PDP UNIVERSITY

Website development

Lecture 6

Parameters rule

There are 2 rules when using function parameters :

1. The type of parameters is not specified
2. The number of parameters is not checked

```
function myFunction(x, y) {  // function accepts 2 parameters
    return x + y;
}
myFunction(15); // only 1 parameter passed. y =undefined
```

If the specified parameter is not included when the function is called, its value will be **undefined**

Parameters rule

You can check that the specified parameter is not included when the function is called in the following way:

```
function myFunction(x, y) {
  if (y === undefined) {
    console.log('y is compulsory');
  }
}
```

Default parameter

We may define default value for a parameter:

```
function myFunction(x, y=2) {
  // if value for y is not passed, y will be equal to 2
}
```

Parameter rule

Javascript can accept more parameters than specified:

```
function sum(x, y) {  // function accepts 2 parameters
    return x + y;
}
sum(15, 5, 10); // but 3 arguments passed
```

We can refer to all the arguments passed to the function using a keyword **`arguments`**. It returns us an array. In the example above, arguments[2] returns 10.

Function expression

A **function expression** is very similar to and has almost the same syntax as a function declaration (lecture 2). There are 3 main differences:

1. The function name can be omitted in function expressions to create anonymous functions.
2. Function expression does not have hoisting
3. Function expressions can be written inside other expressions, e.g. variable declaration or passing arguments.

Function expression

Example of function expression:

```
const sum = function(a, b){
        let result = a + b;
        return result;
}
```

Arrow function

Function expressions can be written in a shorter way using arrow function

syntax:

Normal syntax:

```
const sum = function(a, b){
        let result = a + b;
        return result;
}
```

Arrow syntax:

```
const sum = (a, b) => {
        let result = a + b;
        return result;
}
```

Arrow function

If an arrow function consists of a single line you can omit curly braces:

Full syntax:

```
const greeting = () => {
   console.log('Hello world');
}
```

Shortened syntax:

```
const greeting = () => console.log('Hello world');
```

Arrow function

If an arrow function consists of a single return statement, keyword **`return`** can be omitted, too:

Full syntax:

```
const sum = (a, b) => {
   return a + b;
}
```

Shortened syntax:

```
const sum = (a, b) => a + b;
```

Callback function

We can pass functions as parameters:

Full syntax:

```
function ask(question, yes, no) {
  if (confirm(question))
      yes();
  }
  else {
     no();
  }
}
function showOk() {
  alert( "You agreed");
}
function showCancel() {
  alert( "You did not agree." );
}
```

```
ask("Do you agree?", showOk, showCancel);
```

Callback function expression

```
function ask(question, yes, no) {
  if (confirm(question)){
     yes()
  }
  else {
    no();
  }
}

ask(
  "Do you agree?",
  function() { alert("You agreed},
  function() { alert("You did not agree."); }
);
```

Thank you for your attention