JavaScript Control Structures

SENG 4640
Software Engineering for Web Apps
Winter 2023

Sina Keshvadi
Thompson Rivers University

Conditional Statements

```
var a = . . .

var b = . . .
var max; // undefined

if (a > b) {

   max = a;
}
else {
   max = b;
}

console.log(max);
```

Conditional Statements

```
var a = . . .

var b = . . .
var max; // undefined

if (a > b) {

   max = a;
}
else {
   max = b;
}

console.log(max);
```

Conditional Statements

```
var a = . . .

var b = . . .

var max; // undefined

if (a > b)
   { max =
   a;
}
else {
   max = b;
}

console.log(max);
```

Conditional Statements

```
var a = . . .

var b = . . .

var max; // undefined

if (a > b) {
    max = a;
}
else {
    max = b;
}

console.log(max);
```

Conditional Statements

```
var a = . . .

var b = . . .
var max; // undefined

if (a > b) {

   max = a;
}
else {
   max = b;
}

console.log(max);
```

Conditional Statements

```
var a = . . .

var b = . . .
var max; // undefined

if (a > b) {

   max = a;
}
else {
   max = b;
}

console.log(max);
```

Conditional Statements

```
var a = . . .

var b = . . .
var max; // undefined

if (a > b) {

   max = a;
}
else {
   max = b;
}

console.log(max);
```

Comparison and Logical Operators

Comparison Operators

Operator	Description
==	equal to
===	equal to and same type
!=	not equal to
!==	not equal to or different type
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Comparison and Logical Operators

Comparison Operators

Operator	Description
==	equal to
===	equal to and same type
!=	not equal to
!==	not equal to or different type
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

Logical Operators

Operator	Description
||	logical OR
&&	logical AND
!	logical NOT

Double-equals vs. Triple-equals

- Use double-equals (==) when you only want to compare **values**

```
1 == ‘1’ // true
```

Double-equals vs. Triple-equals

- Use double-equals (==) when you only want to compare **values**

```
1 == ‘1’ // true
```

Double-equals vs. Triple-equals

- Use double-equals (==) when you only want to compare **values**
- Use triple-equals (===) when you want to compare values **and** type

```
1 == ‘1’ // true

1 === ‘1’ // false! different types
```

Comparing Truthy/Falsy Values

- Recall that any value can be used as a boolean
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

```
var x; // undefined
if (x) { . . . } // false! undefined is
                    falsy
x = 0;
if (x) { . . . } // false! 0 is falsy
x = 39;
if (x) { . . . } // true! 39 is truthy


var y = null;
var z; // undefined
if (y == z) { . . . } // true! falsy equals falsy if

(y === z) { . . . } // false! different types
```

Comparing Truthy/Falsy Values

- Recall that any value can be used as a boolean
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

```
var x; // undefined
if (x) { . . . } // false! undefined is
                    falsy
x = 0;
if (x) { . . . } // false! 0 is falsy
x = 39;
if (x) { . . . } // true! 39 is truthy


var y = null;
var z; // undefined
if (y == z) { . . . } // true! falsy equals falsy if

(y === z) { . . . } // false! different types
```

Comparing Truthy/Falsy Values

- Recall that any value can be used as a boolean
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

```
var x; // undefined
if (x) { . . . } // false! undefined is
                    falsy

x = 0;
if (x) { . . . } // false! 0 is falsy
x = 39;
if (x) { . . . } // true! 39 is truthy


var y = null;
var z; // undefined
if (y == z) { . . . } // true! falsy equals falsy if

(y === z) { . . . } // false! different types
```

Comparing Truthy/Falsy Values

- Recall that any value can be used as a boolean
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

```
var x; // undefined
if (x) { . . . } // false! undefined is
                    falsy
x = 0;
if (x) { . . . } // false! 0 is falsy
x = 39;
if (x) { . . . } // true! 39 is truthy


var y = null;
var z; // undefined
if (y == z) { . . . } // true! falsy equals falsy if

(y === z) { . . . } // false! different types
```

Comparing Truthy/Falsy Values

- Recall that any value can be used as a boolean
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

```
var x; // undefined
if (x) { . . . } // false! undefined is
                    falsy
x = 0;
if (x) { . . . } // false! 0 is falsy
x = 39;
if (x) { . . . } // true! 39 is truthy


var y = null;
var z; // undefined
if (y == z) { . . . } // true! falsy equals falsy if

(y === z) { . . . } // false! different types
```

Comparing Truthy/Falsy Values

- Recall that any value can be used as a boolean
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

```
var x; // undefined
if (x) { . . . } // false! undefined is
                    falsy
x = 0;
if (x) { . . . } // false! 0 is falsy
x = 39;
if (x) { . . . } // true! 39 is truthy


var y = null;
var z; // undefined
if (y == z) { . . . } // true! falsy equals falsy if

(y === z) { . . . } // false! different types
```

Comparing Truthy/Falsy Values

- Recall that any value can be used as a boolean
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

```
var x; // undefined
if (x) { . . . } // false! undefined is
                    falsy

x = 0;
if (x) { . . . } // false! 0 is falsy
x = 39;
if (x) { . . . } // true! 39 is truthy


var y = null;
var z; // undefined
if (y == z) { . . . } // true! falsy equals falsy if

(y === z) { . . . } // false! different types
```

