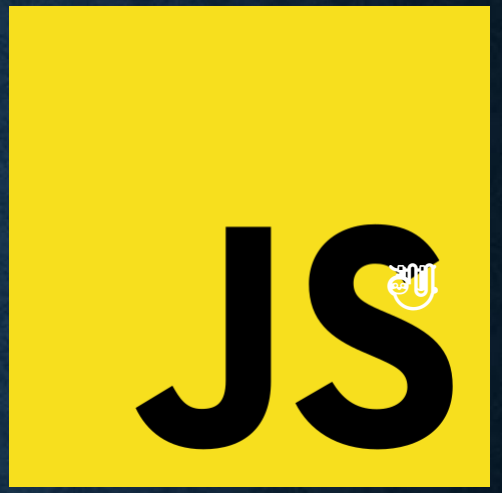

JAVASCRIPT

Conditions, loops, Exceptions, Functions, And Timers

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

JAVASCRIPT – IF STATEMENTS

- Conditional execution of code with an if statement

- Multiple lines

```
if(condition)
{
        // Execute if condition is true
}
```

- Single line

```
if(condition) // Execute if condition is true
```

JAVASCRIPT – IF STATEMENTS

- Conditional execution of code with an if-else statement

```
if(condition)
{
        // Execute if condition is true
}
else
{
        // Execute if condition is false
}
```

JAVASCRIPT – IF STATEMENTS

- Conditional execution of code with an if-else-if statement

```
if(condition1)
{
        // Execute if condition1 is true
}
else if(condition2)
{
        // Execute if condition2 is true
}
```

JAVASCRIPT – IF STATEMENTS

- Example

```
var x = 0.75;
if(x > 0.50 && x < 1.00)
{
        console.log("Yes, it's true");
}
else
{
        console.log("Oh no, it's false");
}
```

JAVASCRIPT – SWITCH STATEMENTS

- Conditional execution of code with a switch statement

```
switch(selector)
{
        case label: statement; break;
        case label:
        {
                // Compound statement
        } break;
}
```

JAVASCRIPT – SWITCH STATEMENTS

- Switch statement with a default

```
switch(selector)
{
        case case1: statement1;
        break;
        case case2: statement2;
        break;
        default: statement3;
}
```

JAVASCRIPT – SWITCH STATEMENTS

- Example

```
var x = 2;
switch(x)
{
        case 1: log("The answer is one");
        break;
        case 2: log("The answer is two");
        break;
}
```

JAVASCRIPT – FOR LOOPS

- Loop a few iterations

```
for(initial value; condition; step)
{
        // Loop body
}
```

JAVASCRIPT – FOR LOOPS

- Example

```
for(var i = 0; i < 10; i++)
{
        // Loop body
}
```

JAVASCRIPT – WHILE LOOPS

- Loop a few iterations

```
while(condition)
{
        // Loop body
}
```

JAVASCRIPT – WHILE LOOPS

- Example

```
var i = 10;
while(i > 0)
{
        // Loop body
        i--;
}
```

JAVASCRIPT – DO-WHILE LOOPS

- Loop a few iterations

```
do
{
        // Loop body
}
while(condition);
```

JAVASCRIPT – DO-WHILE LOOPS

- Example

```
var i = 10;
do
{
        // Loop body
        i--;
}
while(i > 0);
```

JAVASCRIPT – EXCEPTIONS

- Throw your own exception/error with the throw keyword

```
throw exception;
```

- The exception can be a string, number, boolean, array, or object

```
throw "My own exception";
```

JAVASCRIPT – EXCEPTIONS

- Catch exceptions

```
try
{
        // Statements that throw ...
}
catch(exception)
{
        // Do something with exception
}
```

JAVASCRIPT – EXCEPTIONS

- Example

```
var num = 0.5;
try
{
        if(num < 0) throw "too small";
        else if(num > 1) throw "too big";
}
catch(exception)
{
        alert("Error: " + exception);
}
```

JAVASCRIPT – EXCEPTIONS

- Example

```
try
{
        var num = 1 / 0;
        if(num == Infinity) throw "division by 0";
}
catch(exception)
{
        alert("Error: " + exception);
}
```

JAVASCRIPT – FUNCTIONS

- Define a custom function

```
function identifier(parameters)
{
        // Function body
}
```

JAVASCRIPT – FUNCTIONS

- Example

```
function sum(num1, num2)
{
        var result = num1 + num2;
        return result;
}

var answer = sum(10, 2);
```

JAVASCRIPT – FUNCTIONS

- Functions with the variable parameter number

```
function sum()
{
        var result = 0;
        for(var i = 0; i < arguments.length; i++)
                result += arguments[i];
        return result;
}

var result = sum();
result = sum(5, 6);
result = sum(5, 6, 9, 10);
```

JAVASCRIPT – FUNCTIONS

- Passing functions as parameters

```
function executor(func)
{
        func();
}

function executee()
{
        console.log("Executing ...");
}

executor(executee);
```

JAVASCRIPT – FUNCTIONS

- Hold functions in variables

```
function executor()
{
}

var func = executor;
```

JAVASCRIPT – TIMERS

- Timers are used to execute some code after a certain period of time
- Browsers typically assign a single thread per website or opened tab
- Timers can be used as threads
  - Start a timer
  - Continue executing other code
  - The timer will fire at some time in the future and execute in the “background”

JAVASCRIPT – TIMERS

- Timeouts execute some code **once** after a certain period of time

```
setTimeout(function, milliseconds, parameter1, parameter2, ...);
```

JAVASCRIPT – TIMERS

- Timeouts with inline function

```
setTimeout(function(){ alert("Bitcoin is going up"); }, 5000);
```

- Timeouts with custom named functions

```
function bitcoin(direction)
{
        alert("Bitcoin is going " + direction);
}

setTimeout(bitcoin, 2000, "down");
```

JAVASCRIPT – TIMERS

- Intervals execute some code **repeatedly** every few milliseconds

```
setInterval(function, milliseconds, parameter1, parameter2, ...);
```

JAVASCRIPT – TIMERS

- Intervals work the same as timeouts, allowing both inline and named functions

```
setInterval(function(){ alert("Bitcoin is going up"); }, 5000);
```

- Interval timers can be stopped at any moment

```
var interval = setInterval(function(){ alert("Ethereum …"); }, 5000);
clearInterval(interval); // Stops the timer
```

JAVASCRIPT – TIMERS (SUMMARY)

- setTimeout
- setInterval
- clearTimeout
- clearInterval

JAVASCRIPT – TIMERS

http://www.youtube.com/watch?v=0VVJSvlUgtg