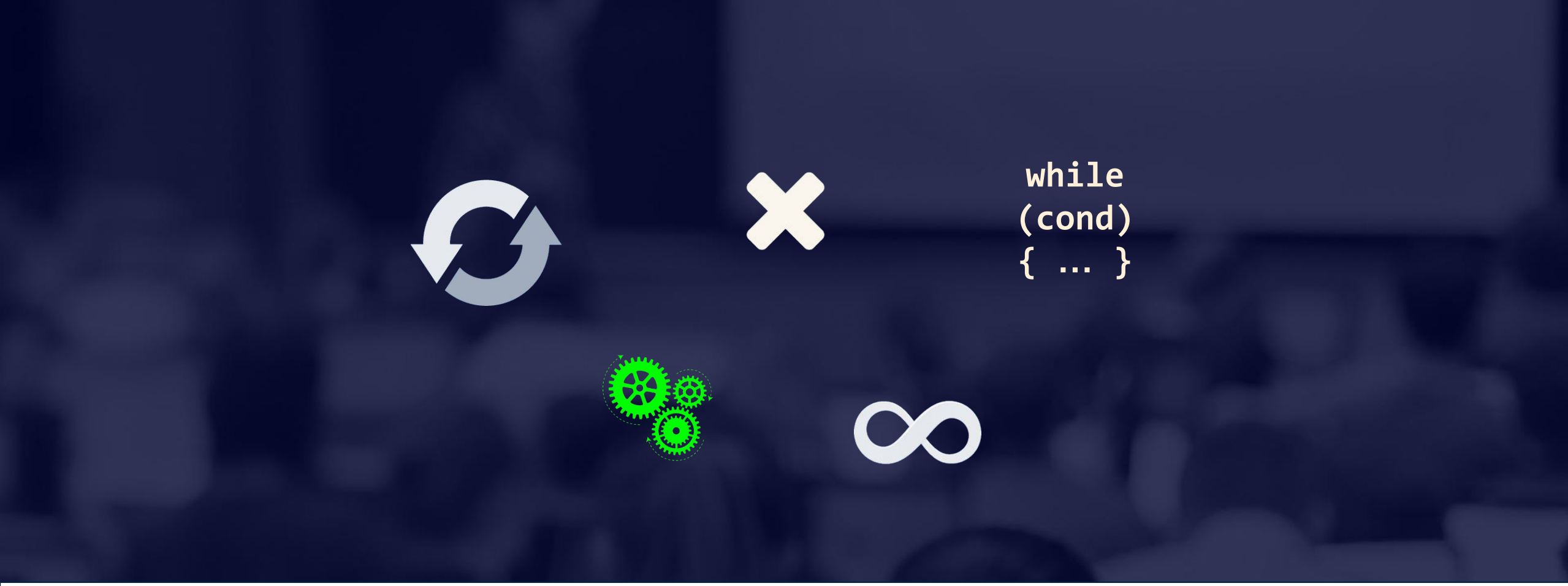
While Loops



Repeating Code While a Condition Is True

Table of Contents

Review

For Loops, Loops with a Step

Prefix and Postfix Increment / Decrement

- **Prefix ++** and **--**

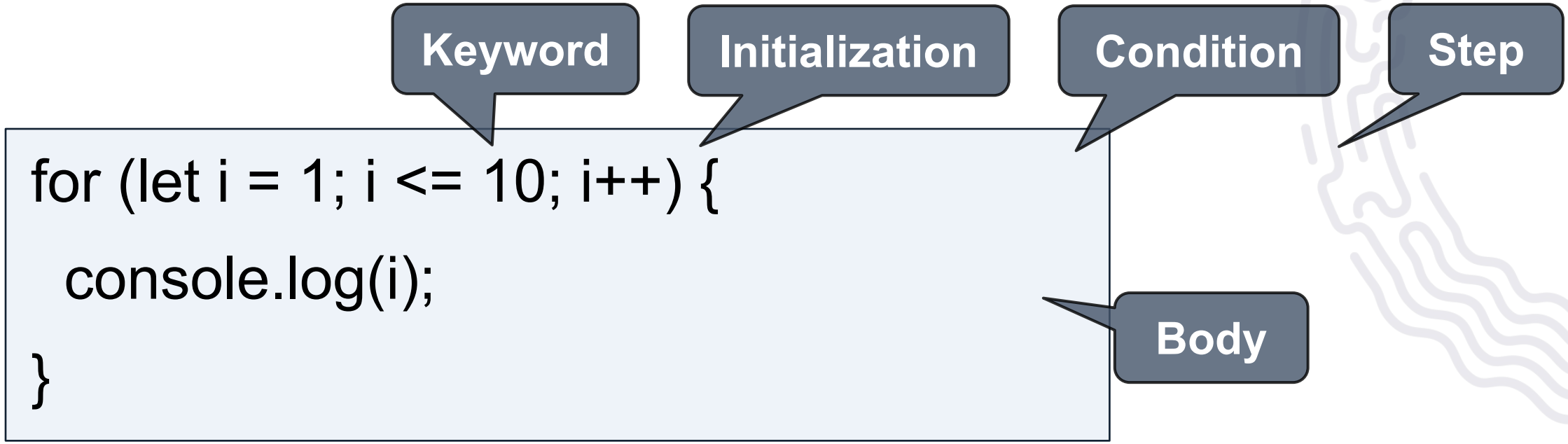
```
let a = 1;
console.log(--a); // 0
console.log(a);   // 0
```

- **Postfix ++** and **--**

```
let a = 1;
console.log(a++); // 1
