infoShare ACADEMY

JavaScript instrukcje warunkowe i pętle

infoShare Academy

infoShareAcademy.com

HELLO

Maciej Mikulski

Senior Front End Developer
@JIT.Team @Dialecticanet.com

infoShare ACADEMY

01. Materiały

Bibliografia

if {} else {}

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/if...else

while

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/while

do while

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/do...while

for

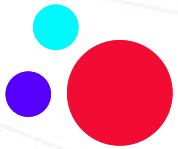
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for

try {} catch(error) {}

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/try...catch

if..else, switch

infoShareAcademy.com



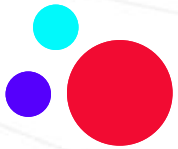
Jeżeli coś to co?

If to podstawowa instrukcja warunkowa (conditional statement), która pozwala nam na wykonanie operacji wtedy i tylko wtedy gdy wskazany warunek jest prawdziwy (w rozumieniu JavaScript – czyli truthy).

```
if (true) {
  // zrób coś
}
```

Umożliwia także podanie alternatywy

```
if (true) {
  // zrób coś
} else {
  // zrób coś innego
}
```



Zagnieżdżone instrukcje warunkowe

Instrukcje warunkowe możemy zagnieżdżać, czyli po sprawdzeniu jednego warunku, uzależnić wykonanie instrukcji od kolejnego warunku:

```
if (warunek1) {
   if (warunek2) {
    // zrób coś
   } else {
    // zrób coś innego
   }
} else {
   // zrób coś jeszcze innego
}
```

Trójargumentowy operator warunkowy

“Ternary operator” to instrukcja, która pozwala nam zapisać instrukcję warunkową if..else w skrócony sposób:

```
const prawdaCzyFałsz = true;

const wynik = prawdaCzyFałsz ? ‘ok’ : ‘błąd’;
```

Najczęściej używamy wtedy gdy chcemy określić wartość zmiennej na podstawie warunku.

Operatory można zagnieżdżać podobnie jak if..else, ale należy tego unikać ze względu na nieczytelność.

switch

Operator switch pozwala nam na określenie działania dla wielu możliwych warunków:

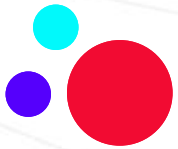
```
const dzień = ‘poniedziałek’; // poniedziałek | wtorek | środa | czwartek | piątek | sobota | niedziela
switch (dzień) {
  case ‘poniedziałek’: {
    console.log(‘uhh’);
    break;
  }
  case ‘środa’:
  case ‘wtorek’: {
    console.log(‘ehh’);
    break;
  }
  case ‘piątek’: {
    console.log(‘yes!’);
    break;
  }
  default {
    console.log(‘mhm’);
  }
}
```

Switch c.d.

W przypadku instrukcji switch powinnismy pamiętać o tym

- czy poszczególne przypadki są rozłączne i wymagają instrukcji break bez którego wykonanie przejdzie do kolejnego warunku (tzw. ‘fall-through’);
- Jaki jest przypadek domyślny,

While & do while



While – pętla warunkowa

Pętla warunkowa while, pozwala na powtarzanie instrukcji tak długo, aż zostanie spełniony określony warunek.

```
let dystans = 0;
while (dystans < 42) {
  console.log('biegnę');
  dystans++;
}
```

do while – z przynajmniej jednym wykonaniem

W przeciwieństwie do zwykłej pętli while, w której warunek sprawdzany jest przed wykonaniem instrukcji, pętla do...while sprawdza warunek po wykonaniu operacji.

```
let głodny;
do {
  console.log(‘zjedz ciastko’);
  głodny = confirm('Czy nadal jesteś głodny?’);
} while (głodny !== false)
```

Break & continue

Na wykonywania pętli możemy wpłyną instrukcjami break oraz continue.

Break – przerywa wykonanie całej pętli.

Continue – przerywa wykonanie bieżącego kroku I przechodzi do następnego.

for

infoShare ACADEMY

Pętla warunkowa for upraszcza operacje, w których chcemy przejść przez szereg wartości.

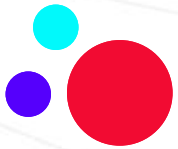
```
for (let i = 0; i < 9; i++) {
  console.log(i);
}
```

Możemy iterować nie tylko o jeden element:

```
for (let i = 0; i < 9; I += 3) {
  console.log(i);
}
```

try...catch

infoShare ACADEMY



Try...catch czyli złap mnie jeśli potrafisz

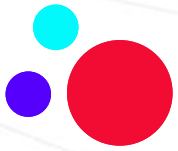
Jeśli spodziewamy się błędu podczas wykonania operacji, możemy się na to przygotować używając konstrukcji try..catch, któa pozwala nam określić działanie awaryjne.

```
try {
  bankomat.wypłaćStówę();
  console.log('jest kasa');
} catch (error) {
  console.log('nie ma kasy');
}
```

Try...catch ...finally

Blok finally jest opcjonalny i pozwala określić działanie, niezależne od wyniku wcześniejszej próby.

```
try {
  bankomat.wypłaćStówę();
  console.log(‘zamawiamy pizzę’);
} catch (error) {
  console.log(‘nie zamawiamy pizzy’);
} finally {
  console.log(‘oglądamy mecz’);
}
```



Zagnieżdżone Try...catch

Blok finally jest opcjonalny i pozwala określić działanie, niezależne od wyniku wcześniejszej próby.

```
try {
  bankomat.wypłać(100);
  console.log(‘zamawiamy pizzę’);
} catch (error) {
 console.log(‘nie zamawiamy pizzy’);
  try {
    bankomat.wypłać(50);
    console.log(‘zamawiamy frytki’)
  } catch {
    console.log(‘fryteczki też nie ☹’)
  }
}
```

Dziękuję za uwagę!

infoShareAcademy.com