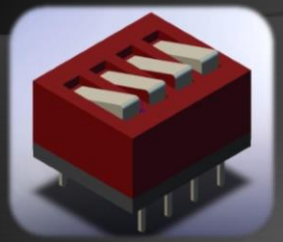
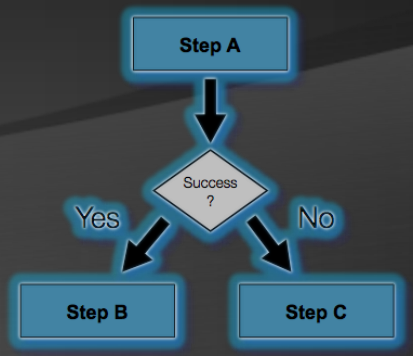
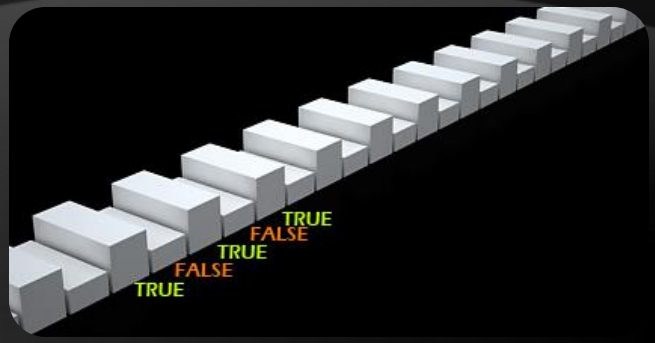
Telerik Academy







