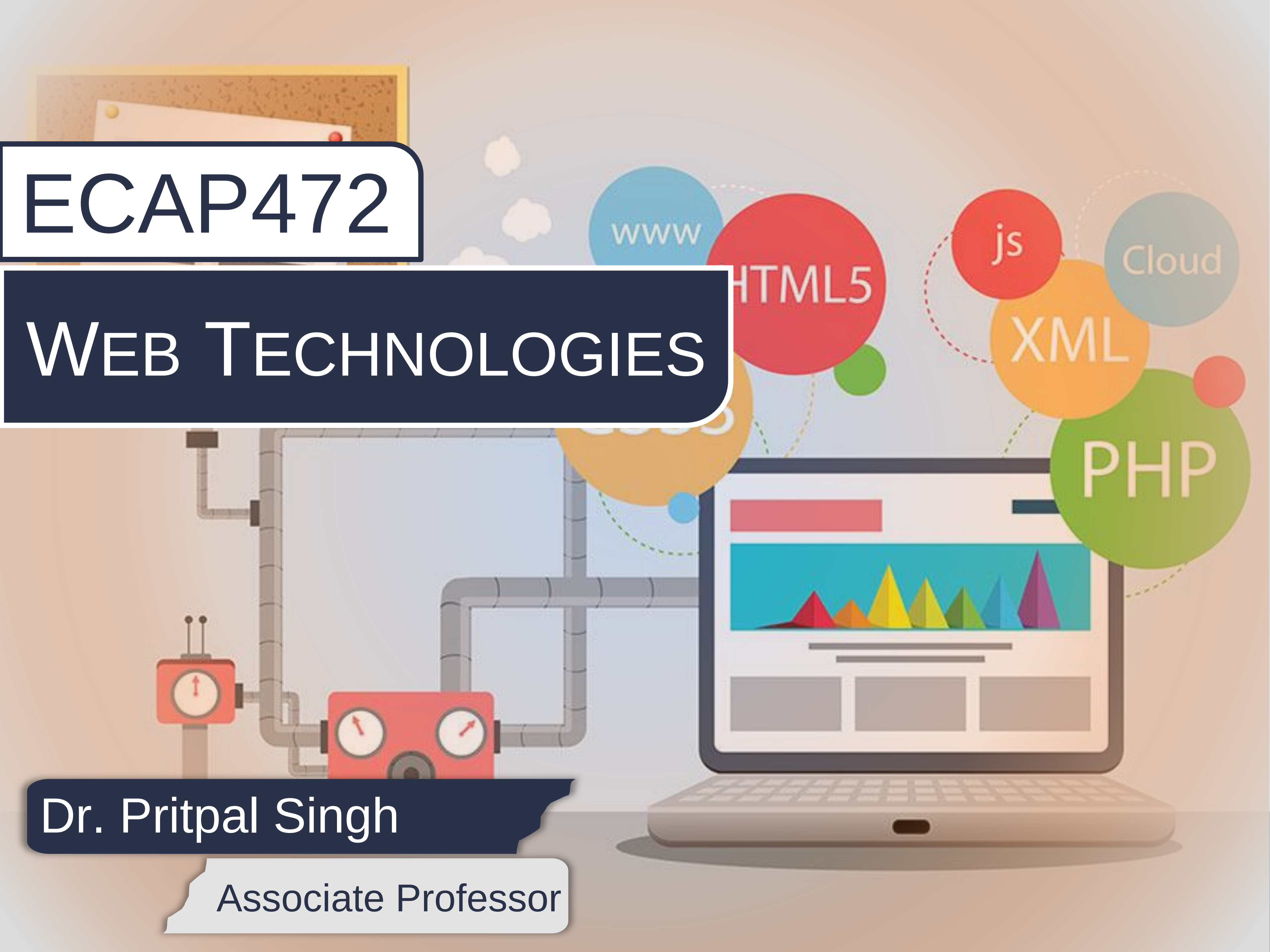
ECAP472

Web Technologies



Dr. Pritpal Singh

Associate Professor

Learning Outcomes

After this lecture, you will be able to

- understand concept of displaying JavaScript Objects.
- understand concept of loops in JavaScript.

