Skripta ke školení

JavaScript

Autor: Tomáš Herout

E-mail: herout@helpmark.cz

Telefon: (+420) 739 719 548

Obsah

Velice stručná definice, co je to JavaScript 2
Pravidla zápisu JavaScriptu (syntax) 3
Základní stavební kameny JavaScriptu 5
alert() 5
confirm() 5
document.write() 5
var promenna 5
var pole 6
if |else if | else 6
switch 8
for – cyklus 8
do while – cyklus 8
Math – matematické operace 9
random – generátor náhodného čísla 10
Date – práce s datumem a časem 10
function – zabalení JS kódu 11
setTimeout – spustit v nastaveném čase 12
Spouštění JavaScriptu v praxi 13
Vyhledání HTML elementů v dokumentu 13
Spuštění na událost 15

HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

Tato skripta slouží jako podklad pro školení začátečníků. Nečiní si ambice stát se plnohodnotnou učebnicí JavaScriptu a rozhodně neobsahují vše o JavaScriptu. Obsahují to, co považuji za důležité pro zvládnutí JavaScriptu od úplných základů. Pro hlubší studium doporučuji některou ze standardních učebnic.

Velice stručná definice, co je to JavaScript

JavaScript je jazyk, kterým můžete dávat pokyny vašemu internetovému prohlížeči (Chrome, Firefox…), aby něco dělal. Může provádět třeba animace, kontrolu polí formuláře před odesláním, výpočty ve webové kalkulačce, zasílat a přijímat od serveru data, měnit obsah stránky bez nutnosti načítat HTML a mnoho dalšího. Kód JavaScriptu se stahuje do počítače návštěvníka spolu s HTML a dalšími kódy, jako je třeba CSS.

Neplést si JavaScript a Java. Java je něco zcela jiného.

Ukázka vložení JavaScriptu do HTML dokumentu:

```
<!DOCTYPE html>
<html>
      <head>
            <script>
            ...
            </script>
      </head>
      <body>
      ...
      </body>
</html>
```

JavaScript většinou pracuje s HTML kódem a s CSS. Proto znalost HTML a CSS je pro zvládnutí JavaScriptu nezbytná.

Soubory JavaScriptu mají příponu *.js. Je možné je otevřít v libovolném textovém editoru. Kód bude ale nepřehledný, proto je lepší zvolit nějaký vhodnější nástroj, třeba Notepad++, PSPad nebo Aptana. PSPad zobrazí barevně jednotlivé části kódu pro lepší přehlednost. Aptana také, ale ještě toho umí daleko více, třeba našeptat kód.

HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

Pravidla zápisu JavaScriptu (syntax)

Každý příkaz se odděluje středníkem. Mezery, tabulátory… neznamenají ukončení příkazu, používají se jen z důvodu přehlednosti kódu.

```
<script>
   var promenna = 'Ahoj';
   alert(promenna);
</script>
```

Vysvětlení:

První a posední řádek jen označují v HTML kódu vložený JavaScript.
Druhý řádek vytváří proměnnou s názvem „proměnná“ a hodnotou „Ahoj“. Středník na konci řádku ukončuje příkaz. Třetí řádek vytváří další příkaz, který zajistí zobrazení okna s textem proměnné, tedy „Ahoj“.

Hodnoty se ukládají do apostrofů nebo uvozovek. V příkladu výše to již bylo vidět na řetězci „Ahoj“. V příkladu byly použity apostrofy, ale stejně tak dobře by posloužili i uvozovky. Rozdíl je v tom, že uvnitř apostrofů není problém použít uvozovky nikoli jako příkaz JavaScriptu, ale prostě jako textové uvozovky. Pozor rozdíl je u čísel. Pokud chcete čísla používat jako počitatelná čísla (sčítat je, odčítat…), tak se uvádějí bez uvozovek nebo apostrofů. Zápis '1' znamená, že číslo již nebude počitatelné. Zápis '1' + '1' vrátí jako výsledek 11.

Ke spojování dvou a více hodnot se používá znaménko +. To už bylo nakousnuto v předchozím odstavci. Znaménko + může mít matematický význam, ale u textových řetězců je prostě spojí k sobě, a to bez mezery.

