JavaScript FUNdamentals

Types and operators

Six basic types in JavaScript

- Booleans
- Numbers
- Empty Values
- Strings
- Arrays
- Objects

⚡ If you are ever unsure of the type of a given value, you can use **typeof <VALUE>**

Strings: Interpolation

```
let pi = 3.14;
let diameter = 30;
let radius = diameter / 2;

// declare a string
let introduction = `The area of a circle is πr².`

// declare a string with interpolation
let example = `A ${diameter}cm pizza has an area of ${pi * radius * radius}cm².`

// Concatenate the strings
let text = introduction + ' ' + example

console.log(text);
```

```
> The area of a circle is πr². A 30cm pizza has an area of 706.5cm².
```

Arrays: Accessing the values in an array

⚠️⚠️ Indexing starts at 0 ⚠️⚠️

```
let anArray = ['bacon', undefined, 900, true]
```

1. What is the value of anArray[0]?	**‘bacon’**
2. How do we access the value 900?	**anArray[2]**

Objects: Accessing the values in an object

Values in an object can be accessed with

- dot notation
- bracket notation

```
let person = { name: 'Bob', age: 23 };
```

1. How do we access ‘Bob’? **person.name** or **person[‘name’]**

JavaScript Operators

- Arithmetic
- Comparison operators
- Logical operators

Comparison operators and logical operators are usually combined in an expression to create a boolean value, like this **2 > 1 && -1 < 0** which is **true**.

Exercise 1

Look at these expressions below and determine whether they evaluate to true or false

true || false	
false && false	
1 < 2 && 2 > 1	
31 < 13 || 1 < 2 && 3 > 1	
400 <= 400 && 399 < 400 && (30 > 31 || 400 > 31)	
true && false && false || false && true	
true && false || true || false	

Exercise 2

Given this data structure:

```
let data = [0, [], [], [1, 2, 3, [4]]];
```

How would you access the value \`0\`?	
How would you access the value \`3\`?	
How would you access the value \`4\`?	

Exercise 3

```
{ label: 'corn', price: 5.3 + '$' };
{ ISBN: 53532, isAvailable: true, author: 'Nakamoto' };
```

- List the number of properties for each object.
- For each property, indicate its key and its value.
- For each property value, indicate its type.

Exercise 4

```
// Given
let person = { name: 'Bob', age: 23 };
```

What is the value of the following expressions?

person.name	
person[‘name’]	
person[name]	

Exercise 5

```
// Given
let person = { name: 'Bob', age: 23 };
```

What is the value of the following expressions?

person.key	
person[‘key’]	
person[key]	

JavaScript FUNdamentals

Variables

JS Variables (or bindings)

- Variables are used to hold data values.
- Variables are essentially data containers.
- They can also hold functions
- There are 3 ways to declare variables.

```
var name = 'Rick';

const profession = 'Scientist';

let grandson = 'Morty';
```

Defining Variables

- Use either **const** or **let**
  - **const** for variables that *will never change*.
  - **let** for variables that *may change*.

- Never use **var**. It really should be deprecated…

Defining Variables

Naming conventions

- Can be just about anything
- Cannot contain spaces.
- Can contain numbers but cannot start with a number.

⚡ Make your variable names *meaningful.* This will ease the debugging process considerably.

Defining Variables

Camel Case, underscores and dashes

● Camel Case:	myVariableName
● Underscores:	my_variable_name or MY_VARIABLE_NAME
● Dashes:	**doesn't work! Will definitely break your code.**

⚡ In the course, we use **camelCase** most of the time.

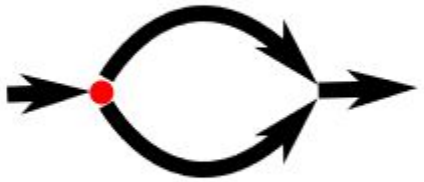
JavaScript FUNdamentals

Control Flow

JS Control Flow

JavaScript executes a program from start to finish, in order.

We will often need a program to do things *conditionally.*



The if statement

The **condition** must be true for the code block to run.

```
if (condition) {
  // do something
}
```

Example 1

```
if (number > 16) {
  console.log('Hold');
}
```

Example 2

```
let allegiance = '';
if (forceUseGood === true) {
  allegiance = 'jedi'
} else {
  allegiance = 'dark side'
}
```

Exercises (5 min)

Go to codesandbox...

```
// 1. If it rains, I stay home.
let currentWeather = 'rainy';



// 2. If I am hungry, I eat.
let hunger = true;



// 3. If it's 10pm, I go to bed. If not, I write code.
```

Problem

Here is a program that outputs all even numbers between 0 and 12.

```
console.log(0);
console.log(2);
console.log(4);
console.log(6);
console.log(8);
console.log(10);
console.log(12);
```

This is tedious, but manageable.

What if we wanted to output all of the even number between 0 and 1000?

Solution (loops)

This is where loops come in!

The while loop

Output all of the even numbers between 0 and 1000

```
let number = 0;
while (number <= 1000) {
  console.log(number);
  number = number + 2;
}
```

1. number is set to 0.
2. while loop condition is true (0 < 1000), so it
   a. Outputs the number
   b. Adds 2 to the number
3. number is now 2
4. repeats until number is 1002.

The while loop

Another example of a while loop...

Write a program that outputs 2 to the power of 10.

The while loop

while is a great tool for some situations, but it can be a little tedious.

We need to

1. Define a counter
2. Define a condition for the loop to run
3. Increment the counter everytime the loop runs.

```
let counter = 0;
let value = 2;
while (number <= 1000) {
  console.log(number);
  number = number + 2;
}
```

The for loop

Let’s convert that last example to use a **for** loop.

```
let counter = 0;
let value = 2;
while (number <= 1000) {
  console.log(number);
  number = number + 2;
}
```

```
for (let i = 0; i <= 1000; i++) {
  If (i % 2 === 0) {
    console.log(number);
  }
}
```

⚠️ The **only** way to break out of a **for** loop is for the condition to be false. ⚠️

The for loop

⚠️ The **only** way to break out of a **for** loop is for the condition to be false. ⚠️

- If you write a **for** loop that *always* evaluates to true, your loop will continue forever.
- This is a bad thing... it will crash your environment (browser window, node env, etc.).
    - To fix it, you will need to force-quit the environment.

Exercises!

Work in pairs to solve the three exercises.

You have 20 minutes.