KINGSLAND
UNIVERSITY

Nested Loops

```
for (...)
    for
  (...)
   { ...  }
```

Loops Inside Other Loops

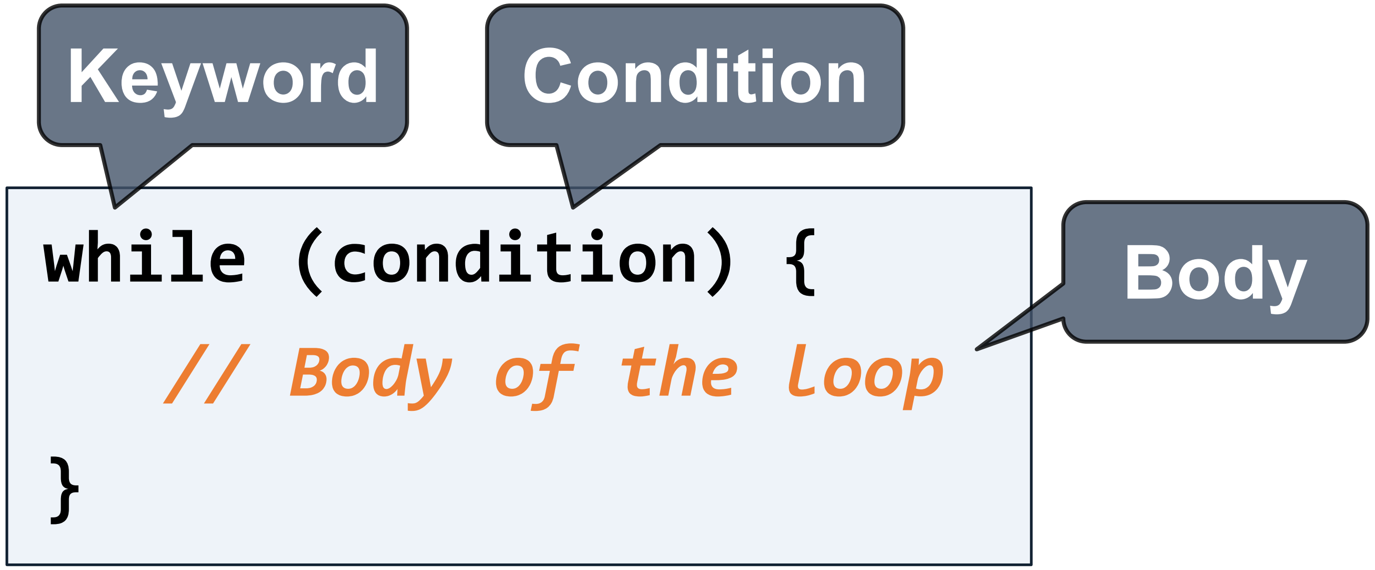
Table of Contents

Review

While Loops

While Loop

- Control flow **statement**
  - Executes code repeatedly while a condition is **true**



```
while (condition) {
    // Body of the Loop
}
```

Example: While Loop

- Print the numbers from **1 to 5**

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

1 ... 5

While or For?

- `while` and `for` loops **repeat** blocks of **code**
  - Use `for` when you know in advance the **number of repetitions**
    - For example, repeat exactly 10 times
  - Use `while` when you don't know when the **exit condition** will be met
    - For example, repeat until `0` is reached

The "break" Operator

- Used for **prematurely exiting** the loop
- Can only be executed from the **body**, during **an iteration** of the loop
- **break** immediately exits from the loop
  - The rest of the loop body **is skipped**

```
let i = 1;
while (true) {
  if (i>10) break;
  i++;
}
```

Complex Loops

Loops with a Special Step

Complex Loops

- For-loops may have different **steps**

```
for (let i = n; i >= 1; i--) …
```

```
for (let j = 1; j <= n; j += 2)
```

```
…
```

```
for (let k = 1; k <= n; k *= 2)
```

```
…
```

```
for (let d = n; d > 0; d /= 2) …
```

Problem: Numbers from N down to 1

- Write a function to print the **numbers from N down to 1**
  - Receives a number **n**
  - Prints the numbers from **n** down to **1**

```
100  →  100, 99, …, 2, 1

10   →  10, 9, 8, …, 1
```

Solution: Numbers from N down to 1

```
function numbersFromNto1(n) {
  let result = '';
  for (let i = n; i >= 1; i--) {
    if (i < n)
      result += ", ";
    result += i;
  }
  console.log(result);
}
```

```
numbersFromNto1(10);
```