console.log(a);   // 2
```

Simple For-Loop

- **For loops** repeat a certain code block a known number of times
  - Example: printing the numbers

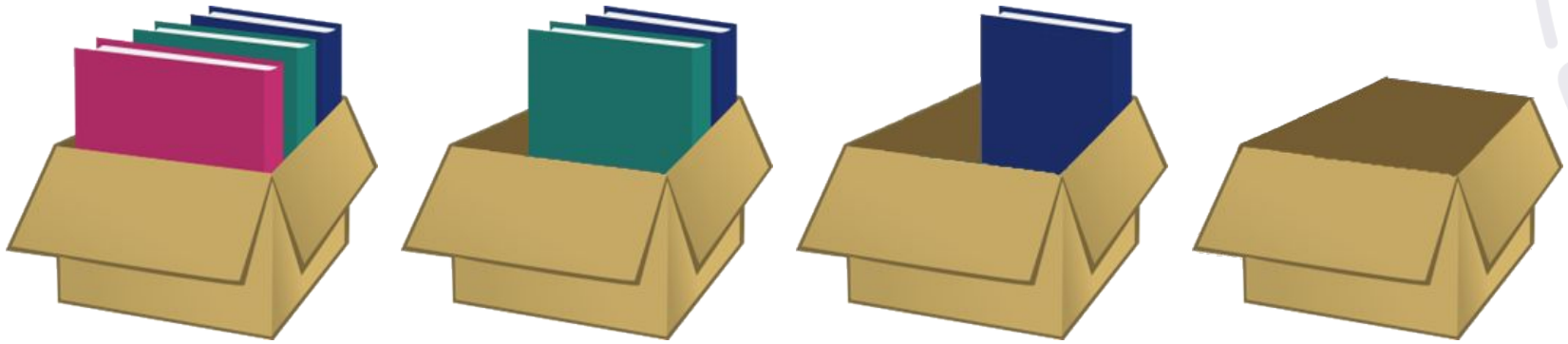


Introduction

The Need of While Loops

Real-Life Example: Box of Books

- Unpack a box of books
  - Remove the first book from the box
  - Keep removing books **until** the box is emptied

