Module 4 Day 1

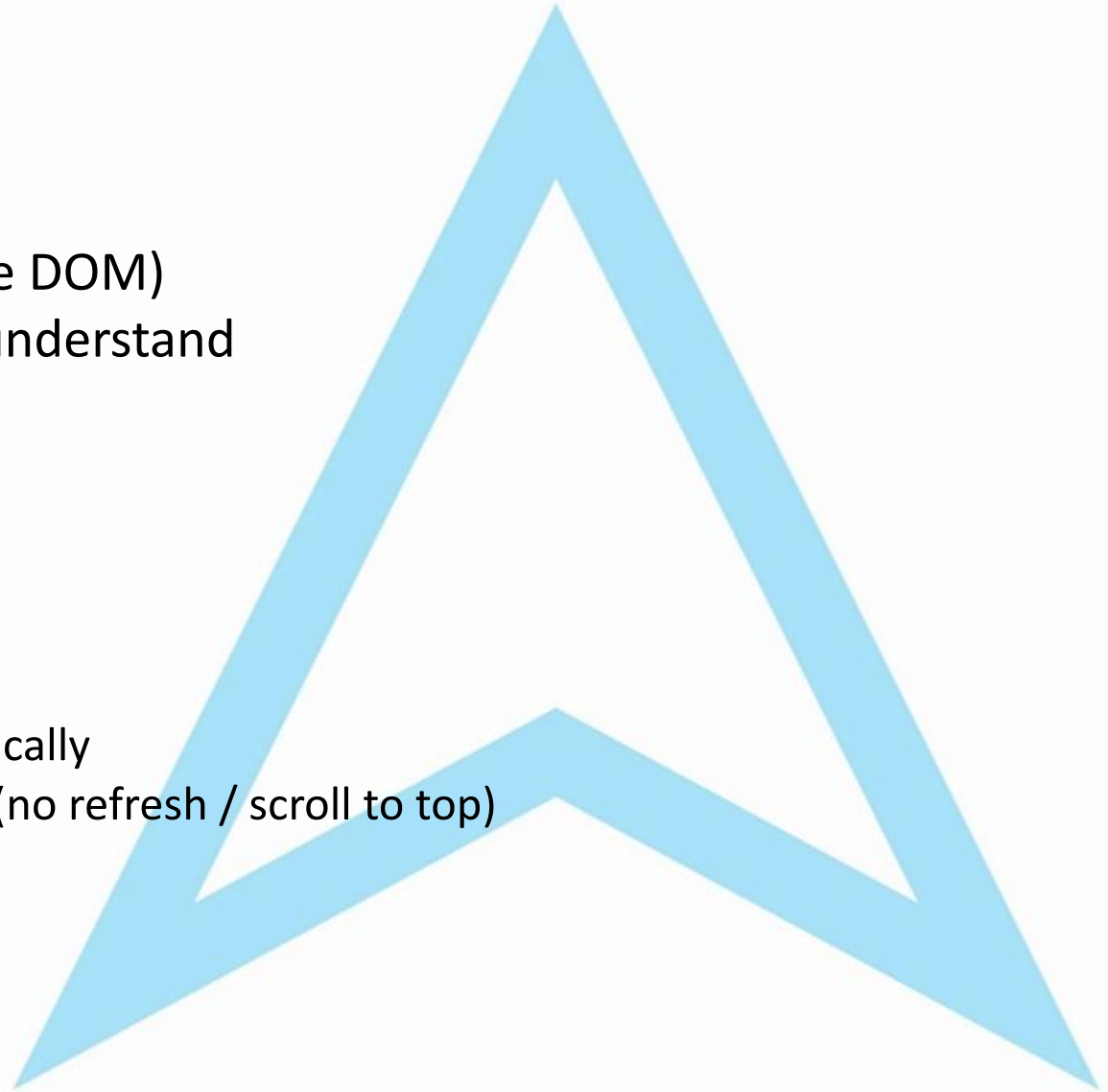
Client-side Scripting with JavaScript

Module 4, Flipped

- No pairs exercises during Module 4
- Afternoon / evening before
  - Reading
  - Student_tutorial
  - Quiz (on reading / tutorial subject)
- Class day
  - Lecture, informed by quiz
  - Exercises

