`if` и `if-else`

Реализация условной логики

`if` утверждение

- Самое простое условное утверждение
- Позволяет создавать условия проверки утверждения
- Создает ветки исполняемого кода, в зависимости от соответствия условию

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

Условия и утверждения

- Условие может быть:
  - Булевой переменной
  - Булевым логическим выражением
  - Выражения сравнения
  - Цифры,объекты, функции, все что угодно!
- Утверждение может содержать любые типы данных
- Утверждение может быть:
  - Простым утверждением, заканчивающимся `';'`
  - Блоком заключенным в фигурные скобки ``'{}'``

Как это работает?

- Условие оценивает
  - если `true` - подобное условие, утверждение исполняется
  - если `false` -подобное, утверждение пропускается

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

```
var str = '1c23';
if(!(+str)){ // если строка не номер, +str хранит NaN
  throw new Error('str не номер!');
}
```

`if-else` утверждение

- Более комплексное и применимое утверждение
- Вызывает код в одной ветке если `true` и в другой если `false`

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

Как это работает?

- Условие оценивает
  - Если `true`-подобное, вызывается первое утверждение
  - Если `false`-подобное, вызывается второе утверждение
- Проверяет число на четность

```
var s = '123';
var number = +s;
if (number % 2) {
    console.log('Это число четное.');
} else {
    console.log('Число нечетное.');
}
```

```
if (+str) {
  console.log('str является числом');
} else {
  console.log('str не является числом');
}
```

Если str не является строкой условие вернет NaN (FALSE-подобие)
Так же как и if(number % 2 === 1)

Вложенные `if` утверждения

- `if` и `if-else` могут быть **вложенными**, то есть использоваться внутри `if` и `else` утверждений

-

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

- Всегда используйте `{ … }` чтобы избежать неоднозначности
  - Даже когда используется одноуровневое условие
- Старайтесь избегать появления более 3 вложенным `if` утверждений
- Старайтесь делать код более читабельным

```
if (first === second) {
  console.log('Эти два числа одинаковы.');
} else {
  if (first > second) {
    console.log('first больше.');
  } else {
    console.log('second больше.');
  }
}
```

```
var n = +str;
if (n) {
  if (n % 2) {
    console.log('Число четное');
  } else {
    console.log('Число нечетное');
  }
} else { //n is NaN
  console.log('Это не число!');
}
```

Множество `if-else-if-else-…`

- Иногда Вам надо использовать `if` проверку условия в `else` блоке
  - Здесь может быть применимо `else if` выражение:

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

Несколько вариантов условных переходов (`switch-case`)

- Выбирает условие из перечня доступных вариантов с помощью `switch`

```
switch (day) {
    case 1: console.log('Понедельник'); break;
    case 2: console.log('Вторник'); break;
    case 3: console.log('Среда'); break;
    case 4: console.log('Четверг'); break;
    case 5: console.log('Пятница'); break;
    case 6: console.log('Суббота'); break;
    case 7: console.log('Воскресенье'); break;
    default: console.log('Ошибка!'); break;
}
```

- Выражение вычисляется
- Когда одна из указанных констант соответствует выражению
- Если нет не одной константы соответствующей выражению
    - Если есть выражение пол умолчанию, то оно вызывается
    - В противном случае проверка завершается и управление кодом передается за пределы блока

Поведение "следовать дальше" в `switch`

- JavaScript поддерживает поведение "следовать дальше"
    - т.е если в `case` утверждении отсутствует `break` , код продолжает выполняться дальше
        - Вплоть до появления оператора `break`

```
switch (day) {
  case 1:
  /* 2, 3 и 4 */
  case 5:
    console.log('Рабочий день'); break;
  case 6:
  case 7:
    console.log('Выходной!'); break;
  default:
    console.log('Ошибка!'); break;
}
```

Выражения в блоке `case`

- В JavaScript, `case` метка может содержать выражения
  - Используется когда надо проверить диапазоны
    - Тем не менее не очень читабельно, поэтому лучше применять `if-else` утверждения

```
switch (false) {
  case !!score: // true when score is NaN
  case !(score < 2 || score > 6):
    console.log('Неправильная оценка'); break;
  case !(score < 3.5):
    console.log('Плохая оценка' + score); break;
  case !(score < 4.5):
    console.log('Хорошая оценка ' + score); break;
  /* case for score < 5.5 */
  default:
    console.log('Превосходная оценка ' + score); break;
}
```

Истинные и ложные значения

- Каждый тип в JavaScript может быть выражен через булево значение
  - Так как вызываеются истинные (TRUE-подобные) и ложные (FALSE-подобные) значения
- Эти значения ложные
  - `false`, `0`, `""` / `''`, `null`, `undefined`, `NaN`
- Все остальные значения истинные
- Информация: http://www.sitepoint.com/javascript-truthy-falsy/