Kulaté závorky () pro předávání hodnot. Kulaté závorky jde v JavaScriptu používat stejně, jako v matematice. Ale mají navíc další funkci, kterou jste viděli hned v prvním příkladu

```
alert(promenna);
```

U příkazu `alert` jsme do kulatých závorek vložili hodnotu, kterou chceme zobrazit ve varovném okně internetového prohlížeče. Více se o kulatých závorkách dozvíte u funkcí.

Složené závorky {} pro ohraničení kódu. Nejlépe to bude vidět na příkladu:

```
...
if(promenna > 2){
      alert(promenna);
}
```

Kulaté závorky předávají hodnotu k posouzení, zda je to pravda, či nikoli. Příkaz `if` se používá právě k tomuto posuzování. V případě, že `promenna` opravdu bude větší, než 2, tak se vykoná to, co je ve složených závorkách. Složené závorky tedy ohraničují kusy kódu.

Význam tečky při zápisu v JS.

```
document.getElementById('hlavni-menu');
```

Tečky mají při zápisu v JavaScriptu speciální význam. Pokusíme se to vysvětlit na příkladu, kdy JS kód výše, se snaží vyhledat nějaký prvek který by mohl v HTML kódu vypadat třeba takto:

```
...
<div id="hlavni-menu">
...
</div>
```

Zápis: `document.getElementById('hlavni-menu');` říká, že nejprve hledáme v dokumentu HTML (`document`) a uvnitř něj hledáme id (`getElementById`). Tečka nám tedy v Javascriptu říká, že hledáme něco vnořeného uvnitř.

Pozor! Tečka může mít samozřejmě i jiné významy v závislosti na jejím použití. Tečkou se určuje desetinné místo při zápisu čísel, takže polovinu zapíšeme jako `0.5`. Význam tečky v URL adresách už je poměrně známý. V regulárních výrazech má zase tečka význam zástupného symbolu za jakýkoli znak.

HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

Základní stavební kameny JavaScriptu

alert()

Internetové prohlížeče na něj reagují tak, že zobrazí varovné okno. Do tohoto okna mohou vypsat hodnotu uloženou do závorek. To je velice užitečné, pokud si vývojář chce třeba ověřit, co má v určitém místě kódu uložené do proměnné. Dalším použití je test, zda určitý kus kódu vůbec je čten prohlížečem. Původní určení je pro chybové hlášky a samozřejmě i za tímto účelem je možné ho použít, ale nevypadá to zrovna elegantně.

Pozor! Při zobrazení okna se zastaví veškerý ostatní kód, který je pod `alert()`.

confirm()

Tak, jako `alert()` i `confirm()` zobrazí okno v prohlížeči. Rozdíl je ale v tom, že návštěvníkovi webu nabídne navíc i možnost potvrdit, případně stornovat. Tvůrci kódu se to hodí v případě, pokud si do proměnné převezme výsledek. Pokud návštěvník klikne na [OK], pak dostanete do proměnné pravda. Při kliknutí na [Zrušit] dostanete nepravda. Zastavení kódu platí stejně jako u `alert()`.

document.write()

Umí do HTML kódu tam, kde se tento JavaScript nachází, vypsat obsah umístěný do kulatých závorek.

var promenna

Slouží k ukládání hodnot do paměti. Tou hodnotou může být: číslo, text (tzv. textový řetězec), pravdivostní hodnota (pravda nebo nepravda).

Zápisy čísel do proměnných:

```
var číslo = 1.5;
```

Zápisy textů do proměnných:

```
var text = 'Jakýkoli text i číslo 1, ale jen jako text';
```

Spojování proměnných pomocí znaménka +:

```
var spojeni = text + ' a text za ním';
```

Zjištění počtu znaků v řetězci

```
var promenna = 'zjistíme délku';
alert(promenna.length); // vrátí číslovku 14
```


HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

var pole

Pole (anglicky „array“) je vlastně balíček více proměnných. V praxi ho můžete použít třeba jako nějaký seznam o určitém množství položek, jejichž počet ale dopředu programátor nezná. Jednotlivé položky v poli mají vždy své pořadí, které se ale počítá od čísla 0 (nula). Položky však mohou mít i názvy, pokud jim tyto názvy určíte.

U polí jsou klíčové hranaté závorky […].