Conditional Statements

Implementing Control Logic in JavaScript

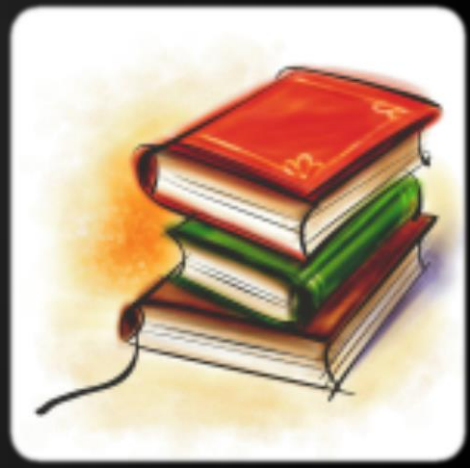
JavaScript Fundamentals

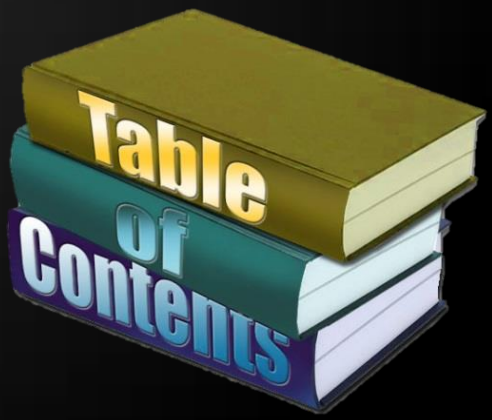
Telerik Software Academy

http://academy.telerik.com

Telerik Academy

Table of Contents





Telerik Academy

if and if-else

Implementing Conditional Logic

The if Statement

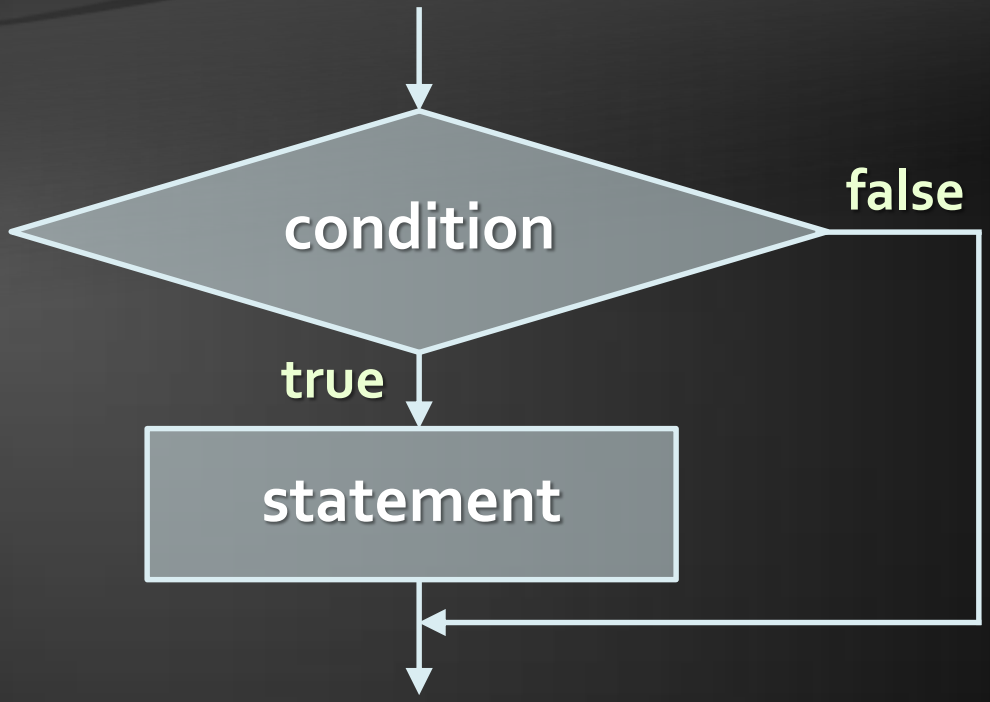
- The most simple conditional statement
- Enables you to test for a condition
- Branch to different parts of the code depending on the result
- The simplest form of an `if` statement:

```
if (condition) {
    statements;
}
```

Condition and Statement

- The condition can be:
  - Boolean variable
  - Boolean logical expression
  - Comparison expression
  - Integer, object, function… anything!
- The condition can be of any type
- The statement can be:
  - Single statement ending with a semicolon
  - Block enclosed in braces

How It Works?



- The condition is evaluated
  - If it is true-like, the statement is executed
  - If it is false-like, the statement is skipped

The if Statement – Example

- Examples with if statements
  - The expression evaluates for true-like or false-like values

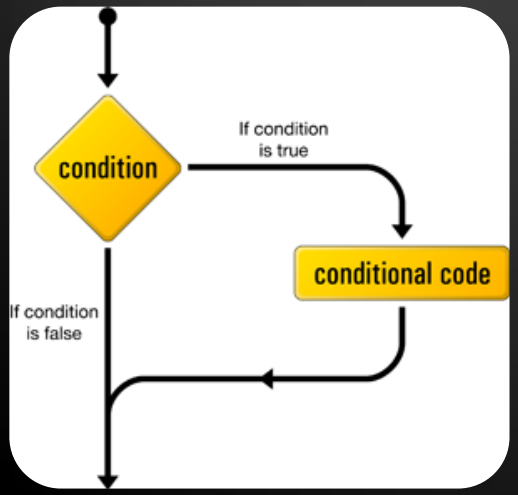
```
var bigger = 123;
var smaller = 24;
if (smaller > bigger) {
  bigger = smaller;
}
console.log('The greater number is: ' + bigger);
```

```
var str = '1c23';
if(!(+str)){ // if str is not a number, +str is NaN
  throw new Error('str is not a Number!');
}
```