Decrement i

Reversed condition

Append comma before each number except the first

Problem: Numbers from 1 to N with Step 3

- Write a function to print the **numbers from 1 to n with step 3**:
  - Receives a number **n**
  - Prints the numbers from **1** to **n** with step **3**

`10`	→	`1, 4, 7, 10`
`7`	→	`1, 4, 7`
`14`	→	`1, 4, 7, 10, 13`

Solution: Numbers from 1 to N with Step 3

```
function numbersWithStep(n) {
  let result = '';
  for (let i = 1; i <= n; i += 3) {
    if (i > 1)
      result += ", ";
    result += i;
  }
  console.log(result);
}
```

Use a step = 3

```
numbersWithStep(9);
```

Problem: Even Powers of 2

- Write a function to print the **even powers of 2**:
  - Receives a number **n**
  - Prints the even powers of 2 up to $2^{\mathbf{n}}$:
    - 2^0, 2^2, 2^4, 2^8, …, 2^n

`10`	→	`1, 4, 16, …, 1024`
`7`	→	`1, 4, 16,…, 64`

Solution: Even Powers of 2

```
function evenPowersOf2(n) {
  let num = 1;
  let result = '';
  for (let i = 0; i <= n; i += 2) {
    if (i > 0)
      result += ", ";
    result += num;
    num = num * 2 * 2;
  }
  console.log(result);
}
```

Step = 2

```
evenPowersOf2(10);
```

Introduction

Nested Loops

Real Life Example: Clock

- Imagine **how the clock works**
  - A sequence of **iterations**
  - At each iteration **the rightmost digit is increased**
  - When a digit **overflows** (reaches 10), it starts from **0** and the digit on its left is increased

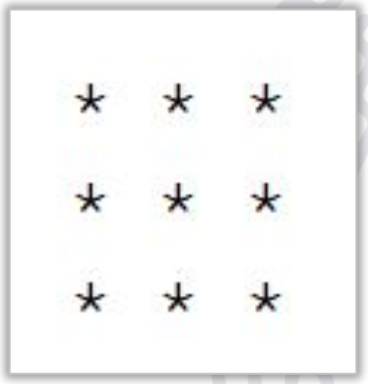
Nested Loops

Loops Inside Other Loops

Nested Loops

- We can nest a **loop inside another loop**:

```
let n = 3;
for (let row = 1; row <= n; row++) {
  let result = '';
  for (let col = 1; col <= n; col++)
{
    result += ' *';
  }
  console.log(result);
}
```



```
 * * *
 * * *
 * * *
```

Nested Loops

- Nested loops == several **loops** placed **inside each other**
- **Nested loops** are used:
    - To execute multiple times an **action**, which **executes** multiple **actions**
    - To implement more **complex** calculations and program logic

Multiple Levels of Nested Loops

```
for (let floor = 1; floor <= n; floor++) {
  for (let row = 1; row <= n; row++) {
    for (let col = 1; col <= n; col++) {
      // ...
    }
  }
}
```

The loop variable names must be different

Nested For-Loops

For-Loop Inside a For-Loop

Nested For Loops

- The syntax for a **nested `for loop`** in JS is as follows:

```
// Outer Loop
for (init; condition; increment) {
  // Inner Loop
  for (init; condition; increment) {
      // Commands
  }
}
```