Jeden z druhů zápisu pole

```
var pole = ['nultá položka', 'první položka', 'druhá položka'];
```

Způsob vložení položky s názvem

```
var pole['nazev'] = ['hodnota'];
```

Zjištění délky pole pomocí length

```
var pole = ['nultá položka', 'první položka', 'druhá položka'];
alert(pole.length); // vrátí číslovku 3
```

Abecední seřazení prvků v poli

```
var pole = ['Karel', 'Franta', 'Pepa'];
pole.sort();
alert(pole); // vrátí pole abecedně setříděné
```

if | else if | else

Slouží k rozhodování podle určitých kritérií. Mohli bychom to také přirovnat k rozcestí, případně odbočce. Za `if` do kulatých závorek se píší podmínky. Např.: `if(1>0)`. Tato podmínka je pravdivá, proto se vykoná to, co je ve složených závorkách. Takto postavená podmínka samozřejmě nedává smysl, ale pro pochopení je výmluvná.

Jednoduchý příklad, kdy otestujeme, zda proměnná existuje. Pokud ano vypíšeme ji do okna. Pokud existovat nebude, nestane se nic:

```
if(promenna > 0){
    alert('Je větší, než nula');
}
```

složitější příklad, testu, zda proměnná existuje. Pokud ano vypíše ji. Pokud ne, oznámí absenci proměnné.

```
if(promenna > 0){
    alert('Je větší, než nula');
} else{
    alert('Není větší, než nula nebo je rovna nule');
}
```


HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

Komplexní příklad zde, kdy předchozí kód zjistí zvolené pohlaví. JavaScript pak otestuje více variant:

```
if(volba_pohlavi == 'muz') {
    alert('Ahoj chlape');
} else if(volba_pohlavi == 'zena'){
    alert('Dobrý den, dámo');
} else{
    alert('Zdravím tě, věci');
}
```

Podmínky můžete i spojovat. Ukázka nutnosti splnit dvě podmínky současně:

```
if(promenna == 1 && promenna2 == 2) {
    // kód
}
```

Ukázka potřeby splnit alespoň jednu z možností:

```
if(promenna == 1 || promenna == 2) {
    // kód
}
```

Tabulka operátorů používaných v podmínkách

Operátor	Ukázka	Popis
==	x==y	Pravda, jestliže x a y jsou stejné
===	x===y	Pravda, jestliže x a y jsou stejné a mají i stejný typ (např. číslo)
!=	x!=y	Pravda, jestliže x a y nejsou stejné
>	x>y	Pravda, jestliže x je větší, než y
>=	x>=y	Pravda, jestliže x je větší nebo rovno y

switch

Má podobou funkci jako `if`. Rozdíl je v tom, že nemá možnost podmínky skládat. Jeho značná výhoda se projeví v okamžiku více možností výběru podmínky.

Ukázka zapsání switch

```
switch (promena){
    case 'Karel':
    // kód;
    case 'Václav':
    // kód;
    case 'Petr':
    // kód;
    default:
    // kód v případě, že žádná z možností nebyla pravda;
}
```

for – cyklus

Cykly slouží k opakování nějakých kódů. Často se používají ke zpracování polí, ale mohou se používat i k jiným účelům. U cyklů je důležité, že vždy musejí mít okamžik, kdy cyklus skončí. Nelze je tedy používat například pro nekonečné střídání obrázků.

Ukázka vypsání pole v cyklu for

```
var pole = ['pondělí', 'úterý', 'středa'];
for( neco in pole ){
     document.write(pole[neco]);
}
```

Ukázka cyklu for mimo pole

```
for( i = 1; i <= 10; i++  ){
     document.write(i);
}
```

do while – cyklus

Cyklus, který stejně jako `for` provádí zadané operace tak dlouho, dokud je zapotřebí. Rozdíl je v tom, že operaci proveden minimálně jednou i když podmínka není splněna. Cyklus `for` totiž tuto vlastnost nedokáže.

HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

Praktická ukázka vypsání pole pomocí do while

```
var pole = ['Karel', 'Václav', 'František'];
var opakovani_cyklu = 0;
do {
   if( pole.length > 0){
     document.write(pole[opakovani_cyklu]);
   }else{
     document.write('Je nám to líto, ale žádná osoba nenalezena');
   }
   opakovani_cyklu++;
}
while( opakovani_cyklu < pole.length )
```

Math – matematické operace

Javascript umí používat matematické operandy, jakým je plus +, mínus -, krát ×, děleno /. Navíc JS umí ++, které číslo navýší o jednu a naopak - - jež funguje opačně. Při práci s čísli ale někdy potřebujeme i další matematické operace, jako např.:

Math.round – zaokrouhlení matematické

```
Math.round(4.7);          // vrátí 5
Math.round(4.4);          // vrátí 4
```

Math.floor – zaokrouhlení vždy dolů

```
Math.floor(4.7);          // vrátí 4
Math.floor(4.4);          // vrátí 4
```

Math.ceil – zaokrouhlení vždy nahoru

```
Math.ceil(4.7);           // vrátí 5
Math.ceil(4.4);           // vrátí 5
```

Math.abs – vrátí absolutní hodnotu

```
Math.abs(4);              // vrátí 4
Math.abs(-4);             // vrátí 4
```

Math.pow – vypočítá mocninu z čísla

```
Math.pow(8, 2);           // vrátí 64
```


HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

Math.sqrt – vypočítá odmocninu

```
Math.pow(64);                // vrátí 8
```

Math.sin – vypočítá sinus

```
Math.sin(1);                 // vrátí 0.8414709848078965
```

Math.cos – vypočítá cosinus

```
Math.cos(1);                 // vrátí 0.5403023058681398
```

Math.tan – vypočítá tangent

```
Math.tan(1);                 // vrátí 1.5574077246549023
```

random – generátor náhodného čísla

V jakémkoli programování je někdy zapotřebí vygenerovat náhodné číslo např. pro zvolení náhodného obrázku v animaci. K tomuto účelu slouží v JavaScriptu `random()`, který umí vygenerovat číslo mezi 0 až 1. Tedy např. číslo 0.4887240220650442.

Příklad generování čísla mezi 1 až 10

```
Math.floor((Math.random() * 10) + 1);
```

Příklad generování čísla mezi 5 až 15

```
Math.floor((Math.random() * 11) + 5);
```

Date – práce s datumem a časem

Datum a čas je leckdy potřeba i v JavaScriptu. Proto se podíváme, jak s datumem pracovat.

Příklad vypsání aktuálního datumu a času

```
document.write( Date() );
```

Příklad vypsání jen číslovky dne

```
document.write( new Date().getDate() );
```


HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

Tabulka s některými metodami objektu Date

getDate()	Den v měsíci (1-31)
getDay()	Číslo dne. 0 = neděle, 1 = pondělí…, 6 = sobota
getMonth()	Číslo měsíce (0 - 11) *pozor o jedno nižší*
getFullYear()	Rok (např. 2018)
getHours()	Hodina (0 - 23)
getMinutes()	Minuty (0 - 59)
getSeconds()	Sekundy (0 - 59)
getMilliseconds()	Milisekundy (0 - 999)
getTime()	Počet milisekund od 1. 1. 1970

function – zabalení JS kódu

Funkce se používají k zabalení výše popsaných JavaScriptových kódů. Proč nějaké kódy balit? Důvodů může být víc, ale mezi ty hlavní patří přehlednost, lepší strukturovatelnost a v neposlední řadě možnost spustit kód ve správný okamžik, klidně i opakovaně, pokud to je třeba.

Jak se funkce vytvářejí

```
function nazev(){
      alert("Nějaká informace");
}
```

Při tvorbě funkce je klíčové slovo `function`. Pak následuje název, který si můžete volit dle svého uvážení. Zvykem bývá, že funkce začíná malým písmenem, ale nedodržením tohoto pravidla se nic nestane. Za názvem musí následovat kulaté závorky, byť by měly být prázdné. Je do závorek je ale možné vložit nějaké hodnoty (proměnné nebo pole) a přenést je do funkce.