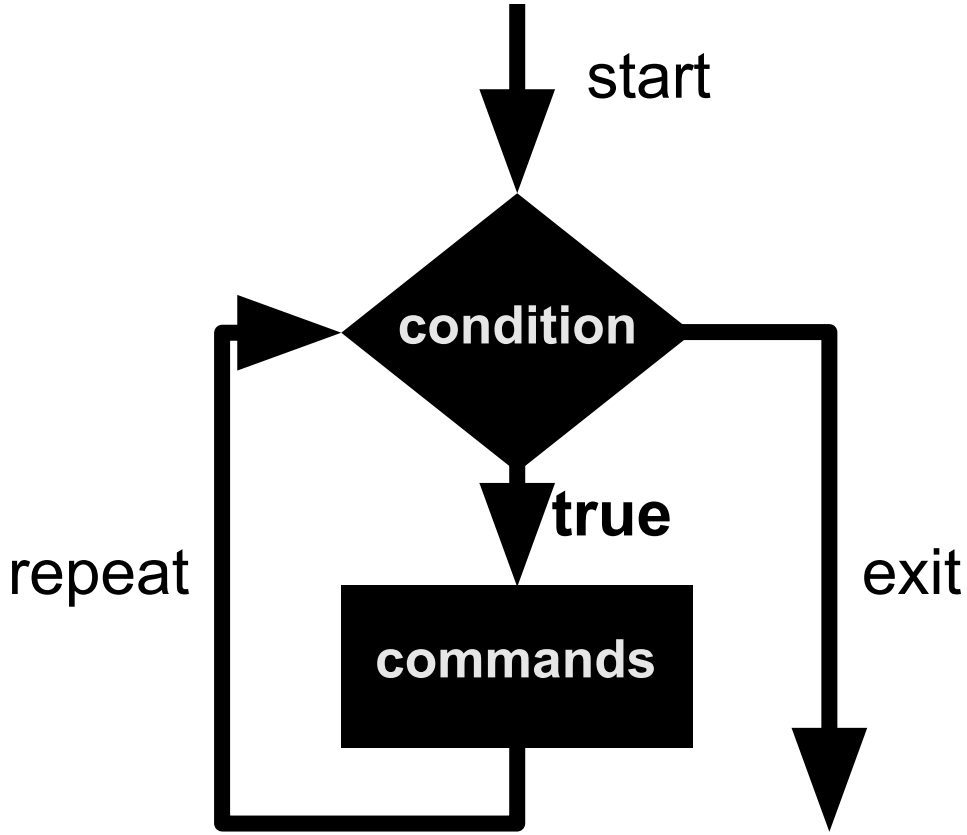


While Loop

Control Flow Statement

While Loop

- Used to **repeat a code** block until an **exit condition** is met




```
while (condition) {
    // commands
}
```

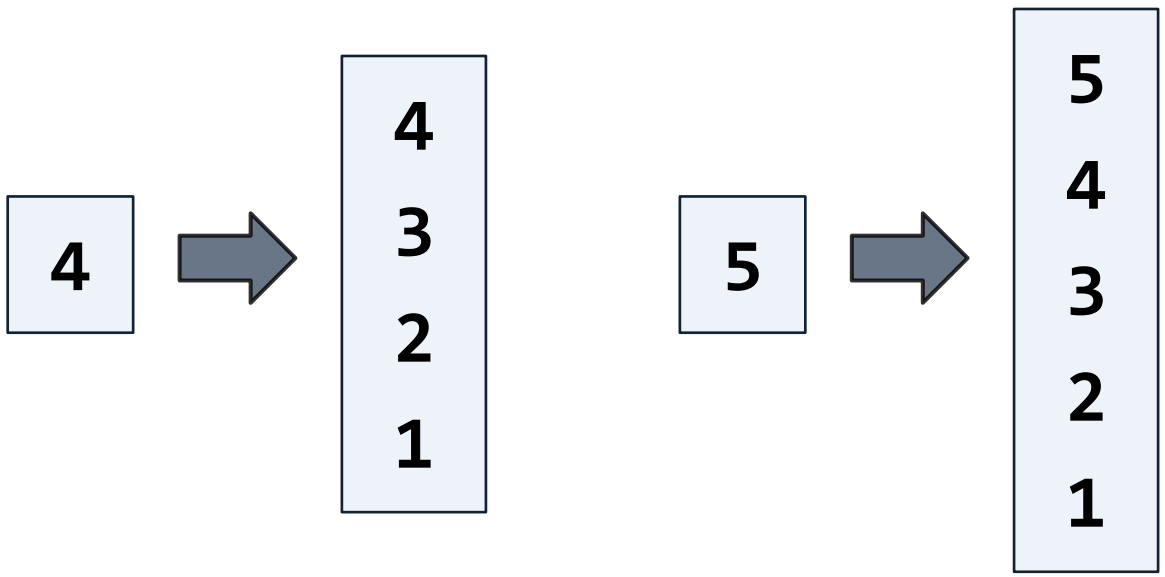
While Loop – Example

- Print the numbers from 1 to 5

```
let i = 1;
while (i <= 5) {
  console.log(i);
  i++;
}
```