Telerik Academy

The if Statement

Live Demo

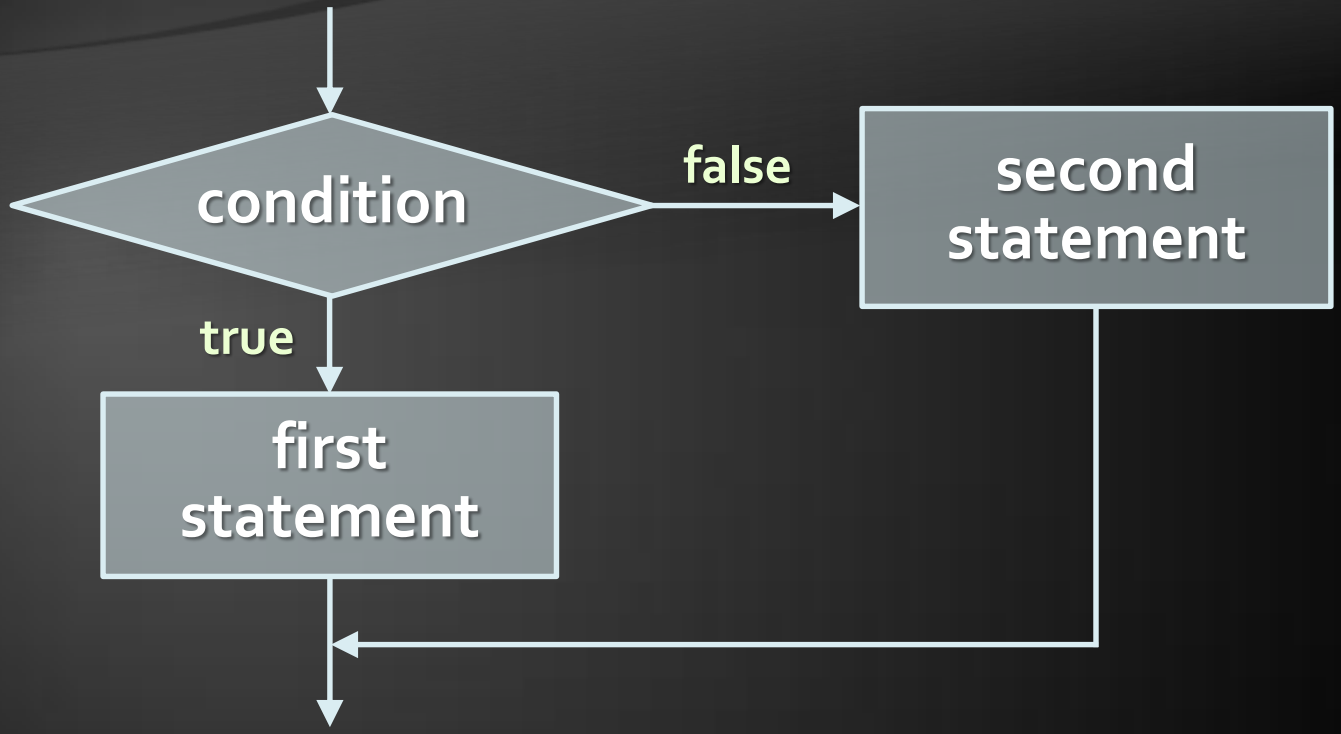


The if-else Statement

- More complex and useful conditional statement
- Executes one branch if the condition is true, and another if it is false
- The simplest form of an `if-else` statement:

```
if (expression) {
    statement1;
} else {
    statement2;
}
```

How It Works ?



- The condition is evaluated
  - If it is true-like, the first statement is executed
  - If it is false-like, the second statement is executed

Telerik Academy

if-else Statement – Example

- Checking a number if it is odd or even

```
var s = '123';
var number = +s;
if (number % 2) {
    console.log('This number is odd.');
} else {
    console.log('This number is even.');
}
```

The same as if(number % 2 === 1)

```
if (+str) {
  console.log('The string is a Number');
} else {
  console.log('The string is not a Number');
}
```

If 'str' is not a Number, the result will be NaN (FALSE-like)