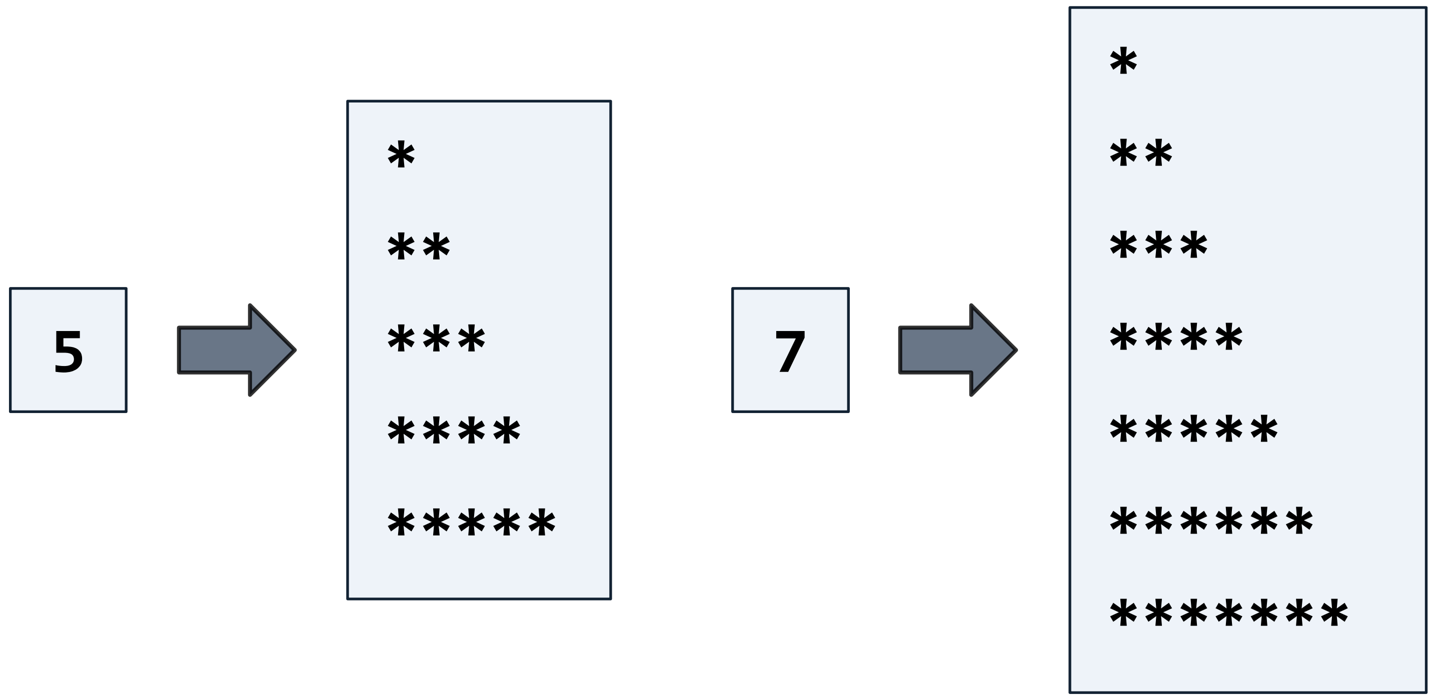
Example: Nested For Loops

```
let rows = 3;
let columns = 2;
for (let r = 1; r <= rows; r++) {
  console.log("row = " + r);
  for (let c = 1; c <= columns; c++)
    console.log("  column = " + c);
}
```

```
row = 1
  column = 1
  column = 2
row = 2
  column = 1
  column = 2
row = 3
  column = 1
  column = 2
```

Problem: Triangle of Stars

- Write a function to print a **triangle of stars** like shown below:
  - Receives the **size** of a triangle from the console
  - Prints a **triangle of stars**



```
5
```

```
*
**
***
****
*****
```

```
7
```

```
*
**
***
****
*****
******
*******
```

Solution: Triangle of Stars

```
function starsTriangle(size) {
  for (let row = 1; row <= size; row++) {
    let stars = '';
    for (let col = 1; col <= row; col++) {
      stars += "*";
    }
    console.log(stars);
  }
}
```

```
starsTriangle(5);
```

Nested While Loops

While Inside Another While

Nested While Loops

```
// Outer Loop
while (condition) {
   // Inner Loop
   while (condition) {
       // Statements
   }
}
```