Client-side Scripting

- What
  - Code executes in the browser
  - Interacts with the HTML of the page (the DOM)
  - **JavaScript** is the language all browsers understand
- Why
  - Validation on the client
    - Immediate response to the user
    - Fewer posts to the server
  - Page interaction
    - Show or hide sections of the page dynamically
    - Dynamically update portions of the page (no refresh / scroll to top)
  - Event responses (button push)

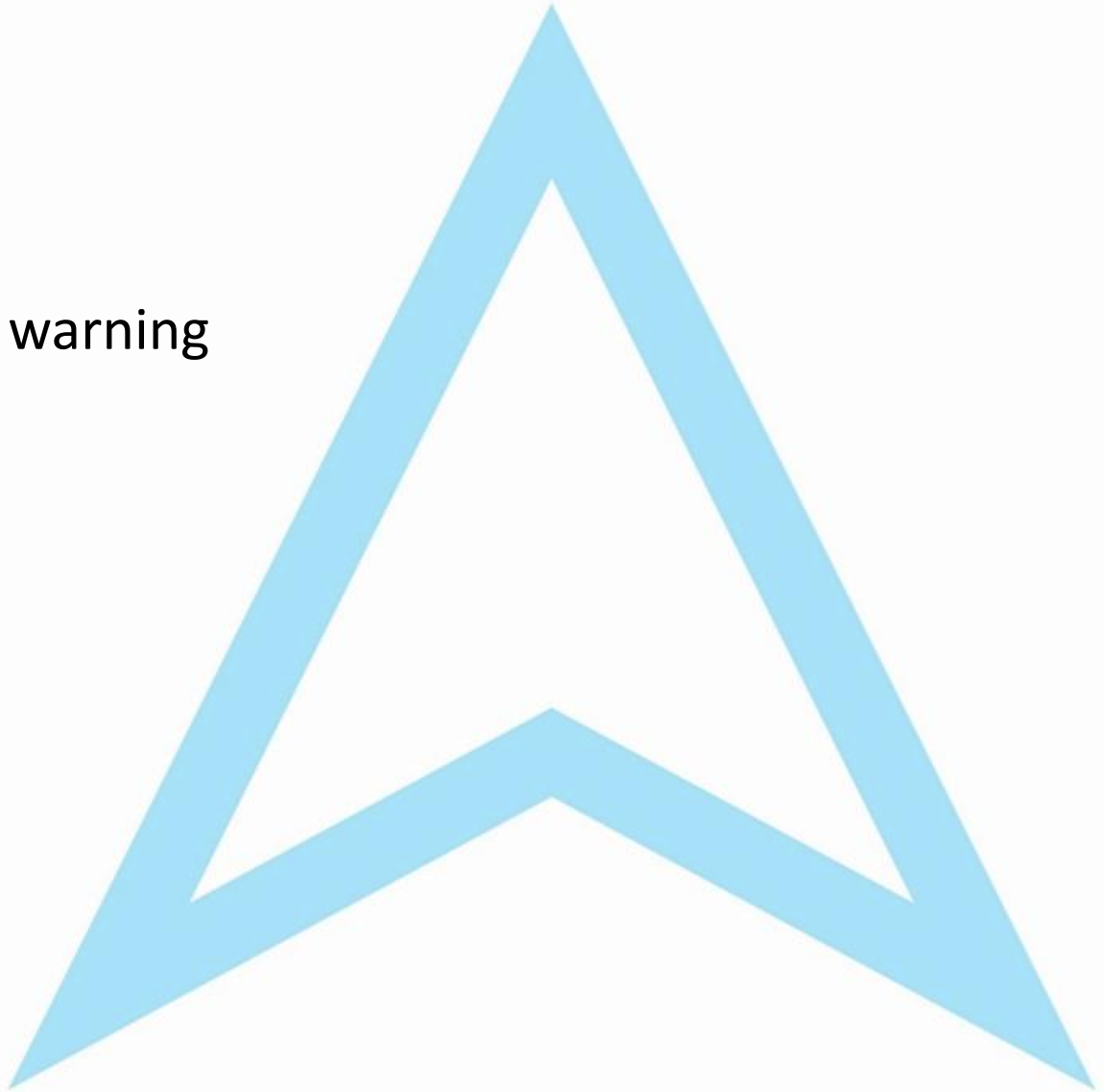


Client-side Responsibilities

- **HTML** provides the **content**
  - Server may dynamically generate the HTML
- **CSS** provides the **style**, or look-and-feel
  - Static style sheets usually provide the style
- **JavaScript** provides the **behavior** and user-interaction
  - Interacts with the HTML page, and may interact with the server