Comparing Truthy/Falsy Values

- Recall that any value can be used as a boolean
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

```
var x; // undefined
if (x) { . . . } // false! undefined is
                    falsy
x = 0;
if (x) { . . . } // false! 0 is falsy
x = 39;
if (x) { . . . } // true! 39 is truthy


var y = null;
var z; // undefined

if (y == z) { . . . } // true! falsy equals falsy

if (y === z) { . . . } // false! different types
```

Comparing Truthy/Falsy Values

- Recall that any value can be used as a boolean
  - "Falsy" values: `null, undefined, 0, NaN, ''`
  - "Truthy" values: `'cow', 'false', 5,` etc...

```
var x; // undefined
if (x) { . . . } // false! undefined is
                    falsy

x = 0;
if (x) { . . . } // false! 0 is falsy
x = 39;
if (x) { . . . } // true! 39 is truthy


var y = null;
var z; // undefined

if (y == z) { . . . } // true! falsy equals falsy

if (y === z) { . . . } // false! different types
```

Comparing Numbers and Strings

- When comparing a string to a number, JavaScript will try to convert the string to a numeric form

Comparing Numbers and Strings

- When comparing a string to a number, JavaScript will try to convert the string to a numeric form

```
5 < ‘20’ // true
```

Comparing Numbers and Strings

- When comparing a string to a number, JavaScript will try to convert the string to a numeric form

```
5 < ‘20’ // true
‘5’ < 20 // true
```

Comparing Numbers and Strings

- When comparing a string to a number, JavaScript will try to convert the string to a numeric form

```
5 < '20' // true
'5' < 20 // true
```

- Non-numeric strings are converted to NaN

```
5 > 'alligator' // false
```

Comparing Numbers and Strings

- When comparing a string to a number, JavaScript will try to convert the string to a numeric form

```
5 < ‘20’ // true
‘5’ < 20 // true
```

- Non-numeric strings are converted to NaN

```
5 > ‘alligator’ // false
5 < ‘alligator’ // also false!
```

Comparing Numbers and Strings

- When comparing a string to a number, JavaScript will try to convert the string to a numeric form

```
5 < '20' // true
'5' < 20 // true
```

- Non-numeric strings are converted to NaN

```
5 > 'alligator' // false
5 < 'alligator' // also false!
```

- Non-numeric strings are compared alphabetically

```
'zebra' > 'giraffe' // true
```

Comparing Objects

- Objects are only considered equal if the variables are **aliases**, i.e. refer to the same object

```
var cooper = { age: 11 }
var flanders = { age: 11
}

if (cooper == flanders) { . . . } // false!



var myDog = cooper;

if (myDog == cooper) { . . . } // true!
```

Comparing Objects

- Objects are only considered equal if the variables are **aliases**, i.e. refer to the same object

```
var cooper = { age: 11 }
var flanders = { age: 11 }

if (cooper == flanders) { . . . } // false!



var myDog = cooper;

if (myDog == cooper) { . . . } // true!
```

Comparing Objects

- Objects are only considered equal if the variables are
aliases, i.e. refer to the same object

```
var cooper = { age: 11 }
var flanders = { age: 11 }

if (cooper == flanders) { . . . } // false!



var myDog = cooper;

if (myDog == cooper) { . . . } // true!
```

Comparing Objects

- Objects are only considered equal if the variables are **aliases**, i.e. refer to the same object

```
var cooper = { age: 11 }
var flanders = { age: 11
}

if (cooper == flanders) { . . . } // false!



var myDog = cooper;

if (myDog == cooper) { . . . } // true!
```

Comparing Objects

- Objects are only considered equal if the variables are **aliases**, i.e. refer to the same object

```
var cooper = { age: 11 }
var flanders = { age: 11
}

if (cooper == flanders) { . . . } // false!



var myDog = cooper;

if (myDog == cooper) { . . . } // true!
```

Comparing Objects

- Objects are only considered equal if the variables are **aliases**, i.e. refer to the same object

```
var cooper = { age: 11 }
var flanders = { age: 11
}

if (cooper == flanders) { . . . } // false!



var myDog = cooper;

if (myDog == cooper) { . . . } // true!
```

Loops

```
var n = ...
var factorial = 1;
```

Loops

```
var n = ...
var factorial = 1;
```

```
for (var i = 1; i <= n; i++)
    { factorial *= i;
}
```

Loops

```
var n = ...
var factorial = 1;
```

```
for (var i = 1; i <= n; i++)
    { factorial *= i;
}
```

```
var i = 1;
while (i <= n) {
    factorial *= i;
    i++;
}
```

Loops

```
var n = ...
var factorial = 1;
```

```
for (var i = 1; i <= n; i++)
    { factorial *= i;
}
```

```
var i = 1;
while (i <= n) {
    factorial *= i;
    i++;
}
```

```
var i = 1;
do {
    factorial *= i;
    i++;
}
while (i <= n);
```

Summary

- JavaScript supports conditional statements and loops
- Comparison operators can be used to compare by value and also by type