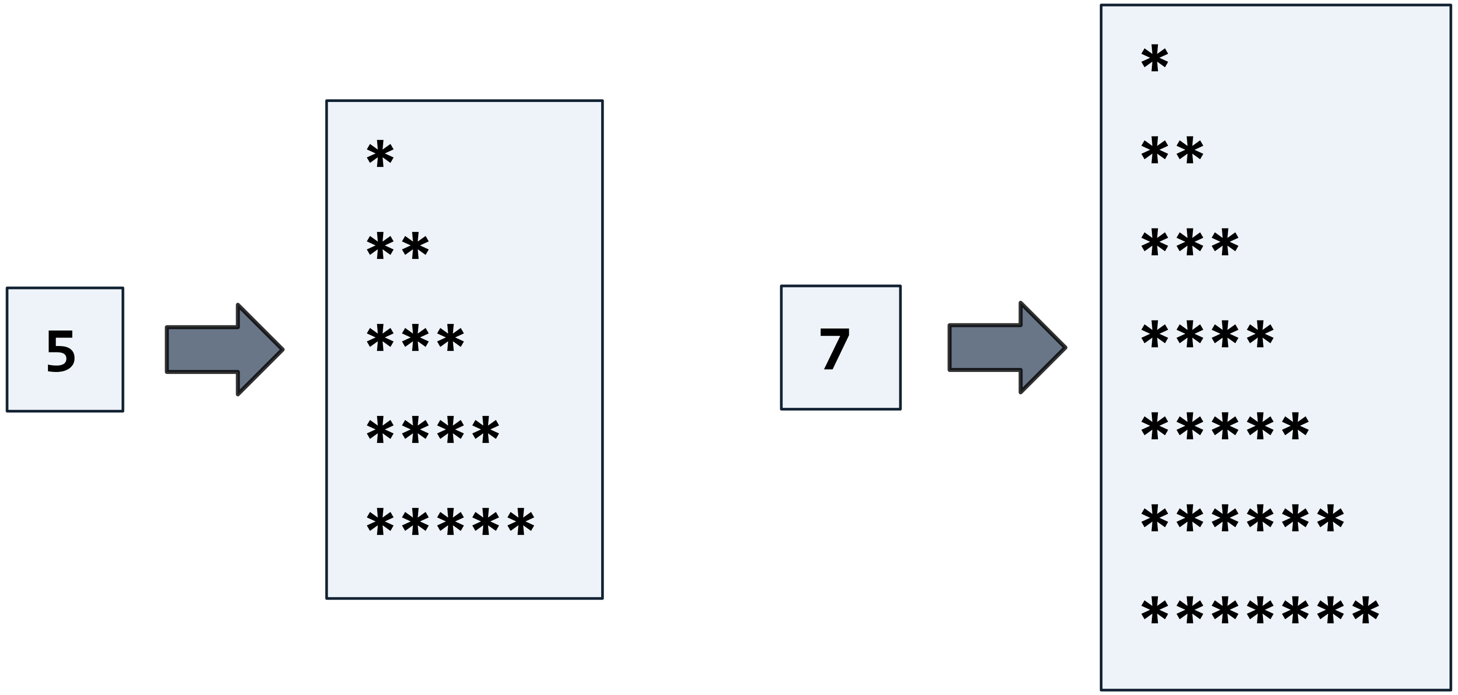
Example: Nested While Loops

```
let row = 1;
while (row <= 2) {
  console.log(`Row: ${row}`);
  let col = 1;
  while (col <= 3) {
    console.log(`  Column: ${col}`);
    col++;
  }
  row++;
}
```

```
// Output
Row: 1
  Column: 1
  Column: 2
  Column: 3
Row: 2
  Column: 1
  Column: 2
  Column: 3
```

Problem: Triangle of Stars with While

- Write a function to print a **triangle of stars** like shown below:
  - Receives the **height** of a triangle from the console
  - Prints a **triangle of stars** using **`while`** loops



```
5
```

```
*
**
***
****
*****
```

```
7
```

```
*
**
***
****
*****
******
*******
```

Solution: Triangle of Stars with While

```
function starsTriangle(height) {
  let row = 1;
  while (row <= height) {
    let stars = '';
    let col = 0;
    while (col++ < row)
      stars += '*';
    console.log(stars);
    row++;
  }
}
```

```
starsTriangle(5);
```

Nesting While and For Loops

Problem: Sum of Digits Calculator

- Continuously **read numbers** until "**End**" is entered
  - Print the **sum of digits** for each number
- Finally, print "**Goodbye**"

```
157
99
5
438
End
```

```
Sum of digits = 13
Sum of digits = 18
Sum of digits = 5
Sum of digits = 15
Goodbye
```

Solution: Sum of Digits Calculator

```
function sumOfDigits(inputLines) {
  while (true) {
    let input = inputLines.shift();
    if (input === "End") break;
    let sum = 0;
    for (let num = Number(input);
      num > 0; num = Math.floor(num / 10))
      sum += num % 10;
    console.log(`Sum of digits: ${sum}`);
  }
  console.log("Goodbye");
}
```

```
sumOfDigits([
    157,
    99,
    5,
    438,
    'End'
  ]
);
```