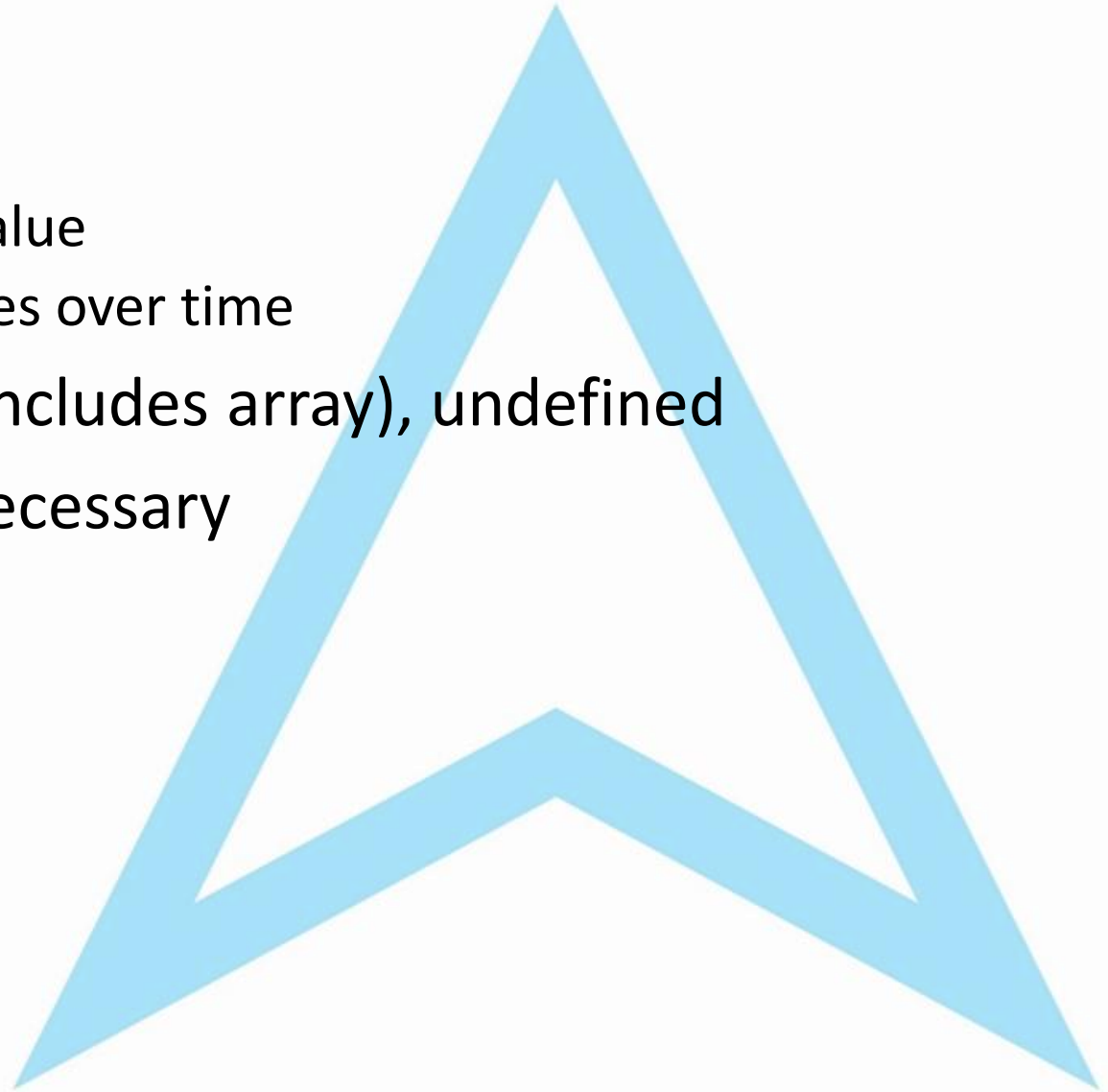
Variables

- let and const
- var – Do not use (best practice)
  - Allows multiple declarations without warning
  - Function scope (vs. block scope)
- Names
  - Usually camel-case in JavaScript
  - Const may be upper-case



Data Types

- Loosely typed
  - JavaScript infers the type from the value
  - Same variable can hold different types over time
- Number, String, Boolean, Object (includes array), undefined
- JavaScript does type coercion as necessary



Strict and loose equality

- === vs. ==
- === means types and values are equal (strict equality)
- == means values are equal (loose equality)