Ukázka zavolání funkce i s předáním argumentu

```
function nazev(pormenna){
      alert(pormenna);
}
var promenna = 1;
nazev(promenna);
```


HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

setTimeout – spustit v nastaveném čase

JavaScript se často používá pro obrazové animace, které pochopitelně potřebují i časování. Z toho důvodu je v JS velice užitečné umět spustit nějakou akci (třeba výměnu obrázku) až za určitý čas. K tomu slouží `setTimeout`, který se zapisuje takto:

```
setTimeout(function(){
      // udělej něco; },
      3000);
```

Vysvětlení:

1. Slovem `setTimeout` požádáte prohlížeč, aby odložil spuštění toho, co je uvnitř
2. Kulaté závorky `(…)` ohraničují obsah `setTimeout`
3. Slovem `function()` zajistíte spuštění kódu, který má něco provést. Zde nedáváte funkci žádný název, vykoná se hned
4. Složenými závorkami `{…}` ohraničíte samotný kód, který se bude provádět
5. Číslovkou na konci určíte dobu, po kterou bude prohlížeč čekat na spuštění. Číslovka určuje milisekundy. Tedy `3000` jsou 3 sekundy.

HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

Spouštění JavaScriptu v praxi

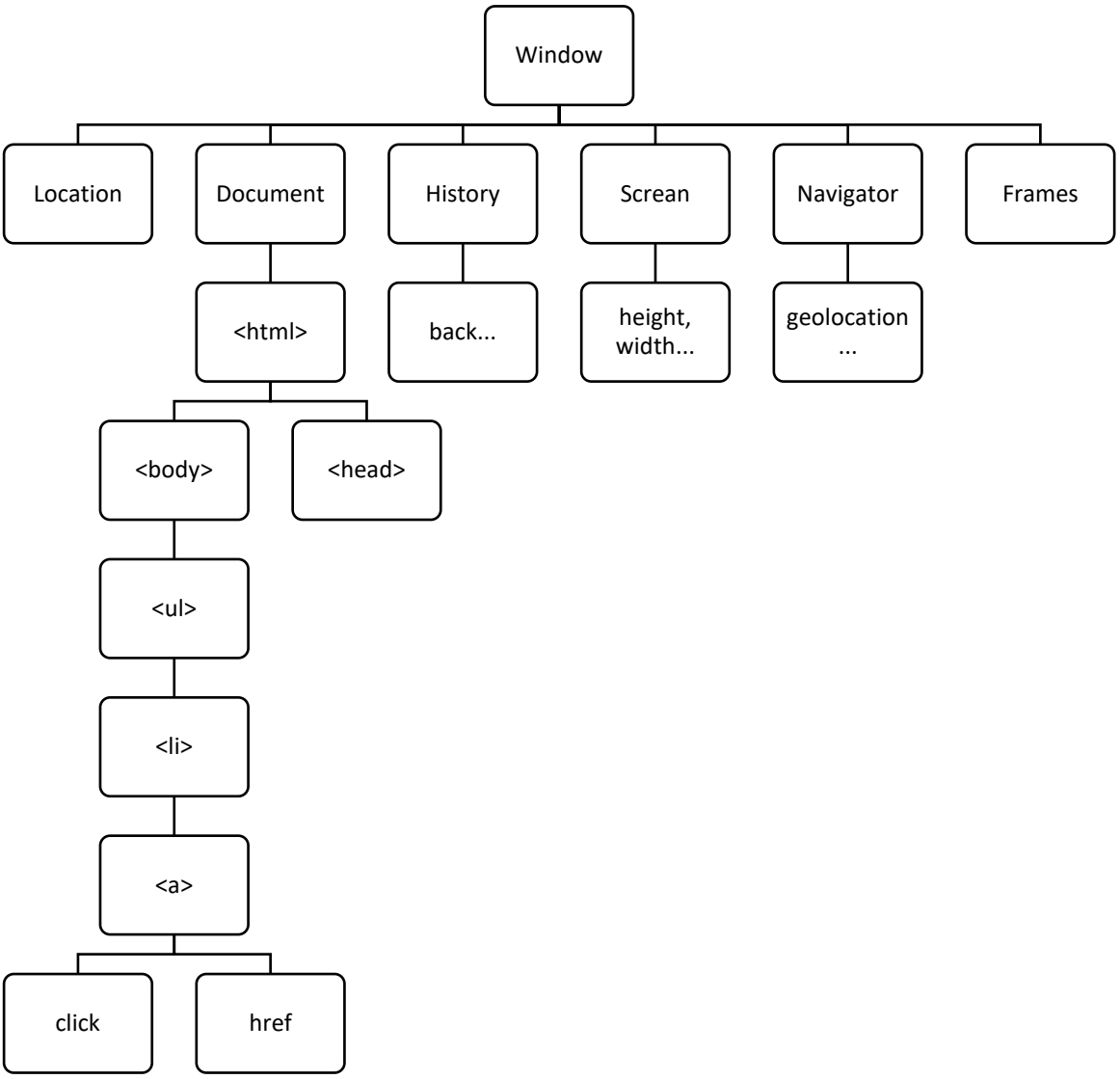
V JavaScriptu na straně klienta (internetového prohlížeče) jsou obecně důležité následující kroky:

1. Najít HTML prvky (např. tlačítka ke spuštění skriptu, obrázky, DIVy…)
2. Spustit kód v definovaný okamžik (po načtení, kliknutím na tlačítko…)
3. Provést požadovanou operaci, která závislá od toho, co má kód provést

Vyhledání HTML elementů v dokumentu

HTML má danou strukturu po které se JS umí „pohybovat“. Této struktuře se říká DOM (Document Object Model)

DOM



HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

K procházení HTML kódu je vhodnější použít knihovnu, např. jQuery. Jde to ale i bez ní. Nejběžnějším způsobem, jak vyhledat HTML element je přes `id`. Způsob zápisu:

```
document.getElementById("nazev-id");
```

Dalším řešením je vyhledat prvky podle `class`. Vyhledání přes `id` je snazší v tom, že `id` by se v HTML dokumentu mělo nacházet pouze jednou. Naproti tomu `class` se v jednom dokumentu může objevit vícekrát, což v praktickém použití znamená složitější kód. Způsob zápisu:

```
document.getElementsByClassName("nazev-tridy");
```

Praktický příklad

```
var x = document.getElementsByClassName("nazev-tridy");
var i;
for (i = 0; i < x.length; i++) {
    x[i].style.backgroundColor = "red";
}
```

Jak je z příkladu zřejmé, tak při vyhledání třídy, musíte procházet výsledek hledání jako pole.

Pokud chcete vyhledat pouze první prvek, můžete použít:

```
document.querySelector(".nazev-tridy ");
```

Další možností je vyhledat konkrétní HTML značky. Způsob zápisu:

```
document.getElementsByTagName("a");
```

I v tomto případě logicky dostanete jako výsledek pole, jelikož značek `<a>` se v HTML dokumentu může nacházet více. Platí tudíž totéž, co v případě `getElementsByClassName`.

Někdy můžete chtít přidat nový prvek na konec již existujících (např. další <li> do seznamu). K tomu se hodí:

```
document.getElementById("nazev").appendChild(neco);
```

V případě vložení na začátek můžete použít:

```
document.getElementById("nazev").insertBefore(neco);
```


HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**

Spuštění na událost

Pokud už umíte HTML prvek vyhledat, tak můžete sledovat události a reagovat na nastalou událost spuštěním požadovaného kódu.

Tabulka s nejpoužívanějšími druhy událostí:

click	Kliknutí myší
mouseenter	Umístění myši nad prvek
mouseleave	Opuštění myši prvku, na kterém předtím myš byla
keypress	Stisknutí klávesy (např. pro <input type="text">)
change	Při změně obsahu. Používá se pro <input>, <select>, <textarea>
focus	Aktivnost prvku
blur	Ztráta aktivnosti prvku (hodí se pro kontrolu obsahu pole ve formuláři)
load	Načtení elementu (dá se použít pro aktivaci po načtení <body>)
drop	Chycením kurzorem myši za účelem přetažení na jinou pozici

Způsob zápisu přímo do HTML kódu. Ukázka:

```
…
<script>
      function nazev_funkce() {
            alert('test')
      }
<script>
…
<a onclick=" nazev_funkce " >
…
```

Způsob zápisu do JavaScriptu. Ukázka:

```
…
<script>
  document.getElementById("nazev-id").onclick=function(){
      alert('test');
  };
<script>
…
<a id="nazev-id">
…
```


HelpMark
www.helpmark.cz |email: **herout@helpmark.cz** | tel.: **739 719 548**