EXAMPLE 17: ADDING PROPERTIES TO OBJECTS DYNAMICALLY

EXAMPLE 17: ADDING PROPERTIES TO OBJECTS DYNAMICALLY

"AN OBJECT IS A SET OF KEY-VALUE PAIRS, WHERE THE VALUES CAN ALSO BE FUNCTIONS"

RECAP

"AN OBJECT IS A SET OF KEY-VALUE PAIRS, WHERE THE VALUES CAN ALSO BE FUNCTIONS"

THAT'S THE BEST WAY TO DESCRIBE A JAVASCRIPT OBJECT.

(THIS IS A VERY DIFFERENT WAY OF THINKING ABOUT OBJECTS THAN IN C++ OR JAVA OR PYTHON)

RECAP

"AN OBJECT IS A SET OF KEY-VALUE PAIRS, WHERE THE VALUES CAN ALSO BE FUNCTIONS"

EACH KEY-VALUE PAIR IS CALLED A PROPERTY OF THE OBJECT

RECAP

EACH KEY-VALUE PAIR IS CALLED A **PROPERTY** OF THE OBJECT

REMEMBER THE TERM '**PROPERTY**', ITS AN IMPORTANT TERM!

RECAP

EXAMPLE 17: ADDING PROPERTIES TO OBJECTS DYNAMICALLY

IN JAVASCRIPT, YOU CAN ADD (OR REMOVE) PROPERTIES ON-THE-FLY

EXAMPLE 17: ADDING PROPERTIES TO OBJECTS DYNAMICALLY

CREATE A CONSTRUCTOR FOR AN OBJECT WITH 3 PROPERTIES

```
function Rectangle(length,breadth,color) {
  this.length = length;
  this.breadth = breadth;
  this.color = color;
}
```

CREATE A CONSTRUCTOR FOR AN OBJECT WITH 3 PROPERTIES

```
function Rectangle(length,breadth,color
  this.length = length;
  this.breadth = breadth;
  this.color = color;
}
```

THE PROPERTIES ARE CALLED LENGTH, BREADTH AND COLOR.

CREATE A CONSTRUCTOR FOR AN OBJECT WITH 3 PROPERTIES

```
function Rectangle(length,breadth,colo
    this.length = length;
    this.breadth = breadth;
    this.color = color;
}
```

THE PROPERTIES ARE CALLED LENGTH, BREADTH AND

NOW CREATE AN OBJECT FROM THIS CONSTRUCTOR

```
var rectangle = new Rectangle(3.3,
2.5,"Blue");
```

CREATE A

```
function Rectangle(length,breadth,color) {
  this.length = length;
  this.breadth = breadth;
  this.color = color;
}
```

THE PROPERTIES ARE CALLED LENGTH, BREADTH AND

NOW CREATE AN OBJECT FROM THIS

```
var rectangle = new Rectangle(3.3,
2.5,"Blue");
```

TRY TO ACCESS A NON-EXISTENT PROPERTY CALLED `OutlineColor`

```
 console.log("our rectangle has outlinecolor = " +
rectangle.OutlineColor);
```

TRY TO ACCESS A NON-EXISTENT PROPERTY CALLED `OutlineColor`

```
console.log("our rectangle has outlinecolor = " +
rectangle.OutlineColor);
```

REMEMBER THAT ACCESSING ANY NON-EXISTENT VALUE IN JAVASCRIPT RETURNS THE SPECIAL VALUE `undefined`

```
our rectangle has outlinecolor = undefined
```

NO WORRIES, JUST ADD THIS PROPERTY ON THE FLY!

SYNTAX #1

ADD THIS PROPERTY ON THE FLY!

```
rectangle["OutlineColor"] = "Black";
```

PAY ATTENTION TO THIS SYNTAX!

SPECIFY THE NAME OF THE OBJECT

SYNTAX #1

ADD THIS PROPERTY ON THE FLY!

```
rectangle["OutlineColor"] =
"Black";
```

SPECIFY THE NAME OF THE PROPERTY IN SQUARE BRACKETS

SYNTAX #1

ADD THIS PROPERTY ON THE FLY!

```
rectangle["OutlineColor"] =
"Black";
```

SPECIFY THE NAME OF THE PROPERTY IN SQUARE BRACKETS

SYNTAX #1

ADD THIS PROPERTY ON THE FLY!

```
rectangle["OutlineColor"] =
"Black";
```

SPECIFY THE VALUE OF THE PROPERTY

AND THAT'S IT, WE CAN USE THE PROPERTY NOW!

THAT WAS

SYNTAX #1

THERE IS ALSO

SYNTAX #2

SYNTAX #2

```
rectangle.OutlineColor =
"Black";
```

THE 2 SYNTAX FORMS ARE EQUIVALENT

, BUT SYNTAX #1 ALLOWS YOU TO SPECIFY THE NAME OF THE PROPERTY IN A VARIABLE

```
var propertyName =
“OutlineColor”;
rectangle[propertyName] =
"Black";
```

SYNTAX #2

```
rectangle.OutlineColor =
"Black";
```

THE 2 SYNTAX FORMS ARE EQUIVALENT

AND THAT'S IT, WE CAN USE THE PROPERTY NOW!

AND THAT'S IT, WE CAN USE THE PROPERTY NOW!

```
 console.log("our rectangle has outlinecolor = " +
rectangle.OutlineColor);
  rectangle["OutlineColor"] = "Black";
  console.log("Trying again");
  console.log("our rectangle has area = " + rectangle.Outlin
```

```
our rectangle has outlinecolor = undefined
Trying again
our rectangle has area = Black
```

AND THAT'S IT, WE CAN USE THE PROPERTY NOW!

```
  console.log("our rectangle has outlinecolor = " +
rectangle.OutlineColor);
  rectangle["OutlineColor"] = "Black";
  console.log("Trying again");
  console.log("our rectangle has area = " + rectangle.Outlin
```

```
our rectangle has outlinecolor = undefined
Trying again
our rectangle has area = Black
```

AND THAT'S IT, WE CAN USE THE PROPERTY NOW!

```
  console.log("our rectangle has outlinecolor = " +
rectangle.OutlineColor);
  rectangle["OutlineColor"] = "Black";
  console.log("Trying again");
  console.log("our rectangle has area = " + rectangle.Outlin
```

```
our rectangle has outlinecolor = undefined
Trying again
our rectangle has area = Black
```