Live Exercises

Practical Problem Solving

Problem: Building

- Write a function to **print a table**, representing a **building**:
  - Odd floors hold **apartments** (type **A**), e.g. **A10**, **A11**, **A12**, …
  - Even floors hold **offices** (type **O**), e.g. **O20**, **O21**, **O22**, …
  - The **last floor** holds large apartments (type **L**), e.g. **L60**, **L61**, **L62**
  - Identifiers consist of: **{type}{floor}{number}**, e.g. **L65**, **A12**, **O24**
  - Example:

```
L60 L61 L62 L63 L64 L65
A30 A31 A32 A33 A34 A35
O20 O21 O22 O23 O24 O25
A10 A11 A12 A13 A14 A15
```

Example: Building

- **Input**: the **count of floors** and the **count of estates per floor**
- **Output**: the building plan (rectangular table of estates)

```
6
4
```

```
L60 L61 L62 L63
A50 A51 A52 A53
O40 O41 O42 O43
A30 A31 A32 A33
O20 O21 O22 O23
A10 A11 A12 A13
```

Solution: Building

```
function building(floors, rooms) {
  for (let f = floors; f >= 1; f--) {
    for (let r = 0; r < rooms; r++)
      if (f === floors) // Print last floor: L{f}{r}
      else if (f % 2 === 0) // Print office: O{f}{r}
      else // Print apartment: A{f}{r}
  }
}
```

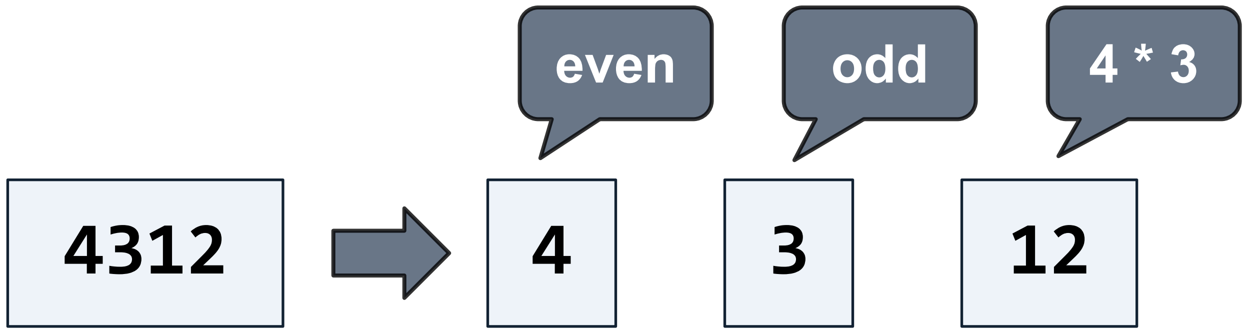
For each line collect the rooms into a variable, them print them together

```
building(6, 4);
```

```
building(5, 3);
```

Problem: Stupid Passwords

- Write a program, which **generates all possible passwords**, consisting of the following 3 parts:
  - The **first** part is an **even** number in the range [2…**n**]
  - The **second** digit is an **odd** number in the range [1…**n**]
  - The **third** is the **product** of the first two
- Example:



Example: Stupid Passwords

- The **input** consists of a single number **n**
- The **output** holds all possible passwords

```
11  →  212 236 2510 2714 2918 21122 414 4312 4520 4728 4936
       41144 616 6318 6530 6742 6954 61166 818 8324 8540 8756
       8972 81188 10110 10330 10550 10770 10990 1011110

5   →  212 236 2510 414 4312 4520

6   →  212 236 2510 414 4312 4520 616 6318
       6530
```

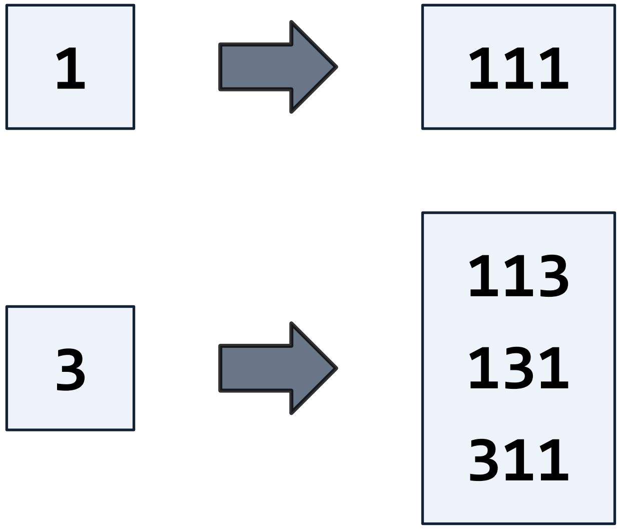
Solution: Stupid Passwords

```
function stupidPasswords(n) {
  let result = '';
  for (let even = 2; even <= n; even += 2) {
    for (let odd = 1; odd <= n; odd += 2) {
      result += `${even}${odd}${even*odd} `;
    }
  }
  console.log(result);
}
```

```
stupidPasswords(5);
```

