JavaScript 1 - Module 1

Getting Started

Variables

Making Decisions

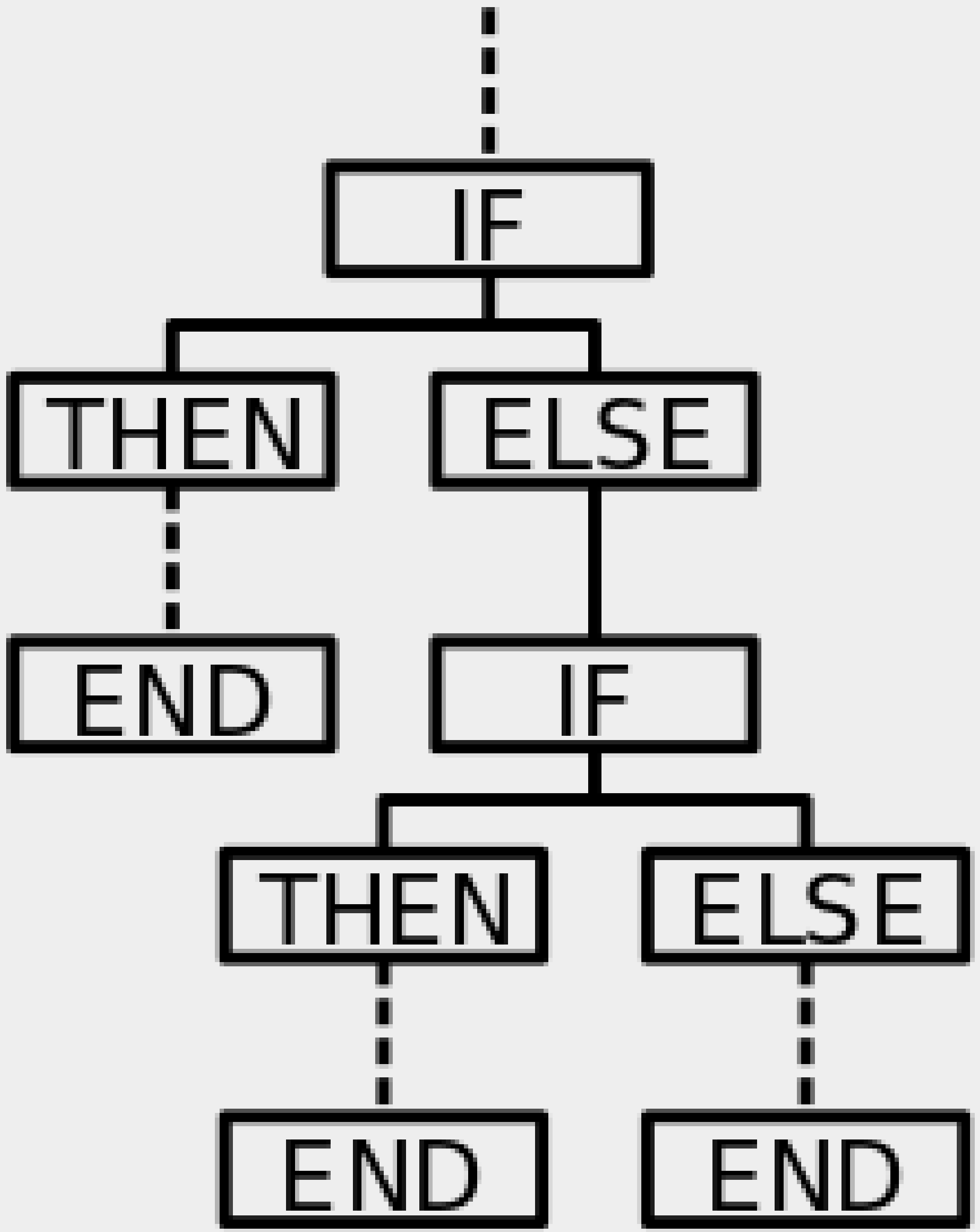
Loops

Solutions for JS1 Lesson 1.2 Variables Exercises

```
// -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Exercise 1
console.log ("Exercise 1");
var firstName = "Lasse";
var lastName = "Hægland";
console.log(firstName + " " + lastName); // "Lasse Hægland"

// -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Exercise 2
console.log ("Exercise 2");
var x = 19;
var y = 23;
var answer = x + y;
console.log(answer); // 42

// -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Exercise 3
console.log ("Exercise 3");
var myBool;
console.log(myBool); // undefined
myBool = true;
console.log(myBool); // true

console.log(!myBool); // false, because ! makes any boolean flip

// -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Exercise 4
console.log ("Exercise 4");
var myText = "Blåbærsyltetøy";
console.log(myText.length); // 14
```

```
// -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Exercise 5
console.log ("Exercise 5");
var a = "19";
var b = 23;
console.log(a + b); // "1923"

console.log(Number(a) + b); // 42

// -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Exercise 6
console.log ("Exercise 6");

var x = 19, y = 23, z = 10;
console.log (x + y * z); // 249

console.log ((x + y) * z); // 420

// -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
// Exercise 7
console.log ("Exercise 7");

console.log(7 % 3); // 1

// -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
console.log("All exercises done...");
```

Making decisions



Making decisions in your code

In any programming language, the code needs to make **decisions** and carry out actions accordingly depending on different inputs.

