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

ARRAY FUNCTIONS

PROF. DAVID ROSSITER

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

AFTER THIS PRESENTATION

- You'll learn some more advanced array functions

WE'LL LOOK AT

`forEach()`
`map()`

FOREACH()

- You can go through every element using loop (`for` / `while`)

```
var pets = ["Dog", "Cat", "Hamster"];
for(var i = 0; i < pets.length; i++) {
    alert(pets[i]);
}
```

- You can also use *array*.`forEach(`*function*`)`:

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

MORE ON FOREACH()

- You can think of `forEach()` in this way:

```
function forEach(theArray, fn) {
  for(var i = 0; i < theArray.length; i++) {
    fn(theArray[i], i, theArray);
  }
}
```

- So, your function should look like this,
  if you need all of the 3 things:

用于函数中

```
function yourFunction(element, index, array) {}
```

```
<!doctype html>
<html>
<body>
   <script>
      var numbers = [1, 2, 3, 4, 5];
      numbers.forEach( function(elem, idx, arr) {
        arr[idx] = elem * elem;
      });
      alert(numbers); // This shows [1,4,9,16,25];
   </script>
</body>
</html>
```

elem: 表示number中的元素

idx: 表示数组arr的下标

最后的结果会replace number数组

MAP()

- `map(`*function*`)` stores the result of each execution of *function* into an array it returns.
  You can think of `map()` in this way:

```
function map(theArray, fn) {
  var results = [];
  for(var i = 0; i < theArray.length; i++) {
    results.push(fn(theArray[i], i, theArray));
  }
  return results;
}
```

1.run the function one by one
2. restore the answer on the results

```
<!doctype html>
<html>
<body>
   <script>
      var square = function(el) { return el * el; }
      var numbers = [1, 2, 3, 4, 5];
      var results = numbers.map(square);
      alert(results); // This shows [1,4,9,16,25];
   </script>
</body>
</html>
```

指向函数的变量

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

MORE ON 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 learn some advanced array functions

ADVANCED ARRAY FUNCTIONS

`sort()`	`indexOf()`	`slice()`
`reverse()`	`lastIndexOf()`	`splice()`

SORTING

- *array*`.sort()` sorts the elements in *array*:

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

REVERSE

- *array*.`reverse()` reverses *array*
- The first element becomes the last;
  The last element becomes the first

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

DESCENDING ORDER

可以组合操作

- By combining `sort()` and `reverse()`, you can sort things in descending order:

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

FINDING AN ELEMENT

- Use *array*.`indexOf(`*target*`)` to find the index of the first occurence of *target* in *array*:

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
alert(pets.indexOf("Rabbit")); // This shows 2
```

- If *target* is not in *array*, `indexOf()` will return **-1**

MORE ON FINDING AN ELEMENT

- Pass a second value to `indexOf()` to control where to start the search

```
array.indexOf(target, startPosition)
```

```
<html><body><script>
  var pets = ["Dog", "Cats", "Rabbit", "Hamster",
          "Rabbit", "Rabbit", "Dog", "Cat",
          "Hamster", "Hamster", "Rabbit"];
  var rabbitPositions = [], startSearchAt = 0;
  do {
    foundAt = pets.indexOf("Rabbit", startSearchAt);
    if(foundAt != -1) {
      rabbitPositions.push(foundAt);
      startSearchAt = foundAt + 1;
    }
  } while(foundAt != -1);
  alert(rabbitPositions); // This shows [2, 4, 5, 10]
</script></body></html>
```

FINDING ELEMENT BACKWARDS

- Use *array*.`lastIndexOf(`*target*`)` to find *target* in *array*, starting from the last element in *array*:

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

SLICE()

- Extract part of an array by *array*.`slice`(*startPosition*):

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
var result = pets.slice(1);
// result is ["Cat", "Rabbit", "Hamster"]
```

- You can also set where to stop, by *array*.`slice`(*startPosition*, *endPosition*):

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
var result = pets.slice(1, 3);
// result is ["Cat", "Rabbit"]
```

REMOVE SOMETHING ANYWHERE IN AN ARRAY

- `splice()` is used when you want to remove element(s) anywhere from an array
- To remove element(s) anywhere from an array, use *array*.`splice`(*position*, *quantity*)

```
var pets = ["Dog", "Cat", "Rabbit", "Hamster"];
var result = pets.splice(1, 1);
// Now pets is ["Dog", "Rabbit", "Hamster"]
// and result is ["Cat"]
```

- `splice()` returns the removed element(s)

ADD SOMETHING ANYWHERE IN AN ARRAY

- `splice()` can also be used when you want to add element(s) anywhere to an array
- To add an element anywhere to an array, use *array*.`splice(`*position*, 0, *element*`)`

```
var pets = ["Dog", "Cat", "Hamster"];
var result = pets.splice(2, 0, "Rabbit");
// Now pets is ["Dog", "Cat", "Rabbit", "Hamster"]
// and result is []
```

- Because nothing is removed from *pets*, *result* is [ ]

REPLACE SOMETHING ANYWHERE IN AN ARRAY

- To replace element(s) anywhere in an array, use *array*`.splice(`*position*, *quantity*, *element(s)*`)`

```
var pets = ["Dog", "Cat", "Hamster"];
var result = pets.splice(1, 1, "Rabbit", "Fish");
// Now pets is ["Dog", "Rabbit", "Fish", "Hamster"]
// and result is ["Cat"]
```