  - Types are coerced
- !== and != are the "not equal" equivalents
- **Falsy** values:
  - When coerced to Boolean, value is false
  - false, 0, "", null, undefined, NaN
  - All other values are **Truthy**
- More craziness: https://codeburst.io/javascript-double-equals-vs-triple-equals-61d4ce5a121a

Null vs. Undefined

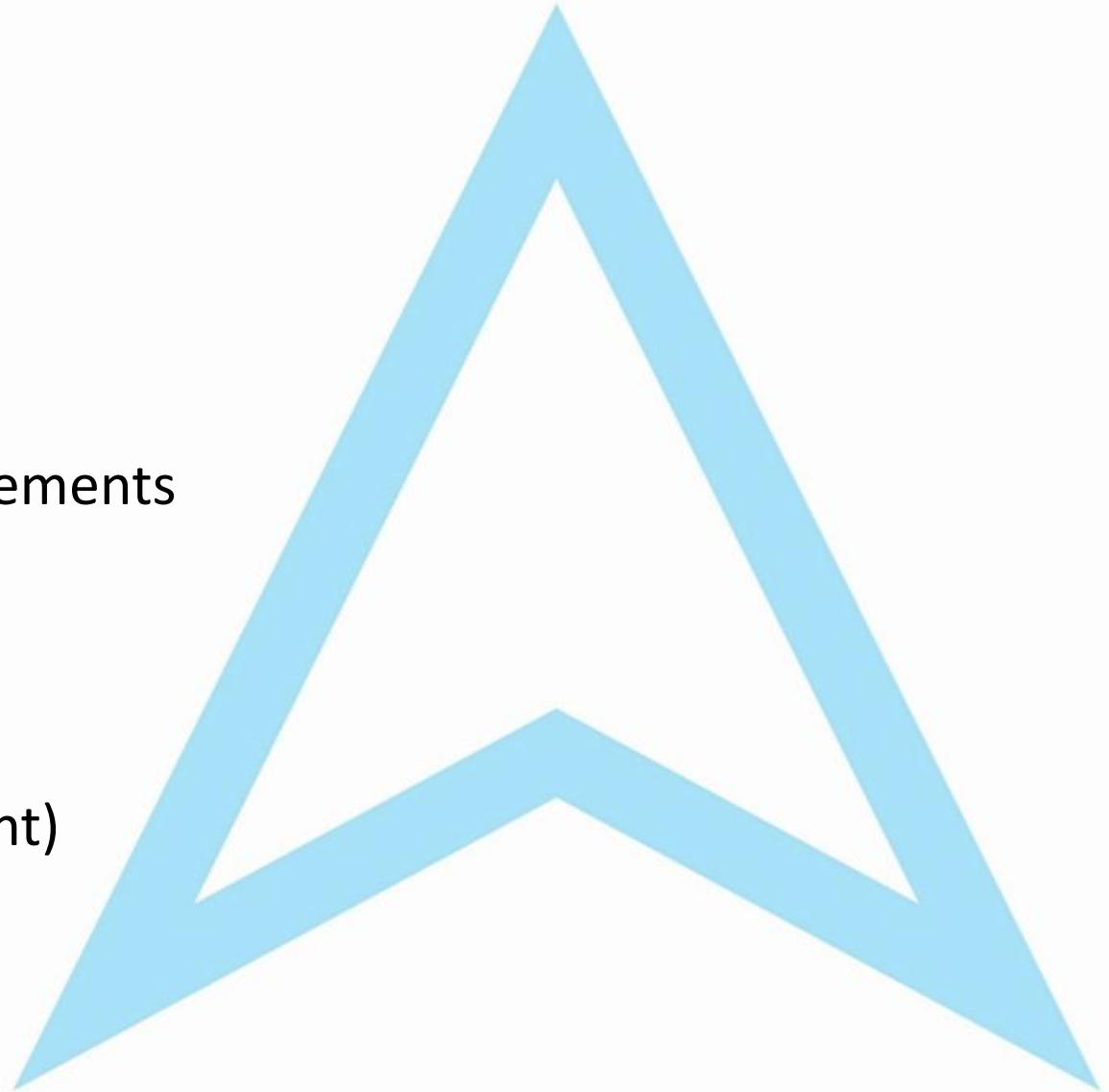
- null is a value of type *Object*
- undefined is a value of type *undefined*
- null must be assigned. It means *nothing*.
- undefined occurs from the "let var_name;" statement
  - It also may be assigned
- null !== undefined, but null == undefined