For example, in a game, if the player's number of lives is 0, then it's game over.

In a weather app, if it is being looked at in the morning, show a sunrise graphic; show stars and a moon if it is nighttime.

Comparison Operators

Comparison and Logical operators are used to test for `true` or `false`.

Comparison operators are used in logical statements to determine equality or difference between variables or values:

```
var x = 5;
(x === 8)   // false
(x === 5)   // true
(x === "5") // false
(x !== 8)   // true
(x !== 5)   // false
(x !== "5") // true
(x > 8)     // false
(x > 5)     // false
(x >= 5)    // true
(x < 8)     // true
(x < 5)     // false
(x <= 5)    // true
```

Source

Operator	Description
==	equal to
===	equal value and equal type (strict equal)
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

`if`

Use the `if` statement to specify a block of JavaScript code to be executed if a condition is `true`:

```
if (condition) {
  // code to run if condition is true
}
```

Make a "Good day" greeting if the hour is less than 18:00:

```
var hour, greeting;
hour = new Date().getHours(); // Use a Date object, and get the hours

if (hour < 18) {
  greeting = "Good day";
}

console.log(greeting);
```

Another `if` example

```
// check if the number is positive

const number = prompt("Enter a number: ");

// check if number is greater than 0
if (number > 0) {
  // the body of the if statement
  console.log("The number is positive");
}

console.log("The if statement is easy");
```

> The `prompt()` method displays a dialog box that prompts the visitor for input. A prompt box is often used if you want the user to input a value before entering a page.

`if`...`else`

Use the `else` statement to specify a block of code to be executed if the condition is `false`.

```
if (condition) {
  // code to run if condition is true
} else {
  // run some other code instead
}
```

If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

Another `if`...`else` example

```
// check if the number is positive or negative/zero

const number = prompt("Enter a number: ");

// check if number is greater than 0
if (number > 0) {
  console.log("The number is positive");
}
// if number is not greater than 0
else {
  console.log("The number is either a negative number or 0");
}

console.log("The if...else statement is easy");
```

Using a `boolean` as condition in an `if...else` statement

```
var myBool = true;
// Change between true and false and see what is logged out

if (myBool === true) {
  console.log("It must be true, when myBool is " + myBool);
} else {
  console.log("It cannot be true, when myBool is " + myBool);
}
```

> Note: In this example, `if (myBool === true)` could be replaced by `if (myBool)` and given the same result.

`if`…`else if`…`else`

Use the `else if` statement to specify a new condition if the first condition is false.

```
if (condition1) {

  // block of code to be executed
  // if condition1 is true

} else if (condition2) {

  // block of code to be executed
  // if the condition1 is false and condition2 is true

} else {

  // block of code to be executed
  // if the condition1 is false and condition2 is false

}
```

`else if` example

If hour is less than 10:00, create a "Good morning" greeting, if not, but hour is less than 18:00, create a "Good day" greeting, otherwise a "Good evening":

```
if (hour < 10) {
  greeting = "Good morning";
} else if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

> If you find that you need more than one (or a few) `else if`'s, then it's often better to use a `switch` statement.

Another `else if` example

```
// check if the number if positive, negative or zero
const number = prompt("Enter a number: ");

// check if number is greater than 0
if (number > 0) {
    console.log("The number is positive");
}
// check if number is 0
else if (number == 0) {
  console.log("The number is 0");
}
// if number is neither greater than 0, nor zero
else {
    console.log("The number is negative");
}

console.log("The if...else if...else statement is easy");
```

Nested if...else Statement

You can also use an `if...else` statement inside of another `if...else` statement. This is known as **nested** `if...else` statement.