Problem: Decreasing Numbers

- Print the **numbers from N down to 1**, using a **while** loop
  - Write a function which receives number: **n**
  - Print the numbers **n … 1**



Solution: Decreasing Numbers

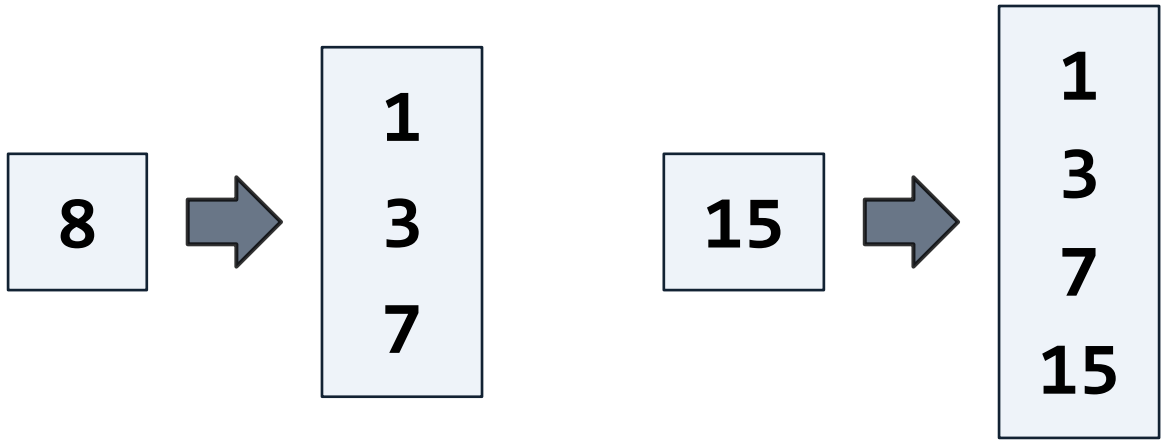
```
function decreasingNumbers(n) {
  while (n >= 1) {
    console.log(n);
    n--;
  }
}
```

```
decreasingNumbers(5);
```

```
decreasingNumbers(10);
```

Problem: Sequence 2k + 1

- Write a function to print a **sequence of numbers**:
  - The first number is **1**
  - Each next number is **2 times the previous number + 1**
  - Take as input the max number **n**
  - Print the numbers from the **sequence**, which are **≤ n**

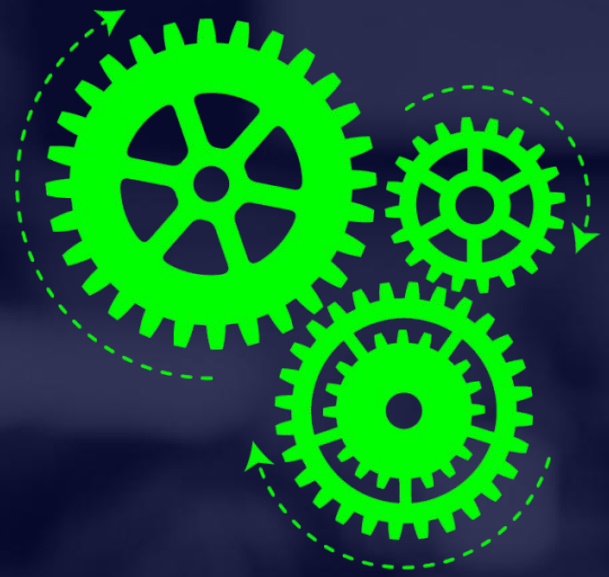


Solution: Sequence 2k + 1

```
function sequence(n) {
  let k = 1;
  while (k <= n) {
    console.log(k);
    k = k * 2 + 1;
  }
}
```

```
sequence(5);
```

```
sequence(10);
```



While or For Loop?

