香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

MORE ON VARIABLES

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll appreciate the concept of local variables
- You'll appreciate the concept of global variables

LOCAL VARIABLES

- Variables declared within a function
  can only be accessed within the function
- They are *local* to the function,
  and so are called local variables

```
<!doctype html>
<html><body>
  <script>
    function show_money() {
      var money = 2;
      alert("In the function, the value is: "+ money);
    }
    money = 99;
    alert("In the main part, the value is: "+ money);
    show_money();
    alert("In the main part, the value is: "+ money);
  </script>
</body></html>
```

Click here to open the example

GLOBAL VARIABLES

- The opposite of a local variable is a *global variable*
- Global variables are created in the main part
- They can work inside or outside functions

```
<!doctype html>
<html><body>
  <script>
    function show_money() {
      alert("In the function, the value is: "+ money);
    }
    var money = 99;
    alert("In the main part, the value is: "+ money);
    show_money();
    alert("In the main part, the value is: "+ money);
  </script>
</body></html>
```

Click here to open the example

LOCAL AND GLOBAL VARIABLES SHARING THE SAME NAME

- JavaScript will give priority to the local variable inside the function

CREATING GLOBAL VARIABLES INSIDE FUNCTIONS

- If you assign a value to a variable that has not been *declared*, it will automatically become a global variable

```
<!doctype html>
<html><body>
  <script>
    function show_money() {
      money = 2;
      alert("In the function, the value is: "+ money);
    }
    show_money();
    alert("In the main part, the value is: "+ money);
  </script>
</body></html>
```

money被自动声明为global variable

Click here to open the example

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

LOGICAL OPERATORS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

- You'll appreciate Boolean values
- You'll know more about logical operators and how to use them

WE WILL LOOK AT

Boolean values	`true`
	`false`
Logical operators	`&&`
	`||`
	`!`

BOOLEAN

- A *Boolean* value is either `true` or `false`
- A variable which has a Boolean value is called a Boolean variable

LOGICAL OPERATORS

- Logical operators work with Boolean values
- JavaScript has these logical operators:
  - Logical And - the && operator
  - Logical Or - the || operator
  - Logical Not - the ! operator

AND - &&

- `&&` - the result is `true` if both inputs are `true`, otherwise the result is `false`

AND - &&

a	b	a && b
false	false	false
false	true	false
true	false	false
true	true	true

```
<html><body><script>
  var you_are_rich = false;
  var you_have_partner = true;
  var you_have_flat = true;
  var life_is_fantastic = you_are_rich
   && you_have_partner && you_have_flat;
  alert("life is fantastic is " +
              life_is_fantastic);
  you_are_rich = true;
  life_is_fantastic = you_are_rich
   && you_have_partner && you_have_flat;
  alert("life is fantastic is now " +
                 life_is_fantastic);
</script></body></html>
```

SHORT-CIRCUIT IN AND

- JavaScript is clever
- When it evaluates an And it checks the first input
- If the value is `false` it knows the result must be `false`
- So it doesn't bother checking the next input

```
<!doctype html>
<html>
  <body><script>
    function first_function() {
      alert("first_function() is running!");
      return true;
    }
    function second_function() {
      alert("second_function() is running!");
      return false;
    }
    var test_function =
       first_function() && second_function();
  </script></body>
</html>
```

AFTER SWAPPING THE FUNCTIONS

```
<html><body><script>
    function first_function() {
      alert("first_function() is running!");
      return true;
    }
    function second_function() {
      alert("second_function() is running!");
      return false;
    }
    var test_function_swapped =
       second_function() && first_function();
</script></body></html>
```

OR - ||

- || - the result is `false` if both inputs are `false`, otherwise the result is `true`

OR - ||

a	b	a || b
false	false	false
false	true	true
true	false	true
true	true	true

```
<!doctype html>
<html>
  <body><script>
    var you_are_rich = false;
    var you_have_partner = true;
    var you_have_flat = false;
    var life_is_good = you_are_rich
      || you_have_partner || you_have_flat;
    alert("life is good is " + life_is_good);
    you_have_partner = false;
    life_is_good = you_are_rich
      || you_have_partner || you_have_flat;
    alert("life is good is now " + life_is_good);
  </script></body>
</html>
```

SHORT-CIRCUIT IN OR

- If JavaScript is evaluating Or and the first input is true, it knows the result must be `true`
- So it doesn't bother checking the second input

```
<!doctype html>
<html>
  <body><script>
    function first_function() {
      alert("first_function() is running!");
      return true;
    }
    function second_function() {
      alert("second_function() is running!");
      return false;
    }
    var test_function =
       first_function() || second_function();
  </script></body>
</html>
```

NOT - !