Objects

- In JavaScript, an object is a standalone entity, with properties and type.
- Compare it with a cup, for example. A cup is an object, with properties. A cup has a color, a design, weight, a material it is made of, etc. The same way, JavaScript objects can have properties, which define their characteristics.

JavaScript Objects

- In JavaScript, objects are king. If you understand objects, you understand JavaScript.
- In JavaScript, almost "everything" is an object.
- Booleans can be objects (if defined with the new keyword)
- Numbers can be objects (if defined with the new keyword)

JavaScript Objects

- Strings can be objects (if defined with the new keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects

How to Display JavaScript Objects?

Some common solutions to display JavaScript objects are:

- Displaying the Object Properties by name
- Displaying the Object Properties in a Loop
- Displaying the Object using Object.values()
- Displaying the Object using JSON.stringify()

Displaying Object Properties

Example

```
const person = {
  name: "John",
  age: 30,
  city: "New York"
};
document.getElementById("demo").innerHTML =
person.name + "," + person.age + "," + person.city;
```

Displaying the Object in a Loop

```
const person = {
  name: "John",
  age: 30,
  city: "New York"
};
let txt = "";
for (let x in person) {
txt += person[x] + " ";
};
document.getElementById("demo").innerHTML = txt
```

You must use **person[x]** in the loop.
person.x will not work
(Because **x** is a variable).

Using Object.values()

Any JavaScript object can be converted to an array using Object.values():

```
const person = {
  name: "John",
  age: 30,
  city: "New York"
};
const myArray = Object.values(person);
```

myArray is now a JavaScript array, ready to be displayed

JavaScript For Loop

- Loops can execute a block of code a number of times.
- JavaScript Loops
- Loops are handy, if you want to run the same code over and over again, each time with a different value.

Often this Is The Case When Working with Arrays

Instead of writing:

- text += cars[0] + "<br>";
- text += cars[1] + "<br>";
- text += cars[2] + "<br>";
- text += cars[3] + "<br>";
- text += cars[4] + "<br>";
- text += cars[5] + "<br>";

You Can Write

```
for

(let i = 0; i < cars.length; i++)

 {

  text += cars[i] + "<br>";

}
```

Different Kinds of Loops

JavaScript supports different kinds of loops:

- for - loops through a block of code a number of times
- for/in - loops through the properties of an object
- for/of - loops through the values of an iterable object

Different Kinds of Loops

JavaScript supports different kinds of loops:

- while - loops through a block of code while a specified condition is true
- do/while - also loops through a block of code while a specified condition is true

The For Loop

The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

Example

-
  ```
  for (let i = 0; i < 5; i++) {
    text += "The number is " + i + "<br>";
  }
  ```
- From the example above, you can read:
- Statement 1 sets a variable before the loop starts (let i = 0).
- Statement 2 defines the condition for the loop to run (i must be less than 5).
- Statement 3 increases a value (i++) each time the code block in the loop has been executed.

Statement 1

- Normally you will use statement 1 to initialize the variable used in the loop (let i = 0).
- This is not always the case, JavaScript doesn't care. Statement 1 is optional.

Statement 2

- Often statement 2 is used to evaluate the condition of the initial variable.
- This is not always the case, JavaScript doesn't care. Statement 2 is also optional.
- If statement 2 returns true, the loop will start over again, if it returns false, the loop will end

Statement 3

- Often statement 3 increments the value of the initial variable.
- This is not always the case, JavaScript doesn't care, and statement 3 is optional.
- Statement 3 can do anything like negative increment (i--), positive increment (i = i + 15), or anything else.
- Statement 3 can also be omitted (like when you increment your values inside the loop)

JavaScript While Loop

Loops can execute a block of code as long as a specified condition is true.

Syntax

```
while (condition) {
  // code block to be executed
}
```

Example

In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:

Example

```
while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

The Do While Loop

The do while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

Syntax

```
do {
  // code block to be executed
}
while (condition);
```

Example

```
do {
  text += "The number is " + i;
  i++;
}
while (i < 10);
```

Comparing For and While

The loop in this example uses a for loop to collect the car names from the cars array

Example