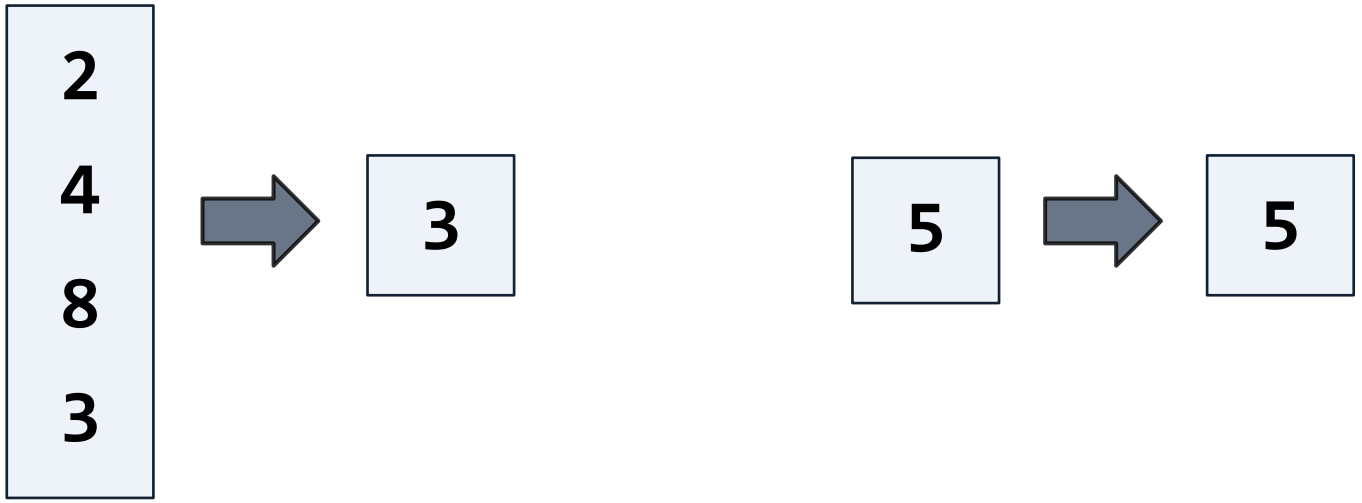
Choosing the Right Loop Type

While or For?

- The **`while`** and **`for`** loops both **repeat a block of code**
- Use **`for`**-loop when you preliminary know the **number of iterations**
  - E.g. repeat exactly **n** times
- Use **`while`** when you don't know when the exit condition will be met
  - E.g. repeat until stopped

Problem: Odd Number

- Write a program to **enter an odd number**
  - Takes numbers from the input until an **odd number** is entered
  - Print the **odd** number as output



Solution: Odd Number

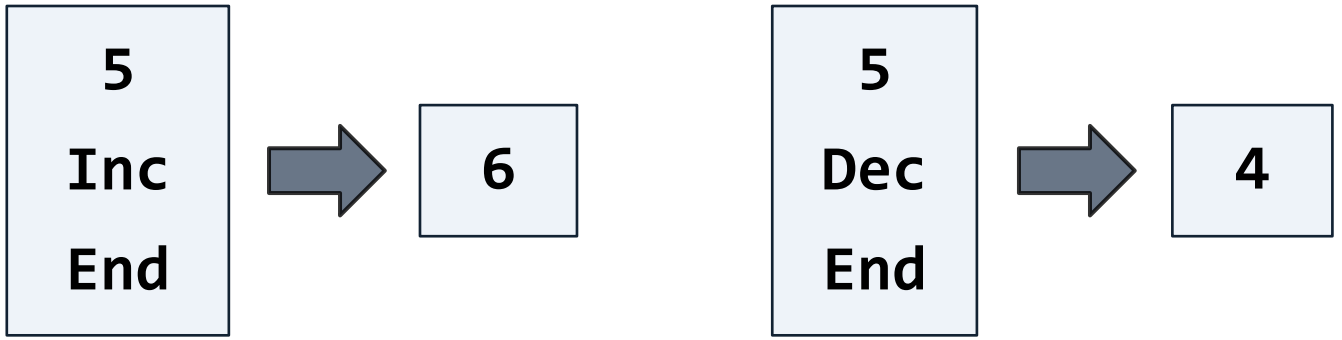
```
function findFirstOddNumber(numbers) {
  let num = numbers.shift();
  while (num % 2 === 0) {
    // The number is not odd □ read a new one
    num = numbers.shift();
  }
  console.log(num);
}
```

```
findFirstOddNumber(
  [2, 4, 8, 3]);
```

```
findFirstOddNumber(
  [1, 3, 5]);
```

