[indexOfTodo];
  }
```

```
        this.todos = [];
```


Easy Async assignment

These assignments dont have tests. The goal is for you to write them yourselves and test them

Counter

Things to learn - setTimeout

```
let counter = 0;
function increaseAndPrint() {
  console.log(counter)
  counter = counter + 1;
}
setInterval(increaseAndPrint, 1000);
```

Counter using setTimeout

```
let counter = 0;
function increaseAndPrint() {
    console.log(counter)
    counter = counter + 1;
    setTimeout(increaseAndPrint, 1000);
}
setTimeout(increaseAndPrint, 1000);
```

Read from a file

```
const fs = require('fs');

const filePath = 'a.txt';

fs.readFile(filePath, 'utf8', (err, data) => {
  if (err) {
    console.error('Error reading file:', err);
    return;
  }
  console.log('File content:', data);

  // Perform the expensive operation after file read completes
});

function expensiveOperation() {
  let sum = 0;
  for (let i = 0; i < 1e8; i++) { // Simulating a very expensive operation
    sum += i;
  }
  console.log('Expensive operation result:', sum);
}

expensiveOperation();
```

Writing to a file

```
const fs = require('fs');

const data = 'This is the content to write to the file.';

function expensiveOperation() {
  let sum = 0;
  for (let i = 0; i < 1e8; i++) { // Simulating a very expensive operation
    sum += i;
  }
  console.log('Expensive operation result:', sum);
```

```
fs.writeFile("a.txt", data, (err) => {
  if (err) {
    console.error('Error writing file:', err);
    return;
  }
  console.log(`Data written to a.txt`);

});

expensiveOperation();
```

Medium async

File cleaner

Things to learn - Doing two async operations one after another

```
const fs = require('fs').promises;

async function cleanFile(filename) {
  try {
    // Read the file
    const content = await fs.readFile(filename, 'utf8');

    // Remove extra spaces
    const cleanedContent = content.replace(/\s+/g, ' ').trim();

    // Write back to the same file
    await fs.writeFile(filename, cleanedContent);

    ... has been cleaned.`);
```

```
    console.error('Error:', error.message);
}

// Usage
const filename = 'a.txt';
cleanFile(filename);
```


Clock

Things to learn -

1. Optional arguments
2. Terenary operator

```
function formatTime(date, use24Hour = true) {
  const hours = use24Hour ? date.getHours() : (date.getHours() % 12 || 12);
  const minutes = date.getMinutes().toString().padStart(2, '0');
  const seconds = date.getSeconds().toString().padStart(2, '0');
  const ampm = use24Hour ? '' : (date.getHours() >= 12 ? 'PM' : 'AM');

  return `${hours.toString().padStart(2, '0')}:${minutes}:${seconds}${ampm ? '
}

function displayTime() {
  const now = new Date();
  const time24 = formatTime(now);
  const time12 = formatTime(now, false);

  console.clear();
  console.log(`24-hour format: ${time24}`);
  console.log(`12-hour format: ${time12}`);
}

function startClock() {
  displayTime();
  setInterval(displayTime, 1000);
}
```

1 of 7

```
nsole.log("Press Ctrl+C to stop the clock.");
```