- ! - the result is the opposite of the input

NOT - !

a	!a
false	true
true	false

```
<!doctype html>
<html>
  <head>
    <title>Not Operator Example</title>
  </head>
  <body>
    <script>
      var you_are_male = true;
      var you_are_female = !you_are_male;
      alert("you_are_male is " + you_are_male);
      alert("you_are_female is " + you_are_female);
    </script>
  </body>
</html>
```

香港科技大學
THE HONG KONG
UNIVERSITY OF SCIENCE
AND TECHNOLOGY

ARRAYS

PROF. DAVID ROSSITER

THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

AFTER THIS PRESENTATION

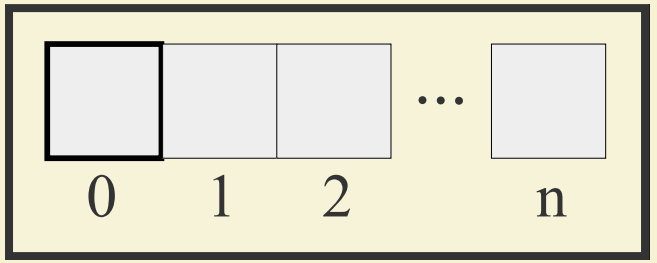
- You'll understand and use the array data structure
- You'll be able use some common array functions

ARRAY FUNCTIONS

`[]`	`push()`	`concat()`
`length`	`shift()`	
`join()`	`pop()`	
	`unshift()`	

ARRAY

- An array is a linear continuous storage



- You can think array as a group of boxes
- Each box has a unique identity, which is called an *index*
- The *index* of the first box is **0**

CREATING AN ARRAY

两种声明方法

- Here is how you create a new array with 3 boxes:

```
var pets = ["Dog", "Cat", "Rabbit"];
```

- You can create a new array with 10 boxes without any element inside the boxes like this:

```
var pets = new Array(10);
```

- You can put anything in an array
- Any element can be any data type

JOIN()

用于打印

- Use *array*.`join(`*separator*`)` to convert *array* into string:

```
var pets = ["Dog", "Cat", "Rabbit"];
alert(pets.join(" and "));
// This shows "Dog and Cat and Rabbit"
```

- *separator* is by default " , "

```
var pets = ["Dog", "Cat", "Rabbit"];
alert(pets.join());
// This shows "Dog,Cat,Rabbit"
```

GETTING SOMETHING

- With this array:

```
var pets = ["Dog", "Cat", "Rabbit"];
```

- You can retrieve something like this:

```
alert(pets[2]); // This shows "Rabbit"
```

CHANGING SOMETHING

- With this array:

```
var pets = ["Dog", "Cat", "Hamster"];
```

- You can change something stored in the array like this:

```
pets[2] = "Rabbit";
// Now pets is ["Dog", "Cat", "Rabbit"]
```

ARRAY SIZE

- You can know the size of an array (i.e. how many boxes it has) using *array*`.length`:

```
var pets = ["Dog", "Cat", "Rabbit"];
alert(pets.length); // This shows 3
```

ADDING TO THE END

- Add a new element to the end of an array with *array*.push( ):

```
var pets = ["Dog", "Cat", "Rabbit"];
pets.push("Hamster");
// Now pets is
// ["Dog", "Cat", "Rabbit", "Hamster"]
```

- The *index* are automatically updated

end: array.push("text") front: array.unshift("text")

array.pop() array.shift()

ADDING TO THE FRONT

- Add a new element to the front with *array*.`unshift()`:

```
var pets = ["Dog", "Cat", "Rabbit"];
pets.unshift("Hamster");
// Now pets is
// ["Hamster", "Dog", "Cat", "Rabbit"]
```

- The *index* are automatically updated

REMOVING FROM THE BACK

- To remove an element from the end, use *array*.`pop()`:

```
var pets = ["Dog", "Cat", "Rabbit"];
var result = pets.pop();
// Now pets is [Dog", "Cat"]
```

- `pop()` returns the removed element, so *result* is `"Rabbit"`

REMOVING FROM THE FRONT

- *array*.`shift()` removes an element from the front:

```
var pets = ["Dog", "Cat", "Rabbit"];
var result = pets.shift();
// Now pets is ["Cat", "Rabbit"]
```

- `shift()` returns the removed element, so *result* is `"Dog"`
- The *index* are automatically updated

COMBINING TWO ARRAYS

- Use *array1*.concat(*array2*) to combine two arrays into one:

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
var primes = [2, 3, 5, 7, 11];
var result = pets.concat(primes);
// result is ["Dog", "Cat", "Rabbit", "Hamster",
//             2, 3, 5, 7, 11]
```