Problem: Number Processor

- Write a function to **process a sequence of commands**:
  - Receives an **initial number** from the input
  - Receives a sequence of the following **commands**:
    - **`Inc`** – add 1 to the number (increment)
    - **`Dec`** – subtract 1 from the number (decrement)
    - **`End`** – print the number and stop the program



Solution: Number Processor

```
function numberProcessor(num, commands) {
  let command = commands.shift();
  while (command !== "End" &&
      command != undefined) {
    switch (command) {
      case "Inc": num++; break;
      case "Dec": num--; break;
    }
    command = commands.shift();
  }
  console.log(num);
}
```

```
numberProcessor(5, ['Inc', 'End']);
```

The "break" Operator

Exiting from a Loop

The "break" Operator

- Used for prematurely **exiting** the loop
- Can only be executed from the loop's **body**
- When **break** is executed, the code inside the loop's body after it **is skipped** (does not execute)

```
while (true) {
  // Some code ...
  if (...) break;
  // More code ...
}
```

Example: Break Operator

```
function sumNumbers(nums) {
  let sum = 0;
  while (true) {
    let nextNum = nums.shift();
    if (nextNum == undefined) {
      // The last number was reached
      break;
    }
    sum += nextNum;
    console.log("Sum:", sum);
  }
}
```

```
Sum: 10
Sum: 30
Sum: 60
```

- Sum numbers until **the input is fully processed**

```
sumNumbers(
  [10, 20, 30]);
```

Infinite While Loop

Using `while (true) { ... }`

Infinite While Loop

- **Infinite loop** = repeating a block of code **an infinite number** of times
- Infinite **`while`** loops: use **true** as loop condition

```
while (true) {
    // Commands
}
```

Example: Infinite While Loop (Bug)

```
function infiniteWhileLoop(commands) {
  let command = commands.shift();
  while (command !== "End") {
    console.log("Executing: " + command);
  }
}
```

Bug: always true (never changed)

```
infiniteWhileLoop(['One', 'End']);
```

Example: Finite Loop (Bug Fixed)

```
function finiteWhileLoop(commands) {
  let command = commands.shift();
  while (command !== "End") {
    console.log("Executing: " + command);
    command = commands.shift();
  }
}
```

```
infiniteWhileLoop(['One', 'End']);
```

Example: Infinite Loop with Break

```
function processCommands(commands) {
  while (true) {
    command = commands.shift();
    if (command == "End" ||
        command == undefined)
      break;
    console.log("Executing: " + command);
  }
}
```

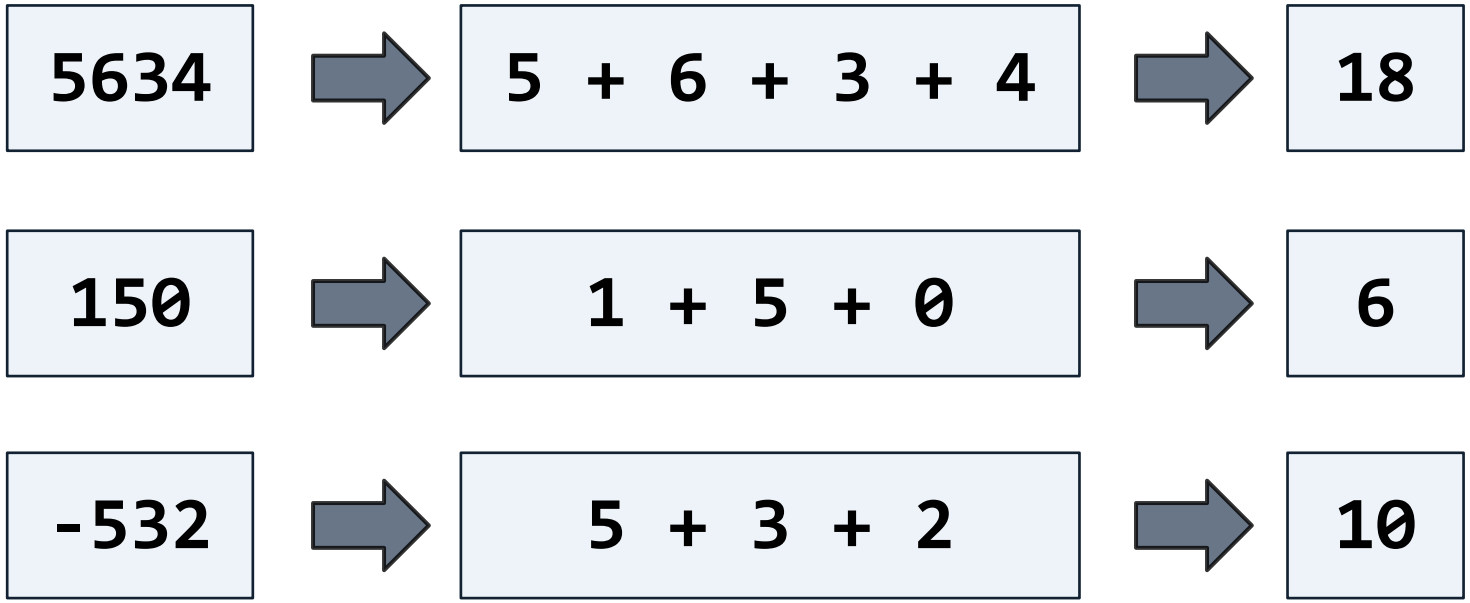
```
processCommands(['One', 'End']);
```

