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

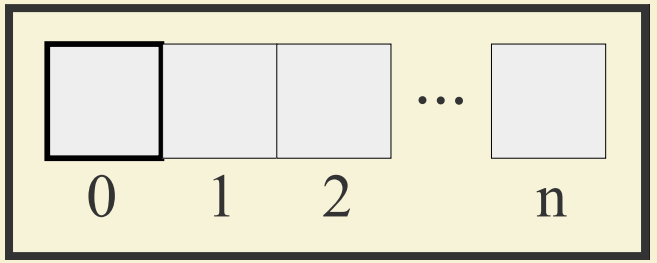
- You'll understand and use the array data structure
- You'll be able use some common array functions

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

- 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()

- 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

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]
```