DATA TYPES IN JAVASCRIPT

NORWICH
UNIVERSITY
OF THE ARTS

What Is a Data Type

- It's a way for us, the programmers, to tell our compiler/interpreter how we intend on using data.

- All information in the real world has a type that we have learnt to recognise.
  - 182
  - "I'm learning JavaScript!"
  - A computer is either ON or OFF

NORWICH UNIVERSITY OF THE ARTS

Programming Data Types

Data Type	Meaning	Example
Integer	A whole number	1, 14, 925, 1024
Float	A number with decimal places	0.56, 3.14
Character	A letter, special character, or number	A, %, @, 6
String	A series of characters	I like 2 code!
Boolean	Something that is true of false	true, false

- Of these data types, JavaScript only uses three.
  - **Number** – Both integers and floats are numbers.
  - **String** – Characters are also strings.
  - **Boolean**

- We can use the **typeof** operator to check data types.

NORWICH UNIVERSITY OF THE ARTS

Variables

- JavaScript is **loosely-typed**, meaning we don't need to care about the data type when creating a **variable**.

- In most C-based languages we would create a **Boolean** like so:

```
bool exampleBool = true;
```

- In JavaScript, we use the word **let** no matter the data type:

```
let exampleBool = true;
```

- Whilst you do not need to **define** the type of variable with JavaScript, you will still need to know what types there are.

NORWICH UNIVERSITY OF THE ARTS

Numbers

- Numbers are written using **numerals**:
  - 30 ✓
  - Thirty ✗
- Numbers can be written with **decimals** and **scientific notation**:
  - 30.23
  - 3e6 = 3000000
- Numbers can be **calculated** together using mathematical operators
  - 30+25 = 55

NORWICH UNIVERSITY OF THE ARTS

Strings

- Strings can be expressed using **single** or **double** quotes.
  - 'Ready' ✓
  - "Player" ✓

- Can contain letters, special characters, numbers.
  - "Player 2 has entered the game!"
  - "£22 and 50 pence, please."

- The plus operator (+) can be used to **concatenate** strings.
  - "Hello" + "World" = "HelloWorld"

NORWICH UNIVERSITY OF THE ARTS

Booleans

- Booleans can only have one of two values.
  - **true**
  - **false**

- Numbers can also be Boolean.
  - Any non-0 number is **true**.
  - 0 is **false**.

- Strings can also be Boolean.
  - A string containing anything is **true**.
  - An empty string "" is **false**.

NORWICH
UNIVERSITY
OF THE ARTS

Conversion

- It is possible to easily convert one data type to another.
- This is useful when checking for the value and type to match in a comparison.
- To do this we can use the global type methods:

`String(false);`	Convert a boolean or number to a string.
`Boolean(1);`	Convert a number or string to a boolean.
`Number("13");`	Convert a boolean or string to a number.

NORWICH UNIVERSITY OF THE ARTS

Coercion

- Coercion is when JavaScript automatically converts the type of data to fit the context of the code.

- For example, we can concatenate a number to a string using the + operator. In the below example the number is coerced into becoming a string.

```
"I have " + 2 + "cats.";
```

→

```
"I have 2 cats.";
```

- Coercion works both ways. Strings can be coerced into being numbers, assuming the string only contains a number.

```
"300" - 80;
```

→

```
320;
```

NORWICH UNIVERSITY OF THE ARTS

Objects

- Objects are a collection of properties and methods about a particular thing.
  - Properties are like variables. They contain data about an object.
  - Methods are like functions. They contain things that an object can do.

```
{
  name: "James",
  height: 188,
  weight: 76.5,
  calculateBMI: function() {
    let sqr = this.height * this.height;
    let bmi = (this.weight / sqr) * 10000;
    console.log("BMI: " + bmi);
  }
}
```

NORWICH UNIVERSITY OF THE ARTS

Arrays

- An Array is thought of as a separate data type in other languages but in JavaScript it is an **object**.

- Arrays are used to keep lists of data. Think of it like a table.

- We can create them using square brackets.

```
let dataTypes = ["String", "Number", "Boolean"];
```

Index	Element
0	"String"
1	"Number"
2	"Boolean"

- Data can be accessed from an array by referring to **indexes** of the **elements.**

```
dataTypes[2];
```

→ `"Boolean"`

NORWICH UNIVERSITY OF THE ARTS

OPERATOR LOGIC

NORWICH
UNIVERSITY
OF THE ARTS

Operators

- An operator is a character (or group of characters) that represent an action.

- So far we have already looked at a few operators. Here is a list of them, including some we haven't yet looked at:
  - Mathematical Operators: `+` `-` `/` `*` `%` `++` `--`
  - Assignment Operators: `=` `+=` `*=`
  - Comparison Operators: `==` `!=` `>` `>=` `<` `<=` `&&` `||` `!`
  - **`typeof`**

NORWICH UNIVERSITY OF THE ARTS

Equality and Comparisons

- One feature of most, if not all, programming you will be using a lot is checking whether something is true or not.
  - If it is true, do one thing; if it is not true, do a different thing.

- This table is a list of the comparison operators you will come across in JavaScript.

==	Equality (equal)
!=	Inequality (not equal)
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
===	Strict equality (data type AND value comparison)
!==	Strict inequality (data type AND value comparison)

Truth Tables

- What if we want to check whether multiple statements are true?

&&	And
||	Or

- Truth tables can be used to (hopefully!) better understand how comparison operators work.

AND (&&)	True	False
True	True	False
False	False	False

OR (||)	True	False
True	True	True
False	True	False

NOT (!)	True	False
	False	True

NORWICH UNIVERSITY OF THE ARTS

If Statements

- Now we understand how to get a true or false value out of data, what can we do with this?

- The if statement will allow you to execute a block of code **only if** a value is true.

```
if (true) {
  console.log("This code will execute");
}
```

NORWICH UNIVERSITY OF THE ARTS

If Else Statements

- If a statement is true, we can execute code. What if we want to execute a different block of code if it is not true?
- We could make a second if statement… but things can start to get messy very quickly.
- Fortunately, the else statement can solve this problem for us.

- The code within the else statement will only run when the code within the if statement **does not** run.

```
if (killstreak > 3) {
  console.log(player + " is on a killing spree");
} else {
  console.log(player + " needs more kills");
}
```

- Another important factor for using else over several if's, your code will compute **faster**.

NORWICH UNIVERSITY OF THE ARTS

If Else If

- An if statement can directly follow an else statement, to create an If Else If

```
if (gems == 0) {
  score = 2
} else if (gems == 1) {
  score = 5
}
```

- In short:
  - If **a is true**, do x
  - If **a is false**, check if **b is true**. If it is, do y
  - If **both a and be are false**, do nothing

NORWICH UNIVERSITY OF THE ARTS

Ternary Operator

- The ternary operator (?) is a quick **if/else** statement that has an output.

```
let score = (gems > 1) ?  5 : 2;
```

- In this example **score** would be set to **5** if gems is greater than 1, but set to **2** if it’s not.

- You aren’t limited to just numbers. You can have anything on either side of the colons!

```
let fruit = (apples > 0) ? apples : "No fruit";
```

NORWICH UNIVERSITY OF THE ARTS

Switch Statements

- A switch is a big chain of if/else statements neatened out.

- In this example enemyHP will change based on the difficultyLvl:

difficultyLvl	enemyHP
1	5
2	8
Anything else	3

```
switch(difficultyLvl) {
  case 1:
    enemyHP = 5;
    break;
  case 2:
    enemyHP = 8;
    break;
  default:
    enemyHP = 3;
}
```