Telerik Academy

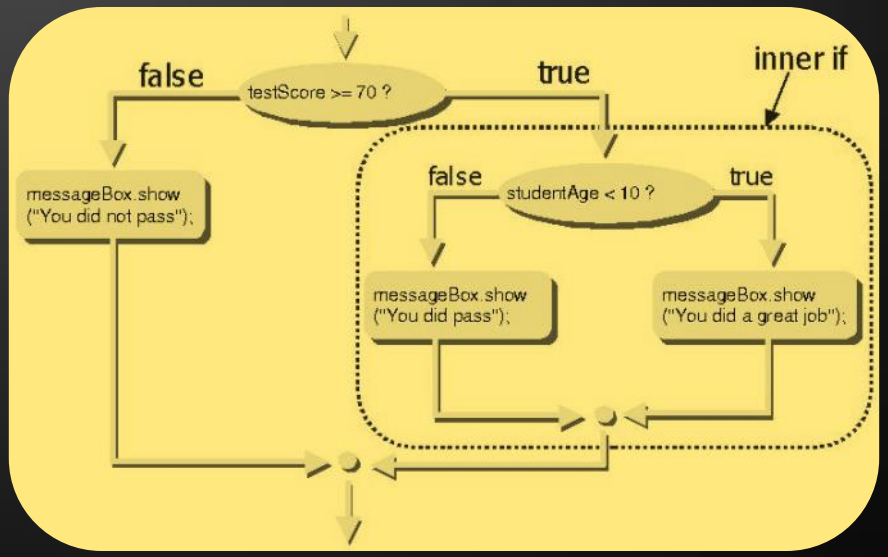
The if-else Statement

Live Demo

Telerik Academy

Nested if Statements

Creating More Complex Logic



Nested if Statements

- if and if-else statements can be nested, i.e. used inside another if or else statement
- Every else corresponds to its closest preceding if

```
if (expression) {
  if (expression) {
    statement;
  } else {
    statement;
  }
} else {
  statement;
}
```

Nested `if` – Good Practices

- Always use { … } blocks to avoid ambiguity
  - Even when a single statement follows
- Avoid using more than three levels of nested `if` statements
- Put the case you normally expect to process first, then write the unusual cases
- Arrange the code to make it more readable

Telerik Academy

Nested if Statements – Example

- Examples with nested if statements

```
if (first === second) {
  console.log('These two numbers are equal.');
} else {
  if (first > second) {
    console.log('The first number is bigger.');
  } else {
    console.log('The second is bigger.');
  }
}
```

```
var n = +str;
if (n) {
  if (n % 2) {
    console.log('The number is odd');
  } else {
    console.log('The number is even');
  }
} else { //n is NaN
  console.log('This is not a number!');
}
```

