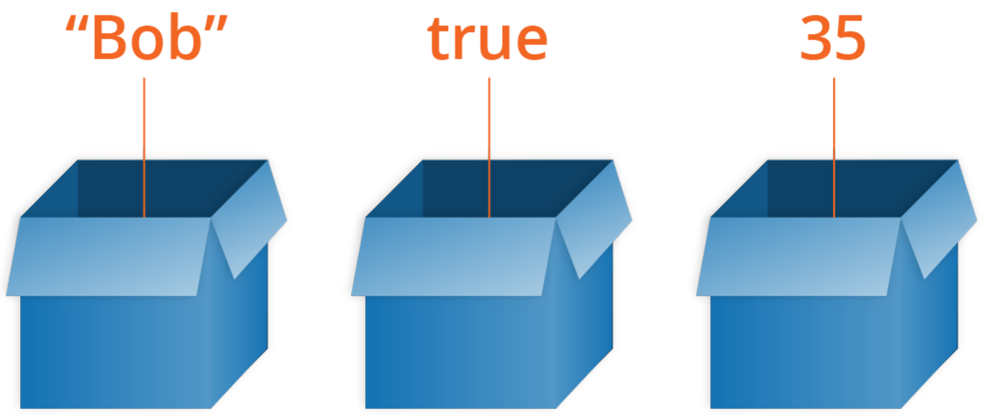
OIM3690 - Web Technologies

Variables

Variables in JavaScript

- A variable is a container that holds a value.
    - It can be used to store a web page element, a property, or a number.
- Note:
    - Variables aren't the values themselves; they are containers for values.
    - You can think of them like little cardboard boxes that store things.



Declaring Variables using `let` and `const`

- Creating a variable in JavaScript is called "**declaring**" a variable
- Use `let` when declaring the variable, if variable's value will change
  - Example:

```
let x = 10;
//Some JS statements
x = 20;
```

- Use `const` to declare variables with a constant value.
  - Example:

```
const COLUMNS = 80;
// ...
COLUMNS = 120; // Uncaught TypeError: Assignment to constant variable.
```

- **DO NOT** use `var`

DO NOT use `var`!

- `var` can cause confusion with variable hoisting and scope.
- `var` is an old keyword that is not recommended to use in modern JavaScript.

Naming Convention

- You can name a variable anyway you want
  - just do not use "reserved" words
    - E.g. avoid naming a variable as `"form"` or `"element"` or `"backgroundColor"`
- Local variable names are written in ***lowerCamelCase***
- Constant names use ***CONSTANT_CASE***
  - Recommended by the [Google JavaScript Style Guide]

Arithmetic Manipulation of Variables

-
  ```
  let x;
  // Declare a variable x.
  x = 10;
  //Assign the value of 10 to variable x
  ```

-
  ```
  let x = 10;
  // Declare a variable and assign it a value of 10
  // (both declaration and assignment in the same one step)
  x = x + 10;
  // Add 10 to the value that is in variable x and store the result in x
  x = x * 5;
  x = x / 5;
  ```

Basic Data Types in JavaScript

- JavaScript is a dynamically-typed language
    - Meaning that variables can hold different types of values.
- Some common data types:
    - **Numbers**: including integers and floating-point numbers.
    - **Strings**: enclosed in either single or double quotes.
    - **Booleans**: either `true` or `false`
    - **Null**: represent the intentional absence of any object value.
    - **Undefined**: represent variables that have not been assigned a value.
    - **Objects**: represent complex data structures, such as arrays and functions.
    - **Symbols**: used to create unique identifiers for object properties.

Arithmetic Operators

- Some common arithmetic operators
    - Addition (`+`)
    - Subtraction (`-`)
    - Division (`/`)
    - Multiplication (`*`)
    - Remainder (`%`)
    - Exponentiation (`**`)
    - Increment (`++`)
    - Decrement (`--`)
- **Practice**: Play with all the arithmetic operators in Concole of DevTools.

Handling Strings

- Example:

```
let x = "My name is Michael"; // note the quote
// Strings are always placed within quotes
let y = " Scott";
let name = x + y // What will the result of this addition be?
let age = 21;
let message = "I am " + age + " years old."; // "I am 21 years old."
```

- Using the `+` operator with a **string** will concatenate (join) the two values together.
  - If one of the two is a **string**, then the other value will be converted into a **string**.
- If both values are numbers, the `+` will perform a regular **addition** operation.

Functions

Functions in JavaScript

- A **function** is a self-contained block of code that performs a specific task.
  - Used to organize code and make it easier to reuse.
  - In JavaScript, think of a function as a set of instructions to the browser to do something.
- We will be creating our own functions.
  - In JavaScript, a function is defined using the `function` keyword, followed by a name, a list of parameters, and the code to be executed.
- JavaScript also has **pre-defined functions**
  - Global functions
  - From Web APIs
    - E.g. window.alert() method: `window.alert("Hello world!");`
    - Technically they are called ***methods***.