```
const cars = ["BMW", "Volvo", "Saab", "Ford"];
let i = 0;
let text = "";

for (;cars[i];) {
  text += cars[i];
  i++;
}
```

Comparing For and While

The loop in this example uses a while loop to collect the car names from the cars array

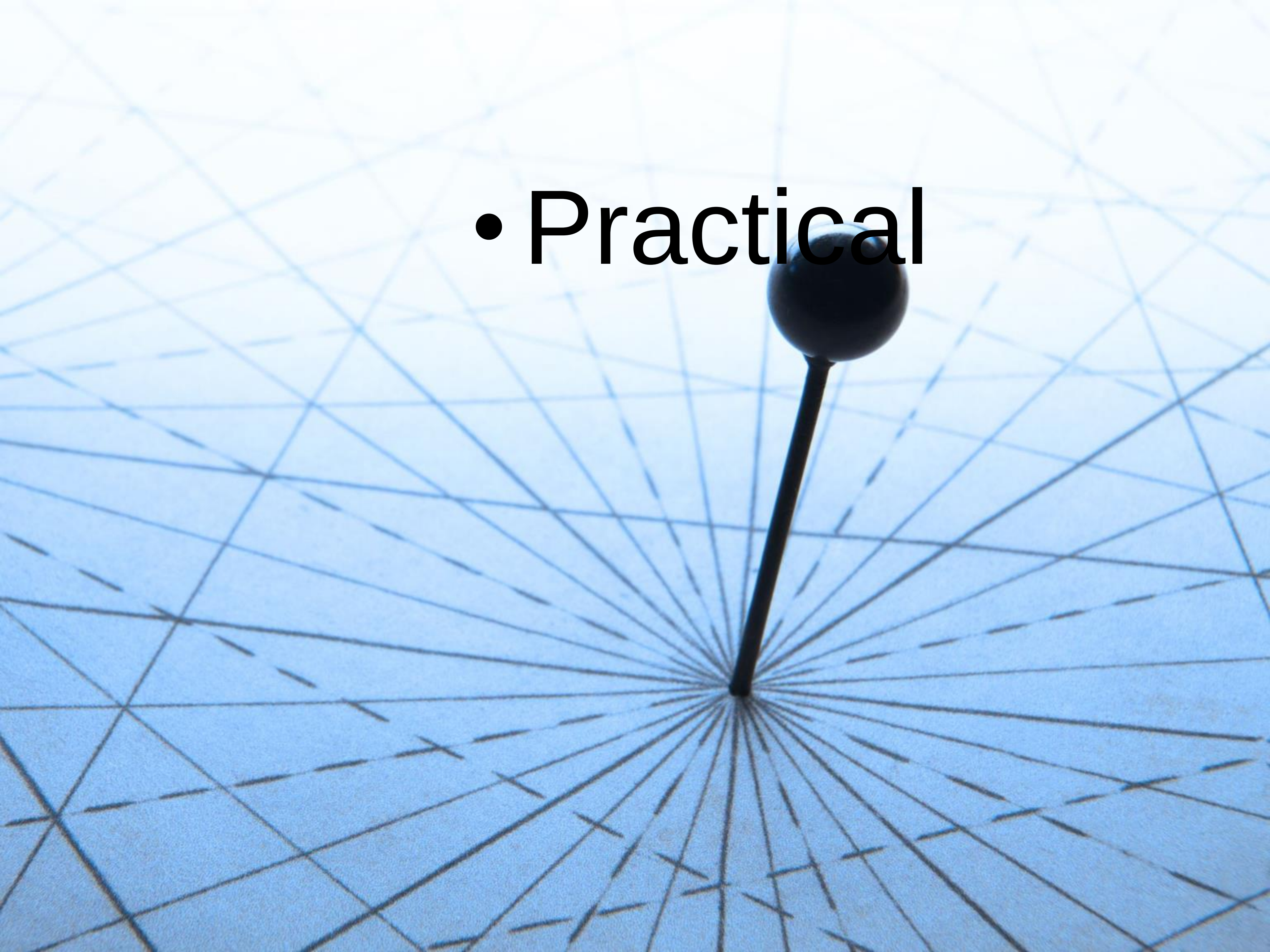
```
const cars = ["BMW", "Volvo", "Saab", "Ford"];
let i = 0;
let text = "";

while (cars[i]) {
  text += cars[i];
  i++;
}
```

JavaScript Break and Continue

- The break statement "jumps out" of a loop.
- The continue statement "jumps over" one iteration in the loop.

- Practical



That's all for now…