Branching

- if (condition) {
  // do something
  }
- if (condition) {
  // do something
  } else {
  // do something else
  }
- if (condition1) {
  // do this
  } else if (condition2) {
  // do that
  } else {
  // do the other
  }
- **switch** also works

Arrays

- let scores = [];
  let grades = [90, 80, 70];
  scores[10] = 100;
- Size is changeable!
  - scores.length gives the number of elements
- Functions:
  - push, pop (end of array)
  - unshift, shift (beginning of the array)
  - indexOf, lastIndexOf (finds an element)



Loops

- for (let i = 1; i <= 5; i++) {
  console.log("Hello World!");
  }
- while and do also exist
- No foreach!
  - (at least, not yet ☺)

Objects

- { } denotes an object
- Key : value pairs, separated by commas

```
const person = {
  firstName: "Bill",
  lastName: "Lumbergh",
  age: 42,
  employees: [
    "Peter Gibbons",
    "Milton Waddams",
    "Samir Nagheenanajar",
    "Michael Bolton"
  ]
};
```

Functions (think methods)

- function keyword
- Function name
  - Usually camel-case
- No return type
- Parameter names
  - No type defined
- return statement

JavaScript vs C#

	JavaScript	C#
Variables	let age = 35; • Also, const, var (don't use)	string age = 35;
Data Types	Number, String, Boolean, Symbol, Object, null, undefined	Int, short, long, float, decimal, single, double, string bool…
Strings	Single or double quotes	Double quotes
Undefined data type	"undefined" is both a datatype, and a value	Not applicable
String interpolation (template literals)	`My name is ${firstname} ${lastname}`	$"My name is {firstname} {lastname}"
Equality Comparison	==, ===, !=, !===	==, !=
Array	Size can change Multiple types Directly address even new elements	Size fixed and set when allocated Single data type Must address in-bounds
Functions / methods	function myFunction(param1)	public int myMethod(string param1)