Write Our Own Functions

- Syntax:

```
function functionName(arguments) {
    // JavaScript statements;
}
```

- Start with `function` keyword
- The function body is enclosed in curly braces `{}` and contains statements that define the function's behavior.
- The `arguments` list is required
  - It can be empty (just the parentheses) or contain one or more arguments separated by commas.
  - Arguments are the values that we pass to the function when we call it.
  - Functions can also return values using the `return` keyword.

Exercise: *ex14.html*

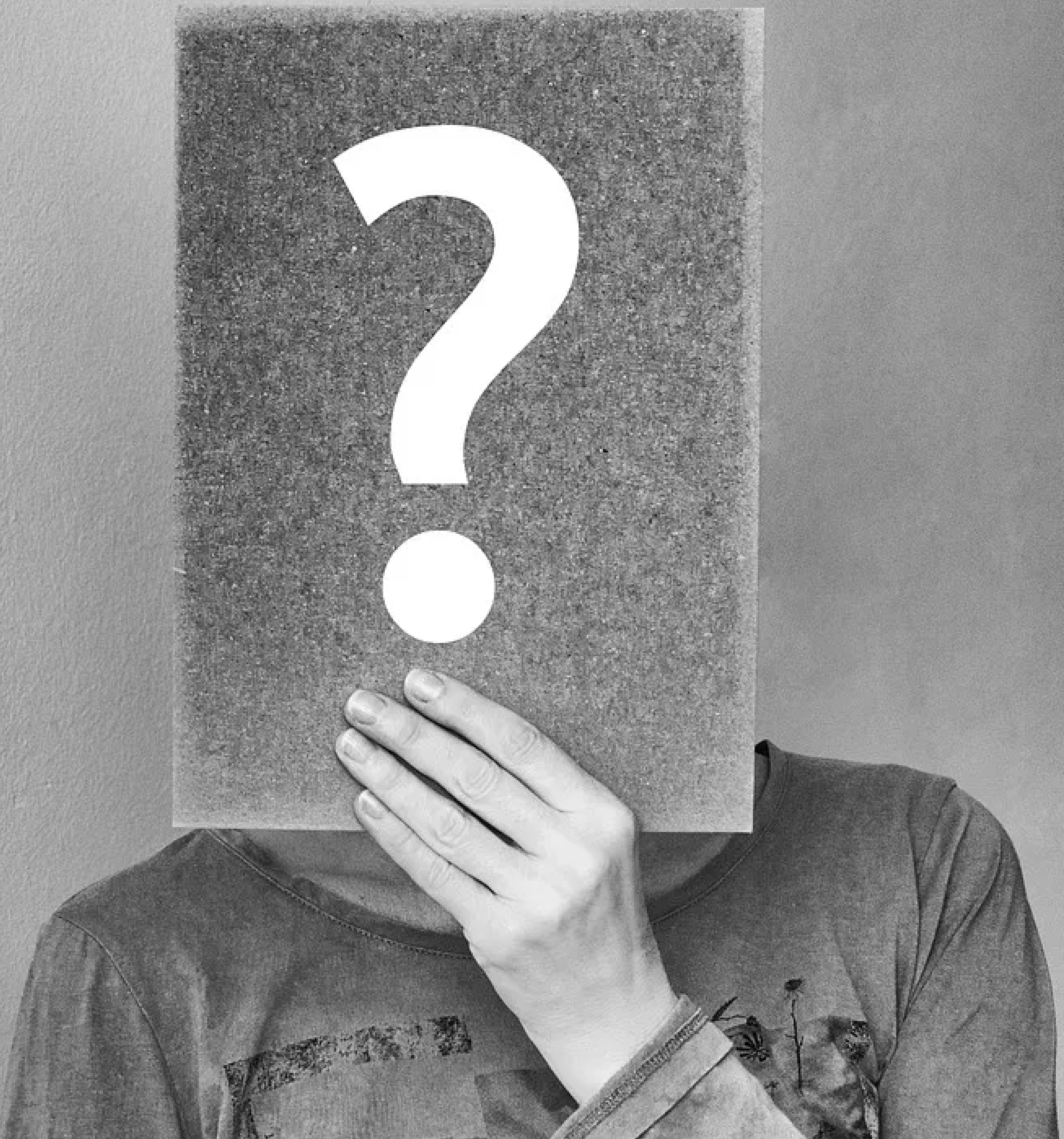
- Download *ex14.html* from GitHub (*OIM3690/resources/templates*).
- Write a **function** that will enlarge the image when user moves mouse ***over*** the image.
    - How do we define this function in `script`?
    - What is the ***event***? ***Element*** (*eventTarget*)?
    - How do we change the image **size** using JavaScript?
    - Let's write **pseudo-code** together.
    - Besides enlarging the image, can you also change something else in the same function?

Exercise: *ex14.html* (cont.)

- Write another function that will resize the image to original size when user moves mouse ***off*** it.
- Update ***sitemap.html*** and **commit/push** to GitHub

Exercise: *ex14-2.html* (Optional)

- Same as ex14.html, but you need to use same functions to work with **multiple** images.
  - Need to add ***arguments*** to functions.
  - This could be very confusing if you don't understand the purpose of arguments.
- Update ***sitemap.html*** and **commit/push** to GitHub



Questions?