Live Exercises

Practical Problem Solving

Problem: Sum Digits

- Write a function to **sum the digits** of given number
  - Receives an **input number**
  - **Sum** its **digits** and print the sum



Solution: Sum Digits

```
function sumDigits(n) {
  let sum = 0;
  while (n > 0) {
    sum += n % 10;
    n = Math.floor(n / 10);
  }
  console.log(sum);
}
```

Also consider negative n

Sum the last digit

Remove the last digit

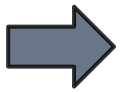
```
sumDigits(5634);
```

```
sumDigits(120);
```

Problem: Favorite Book

- Write a function to **guess for a favorite book**, which:
  - Receives a **favorite book's name**
  - Receives **book names** until it reaches the **favorite book**
  - Prints "**`Book found!`**" and stops afterwards
  - Prints "**`Invalid book: `** " + book for all invalid books

```
Alice in Wonderland
Winnie the Pooh
Alice in Wonderland
```



```
Invalid book: Winnie the Pooh
Book Found!
```

Solution: Favorite Book

```
function favoriteBook(favoriteBook, books) {
  let book = books.shift();
  while (book !== favoriteBook) {
    console.log(`Invalid book: ${book}`);
    // Read the next book
    book = books.shift();
  }
  console.log("Book found!");
}
```

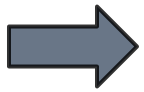
TODO: check for undefined and break

```
favoriteBook('Alice in Wonderland',
['Winnie the Pooh', 'Alice in Wonderland']);
```

Problem: Find Min and Max

- Write a function to **find the min and max numbers**
  - Reads numbers until "**END**" is reached
  - Prints the **biggest** and the **smallest** number

```
10
20
304
0
50
END
```



```
Min number: 0
Max number: 304
```

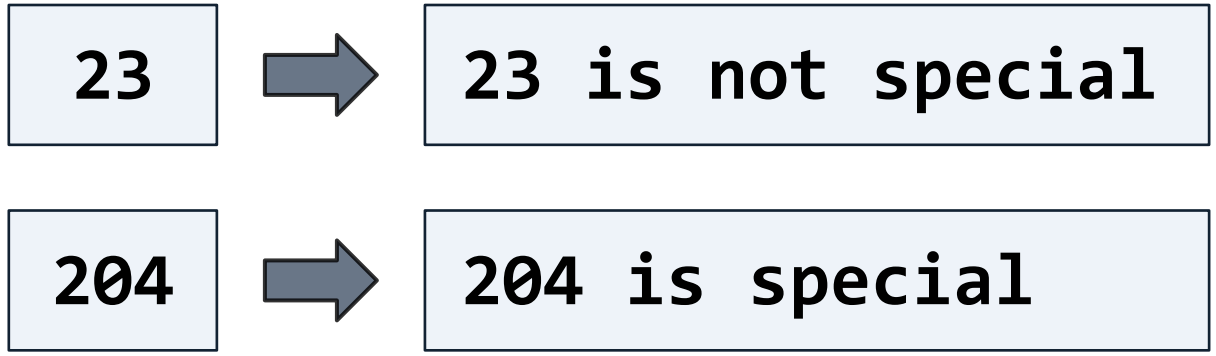
Solution: Find Min and Max

```
function minAndMax(lines) {
  let min = Infinity;
  let max = -Infinity;
  let nextLine = lines.shift();
  while (nextLine !== "END" &&
      nextLine !== undefined) {
    let num = Number(nextLine);
    if (num < min) min = num;
    if (num > max) max = num;
    nextLine = lines.shift();
  }
  // TODO: Print the output
}
```

```
minAndMax([
  10,
  20,
  30,
  -5,
  'END'
]);
```