Telerik Academy

Nested if Statements

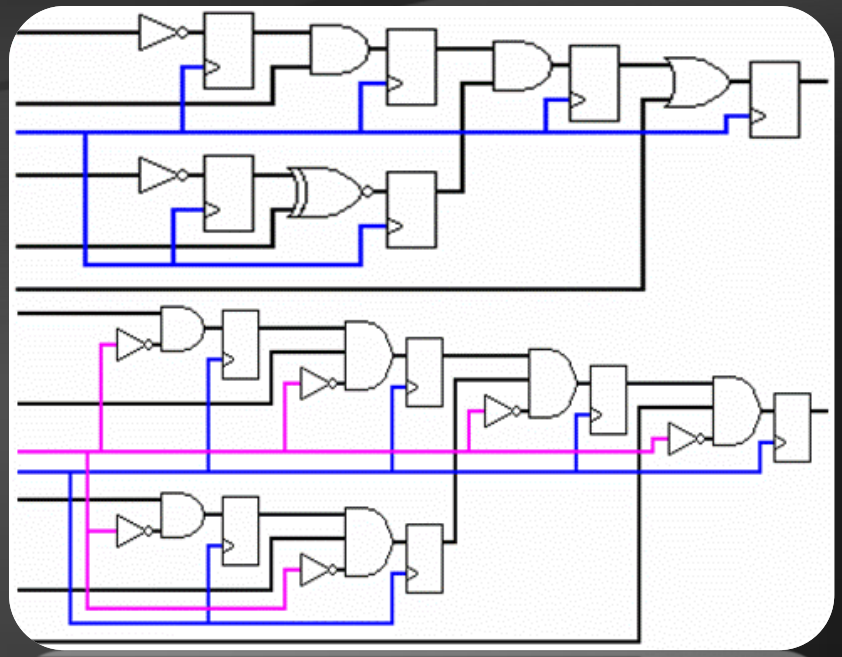
Live Demo

Multiple if-else-if-else-…

- Sometimes we need to use another `if` construction in the `else` block
  - Thus `else if` can be used:

```
var ch = 'X';
if (ch === 'A' || ch === 'a') {
  console.log('Vowel [ei]');
} else if (ch === 'E' || ch === 'e') {
  console.log('Vowel [i:]');
} else if  …
else …
```

Telerik Academy



Multiple if-else Statements

Live Demo

Telerik Academy

switch-case

Making Several Comparisons at Once

Telerik Academy

The switch-case Statement

- Selects for execution a statement from a list depending on the value of the switch expression

```
switch (day) {
    case 1: console.log('Monday'); break;
    case 2: console.log('Tuesday'); break;
    case 3: console.log('Wednesday'); break;
    case 4: console.log('Thursday'); break;
    case 5: console.log('Friday'); break;
    case 6: console.log('Saturday'); break;
    case 7: console.log('Sunday'); break;
    default: console.log('Error!'); break;
}
```

How switch-case Works?

1. The expression is evaluated
2. When one of the constants specified in a case label is equal to the expression
   - The statement that corresponds to that case is executed
3. If no case is equal to the expression
   - If there is default case, it is executed
   - Otherwise the control is transferred to the end point of the switch statement

Telerik Academy

The switch-case Statement

Live Demo

Telerik Academy

The Fall-through Behavior in Switch

- JavaScript supports the fall-through behavior
  - i.e. if a case statement misses a break, the code for the next cases is also executed
    - Until a break is found

```
switch (day) {
  case 1:
  /* 2, 3 and 4 */
  case 5:
    console.log('Working day'); break;
  case 6:
  case 7:
    console.log('Weekend!'); break;
  default:
    console.log('Error!'); break;
}
```

The Fall-through Behavior in Switch

Live Demo

Telerik Academy

Expressions in the Case Label

- In JavaScript, the case label can be an expression
  - Useful when in need to check ranges
    - Yet, kind of confusing, better use if-else statements
  - The cases are evaluated from top to bottom
    - The first reached break exits the switch

```
switch (false) {
  case !!score: // true when score is NaN
  case !(score < 2 || score > 6):
    console.log('Invalid score'); break;
  case !(score < 3.5):
    console.log('Poor' + score); break;
  case !(score < 4.5):
    console.log('Good ' + score); break;
  /* case for score < 5.5 */
  default:
    console.log('Excellent ' + score); break;
}
```

Expressions in the Case Label

Live Demo

Telerik Academy

Truthy and Falsy Values

First steps in the dynamic beauty of JavaScript

Truthy and Falsy Values

- Every type in JavaScript has a inherent Boolean value
  - So called truthy (TRUE-like) and falsy (FALSE-like) values
- These values are falsy
  - false, 0 (zero), "" (empty string), null, undefined, NaN
- All other values are truthy
- Info: http://www.sitepoint.com/javascript-truthy-falsy/

Telerik Academy

Truthy and Falsy Values

Live Demo

Summary

- Comparison and logical operators are used to compose logical conditions
- The conditional statements `if` and `if-else` provide conditional execution of blocks of code
  - Constantly used in computer programming
  - Conditional statements can be nested
- The `switch` statement easily and elegantly checks an expression for a sequence of values
  - Supports the fall-though behavior
  - Can contain expressions in the case value

Telerik Academy

Conditional Statements



Questions?



http://academy.telerik.com

Free Trainings @ Telerik Academy

- "Web Design with HTML 5, CSS 3 and JavaScript" course @ Telerik Academy
  - html5course.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com