Problem: Magic Numbers

- Write a function to find all **3-digit magic numbers** of order **n**
  - A number is **magic of order n** if the product of its digits is **n**



Solution: Magic Numbers

```
function magicNumbers(n) {
  for (let d1 = 1; d1 <= 9; d1++)
    for (let d2 = 0; d2 <= 9; d2++)
      for (let d3 = 0; d3 <= 9; d3++)
        if (d1 * d2 * d3 === n)
          // TODO: Print {d1}{d2}{d3}
}
```

```
magicNumbers(5);
```

```
magicNumbers(7);
```

Problem: Travel Savings

- Calculate the **money collection** for multiple travel destinations:
  - Read **destination** and **needed budget** for destination
  - Read many times amounts of collected money, until they are **enough** for the destination (start from 0)
    - Print "`Collected: {sum}`" or "`Going to {destination}`"
  - Read another destination and budget and collect money again
  - A destination "`End`" ends the program

Example: Travel Savings

```
Bali
3500
800
1800
1000
Brazil
4600
5000
End
```

```
Collected: 800
Collected: 2600
Collected: 3600
Going to Bali!
Collected: 5000
Going to Brazil!
```

Solution: Travel Savings

```
function travelSavings(input) {
  let destination;
  while ((destination = input.shift()) != "End") {
    let neededSum = Number(input.shift());
    let collectedSum = 0;
    while (collectedSum < neededSum) {
      collectedSum += Number(input.shift());
      console.log(`Collected: ${collectedSum}`);
    }
    console.log(`Going to ${destination}!`);
  }
}
```

```
travelSavings(
['Bali',
3500, 800, 1800,
1000,
'Brazil',
4600, 5000,
'End']);
```

Problem: Prime Numbers

- Write a function to print all **prime numbers in given range**

```
5
50
```

```
5 7 11 13 17 19 23 29 31 37 41 43 47
```

```
20
30
```

```
23 29
```

Solution: Prime Numbers

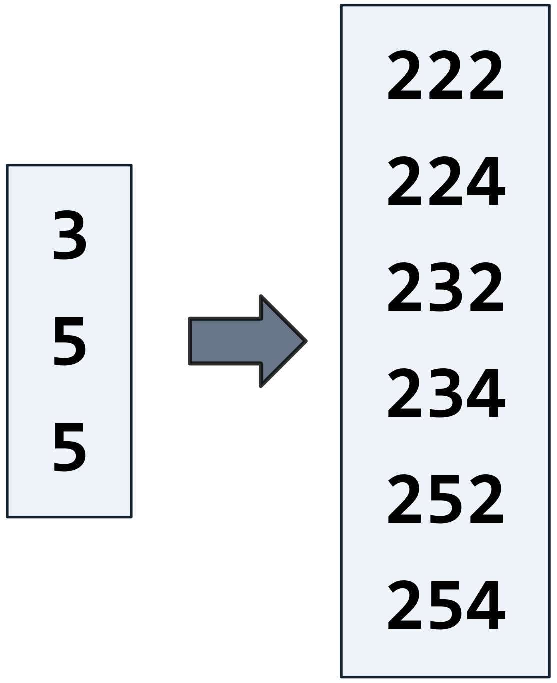
```
function primeNumbers(start, end) {
  for (let num = start; num <= end; num++) {
    let prime = true, divider = 2;
    let maxDivider = Math.floor(Math.sqrt(num));
    while (divider <= maxDivider) {
      if (num % divider == 0) {
        prime = false;
        break;
      }
      divider++;
    }
    if (prime) console.log(num);
  }
}
```