Problem: Special Number

- Write a function to check if given number is **special**:
  - **Special numbers** are divisible by all of their digits without remainder
  - Receives a number and check whether it is a special number
  - Print "**`{num} is special`**" if the number is special
  - Otherwise, print "**`{num} is not special`**"



Solution: Special Number

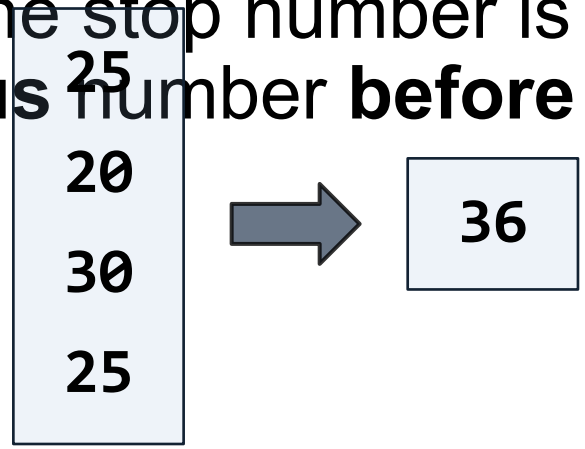
```
function specialNumber(num) {
  let numDigits = num;
  let isSpecial = true;
  while (numDigits > 0) {
    let digit = numDigits % 10;
    numDigits = Math.floor(numDigits / 10);
    if (digit != 0 && num % digit != 0) {
      isSpecial = false;
      break;
    }
  }
  // TODO: Print the final output
}
```

```
specialNumber(204);
```

```
specialNumber(23);
```

Problem: Special Bonus

- Write a function to apply a **20% bonus** for certain number:
  - Receives a number from the input: the "**stop number**"
  - Receives **numbers** from the input until it finds the stop number
  - When the stop number is found, increase the value of the **previous** number **before it** with **20%** and print it



30 * 120% = **36**

Solution: Special Bonus

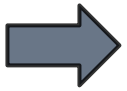
```
function specialBonus(stopNum, nums) {
  let previousNum = stopNum;
  while (true) {
    let num = nums.shift();
    if (num == stopNum ||
        num == undefined)
      break;
    previousNum = num;
  }
  console.log(previousNum * 1.2);
}
```

```
specialBonus(
  25,
  [20, 30, 25]
);
```

Problem: Account Balance

- Write a function to calculate an **account balance**
  - Receives a sequence of **incomes** / **expenses**, until "**End**" is read
  - Adds the money to the balance (starting from 0) and print "**`Increase: {money}`** " or "**`Decrease: {money}`**"
  - Finally, prints the **account balance** (like shown below)

```
500
15.5
-80.35
End
```



```
Increase: 500.00
Increase: 15.50
Decrease: 80.35
Balance: 435.15
```

Solution: Account Balance

```
function accountBalance(lines) {
  let balance = 0;
  let line = lines.shift();
  while (line !== 'End' && line !== undefined) {
    let amount = Number(line);
    balance += amount;
    // TODO: Print Increase / Decrease
    line = lines.shift();
  }
  console.log(`Balance: ${balance.toFixed(2)}`);
}
```

```
accountBalance(['500', '15.5', '80.35', 'End']);
```

Summary

- The `while` loop executes a block of code multiple times
  - While the loop condition is true
- Use `for` when you initially know the number of repetitions, `while` otherwise
- `while` loops can be infinite
  - Use the `break` operator to exit from the loop on certain condition

Questions?

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © Kingsland University – https://kingslanduniversity.com


KINGSLAND
UNIVERSITY
THANK YOU