```
// check if the number is positive, negative or zero
const number = prompt("Enter a number: ");

if (number >= 0) {
    if (number == 0) {
        console.log("You entered number 0");
    } else {
        console.log("You entered a positive number");
    }
} else {
    console.log("You entered a negative number");
}
```

Source

`switch`

The switch statement is used to perform different actions based on different conditions.

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

`switch`, cont.

This is how it works:

- The switch `expression` is evaluated once.
- The value of the `expression` is compared with the values of each `case`.
- If there is a match, the associated block of code is executed.
    - If multiple cases matches a case value, the **first** case is selected.
- If no matching cases are found, the program continues to the `default` label, and the default code block is executed.
    - If no default label is found, the program continues to the statement(s) after the switch.

The `break` Keyword

When JavaScript reaches a `break` keyword, it breaks out of the switch block.

This will stop the execution of inside the block.

It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.

> **Note**: If you omit the break statement, the next case will be executed even if the evaluation does not match the case.

The `default` Keyword

The default keyword specifies the code to run if there is no case match

`switch` example

```
// getDay() returns 0 (Sunday), 1 (Monday), ..., 6 (Saturday)
var day, today = new Date().getDay();

switch (today) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
     day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}

console.log ("Today is " + day);
```

`switch`, more examples

If today is neither Saturday (6) nor Sunday (0), write a default message:

```
var text, today = new Date().getDay();
switch (today) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
console.log(text);
```

Sometimes you will want different `switch` cases to use the same code. In this example case 4 and 5 share the same code block, and 0 and 6 share another code block:

```javascript
var text, today = new Date().getDay();
switch (today) {
  case 4:
  case 5:
    text = "It will soon be the Weekend!";
    break;
  case 0:
  case 6:
    text = "It's the Weekend!!!";
    break;
  default:
    text = "Looking forward to the Weekend.";
}
console.log(text);
```

`switch`, yet another example

```
var text;
var favouriteDrink = prompt("What's your favorite cocktail drink?");

switch(favouriteDrink) {
  case "Martini":
    text = "Excellent choice! Martini is good for your soul.";
    break;
  case "Mojito":
    text = "Mojito is my favorite too!";
    break;
  case "Cosmopolitan":
    text = "Really? Are you sure the Cosmopolitan is your favorite?";
    break;
  default:
    text = "I have never heard of that one...";
    break;
}

alert(text);
```

Tip:

When comparing strings you may wanna use str.toLowerCase() before comparing, to make sure "mojito" and "Mojito" are treated the same way.

Example:

```
let text = "Hello World!";
let result = text.toLowerCase();
console.log (result); // hello world!

let a = "Hei", b = "hei";
console.log (a == b); // false
console.log (a.toLowerCase() == b.toLowerCase()); // true
```

Strict Comparison

Switch cases use **strict** comparison (`===`). The values must be of the same type to match.

In this example there will be no match for `x` :

```
var text;
var x = "0";
switch (x) {
  case 0:
    text = "Off";
    break;
  case 1:
    text = "On";
    break;
  default:
    text = "No value found";
}
console.log(text); // Will ALWAYS be "No value found"
```

Ternary operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition:

```
variablename = (condition) ? value1 : value2
```

```javascript
var voteable = (age < 18) ? "Too young" : "Old enough";
```

If the variable `age` is a value below 18, the value of the variable `voteable` will be "Too young", otherwise the value of `voteable` will be "Old enough".

Comparing Different Types

Comparing data of different types may give unexpected results.

When comparing a string with a number, JavaScript will convert the string to a number when doing the comparison. An empty string converts to 0. A non-numeric string converts to `NaN` which is always `false`.

```
(2 < 12)        //true
(2 < "12")      //true
(2 < "John")    //false
(2 > "John")    //false
(2 == "John")   //false
("2" < "12")    //false
("2" > "12")    //true ** See next slide
("2" == "12")   //false
```

```
"2" > "12"  //true **
```

When comparing two strings, "2" will be greater than "12", because (*alphabetically*) 1 is less than 2.

To secure a proper result, variables should be converted to the proper type before comparison:

```
var age = prompt("Enter you age, to see if you are old enough to vote");

age = Number(age); // Convert variable age to numeric value

if (isNaN(age)) { // isNaN(value) checks whether the value is NaN
  voteable = "Input is not a number";
} else {
  voteable = (age < 18) ? "Too young" : "Old enough";
}

alert(voteable);
```

Logical Operators

Logical operators are used to determine the logic between variables or values.

```
var x = 6, y = 3;
(x < 10 && y > 1)   // true
(x == 5 || y == 5)  // false
!(x === y)          // true
```

Operator	Description
&&	**and** (both expressions must be true)
||	**or** (at least one expression must be true)
!	**not** (changes the expression to its opposite)

Sources and resourses

MDN:

Making decisions in your code — conditionals

Expressions and operators

Control flow and error handling

W3School:

JavaScript Comparison and Logical Operators

JavaScript if else and else if

JavaScript Switch Statement

Wikipedia:

Decision tree

Todos

GitHub Classroom:

JS1 Lesson 1.3 Making Decisions

Mollify:

Read Making Decisions and do the Lesson Task.