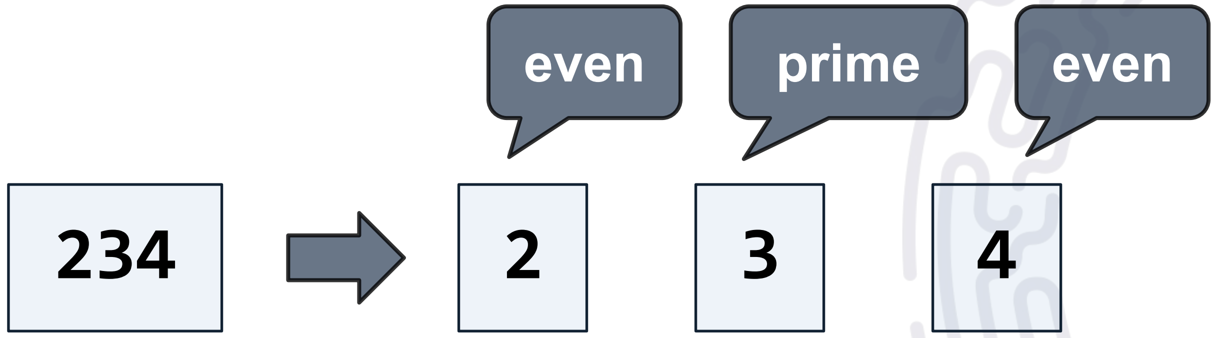
```
primeNumbers(5, 50);
```

Problem: Unique PIN Codes

- Write a function to **generate PIN codes** following certain rules
  - Receives **3 digits**: **max1**, **max2**, **max3** (each is an upper limit)
  - Generates unique 3-digit **PIN codes**, matching the following:
    - Each digit is **within its range**: [1..max1], [1..max2], [1..max3]
    - The **first** and the **third digit** must be **even**
    - The **second digit** must be a **prime number** in the range [**2…7**]
  - Prints the PIN codes in increasing order

Example: Unique PIN Codes





Solution: Unique PIN Codes

```
function pinCodes(max1, max2, max3) {
  for (let d1 = 2; d1 <= max1; d1 += 2)
    for (let d2 = 2; d2 <= max2; d2++)
      for (let d3 = 2; d3 <= max3; d3 += 2) {
        // TODO: Check if d2 is 2, 3, 5 or 7
and
        // print the 3 digits one after another
      }
}
```

```
pinCodes(3, 5, 5);
```

Problem: Letters Combinations

- Write a function to generate all **3-letter combinations** under certain conditions:
  - Receives a start letter **s**, end letter **e** and excluded letter **x**
  - Prints all **combinations of 3 letters** in the range [**s**…**e**], excluding **x**, and their **count**

```
a
c
b
```

```
aaa aac aca acc caa
cac cca ccc 8
```

Solution: Letters Combinations

```
function lettersCombinations(start, end, x) {
  let counter = 0;
  let startChar = start.charCodeAt(0);
  let endChar = end.charCodeAt(0);
  for (let l1 = startChar; l1 <= endChar; l1++)
    for (let l2 = startChar; l2 <= endChar; l2++)
      for (let l3 = startChar; l3 <= endChar; l3++)
        if (l1 !== x && l2 !== x && l3 !== x) {
          // TODO: Convert to char and print the combination
          // TODO: increment the counter++
        }
  // TODO: Print the counter
}
```

```
lettersCombinations
('a', 'c', 'b');
```

Problem: Happy Numbers

- Write a function to generate all **4-digit happy numbers {d1}{d2}{d3}{d4}** for given integer **n**:
  - A number is happy if **d1** + **d2** == **d3** + **d4** == **n**

```
3
```

```
1203 1212 1221 1230 2103 2112 2121
2130 3003 3012 3021 3030
```

```
4
```

```
1304 1313 1322 1331 1340 2204 2213
2222 2231 2240 3104 3113 3122 3131
3140 4004 4013 4022 4031 4040
```

Solution: Happy Numbers

```
function happyNumbers(n) {
  let result = '';
  for (let d1 = 1; d1 <= 9; d1++)
    for (let d2 = 0; d2 <= 9; d2++)
      if (d1 + d2 === n)
        for (let d3 = 0; d3 <= 9; d3++)
          for (let d4 = 0; d4 <= 9; d4++)
            if (d3 + d4 === n)
              result += `${d1}${d2}${d3}${d4} `;
  console.log(result);
}
```

```
happyNumbers(5);
```

Summary

- For-loops can use different steps
  - E.g. i += 2 or i *= 2 or i = i * i
- Nested loops are loops, placed within another loop
  - Nested `for` loops, e.g. process data by rows and columns
  - Nested `while` loops, e.g. nested repeating logic with exit conditions

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