**Maturitní témata 2020/21**

**Otázka č. 19 – Počítačové sítě a programování**

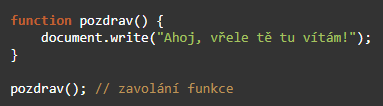
**Javascript:**

- multiplatformní, objektově orientovaný skriptovací jazyk, jeho syntaxe patří do rodiny jazyků C/C++/Java, ale od těchto jazyků se zásadně liší funkčně a principem

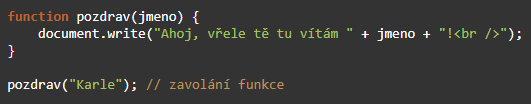
- doplněk k jazykům HTML a Java, slouží k tvorbě webových stránek - vkládaný je často přímo jako součást HTML kódu stránky (jsou jím tvořeny různé prvky GUI např. tlačítka, textová políčka nebo tvořeny animace a efekty obrázků), využívá se i na straně serveru

**Funkce:**

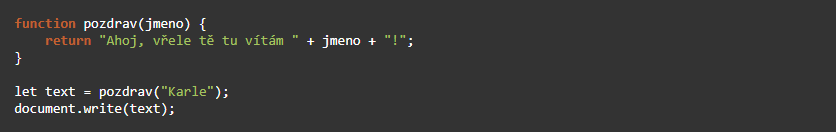
- bok kódu který jednoduše napíšeme a potom ho můžeme libovolně volat bez toho abychom ho psali znovu a opakovali se, funkci deklarujeme pomocí klíčového slova **function** a obsahuje blok kódu ve složených závorkách, funkci bychom měli volat až poté co ji deklarujeme



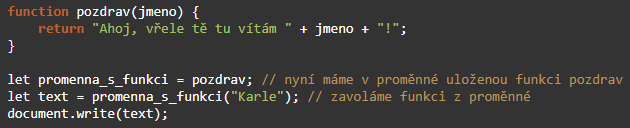
- funkce může mít také libovolný počet vstupních parametrů které píšeme do závorky v její definici a podle nich ovlivňujeme její chování



- funkce může dále vracet nějakou hodnotu, např. abychom nemuseli do dokumentu rovnou zapisovat (např. proto, že budeme chtít s textem ještě dále pracovat a ne ho rovnou vypsat), slouží k tomu příkaz **return**



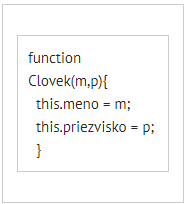
- JavaScript se liší od jiných jazyků tím jak pracuje s funkcemi, tzv. funkcionální paradigma - jedná se o specifický styl programování a myšlení pomocí funkcí, funkci můžeme totiž uložit do běžné proměnné a z této proměnné ji později volat



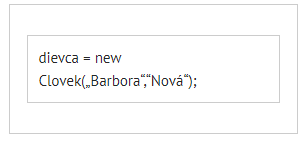
**Objekty:**

- při tvorbě skriptů se často používají objekty, JavaScript nám nabízí některé zabudované objekty např. pole (Array), většinou nás při tvorbě skriptů zajímají vlastnosti a metody těchto objektů

- objekt je datový typ, každý objekt má (ale nemusí) svoje vlastnosti a metody, funkce i pole jsou objektem, **object** je v hierarchii všech typů nejvýše



- tato funkce je tzv. konstruktorem objektu, samotný objekt potom vytvoříme pomocí operátoru **new**

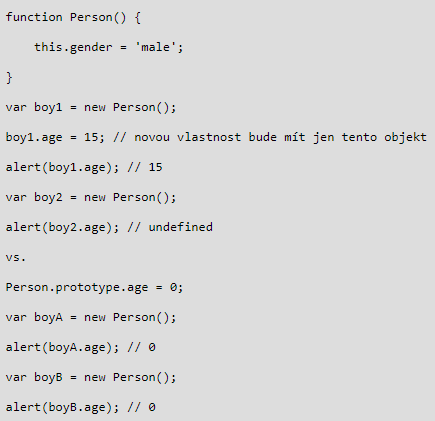


**Prototypy:**

- JS objekty jsou založeny na prototypu, kdy se oproti Javě tolik nerozlišuje rozdíl mezi třídou a objektem (instancí třídy)

- v JS je každý objekt asociován s objektem **prototype** který nese vlastnosti třídy (říká jak má vypadat objekt), vlastnosti dané třídy jsou zapouzdřeny v prototypu proto aby se mohli dědit, i funkce je objekt takže také má prototype, v JS neexistují třídy v pravém slova smyslu 🡪 proto se používá **prototype**

- pokud přidáme novou vlastnost přímo objektu, jiný objekt stejné třídy ji mít nebude ale vlastnosti celé třídy lze modifikovat a to přes prototyp

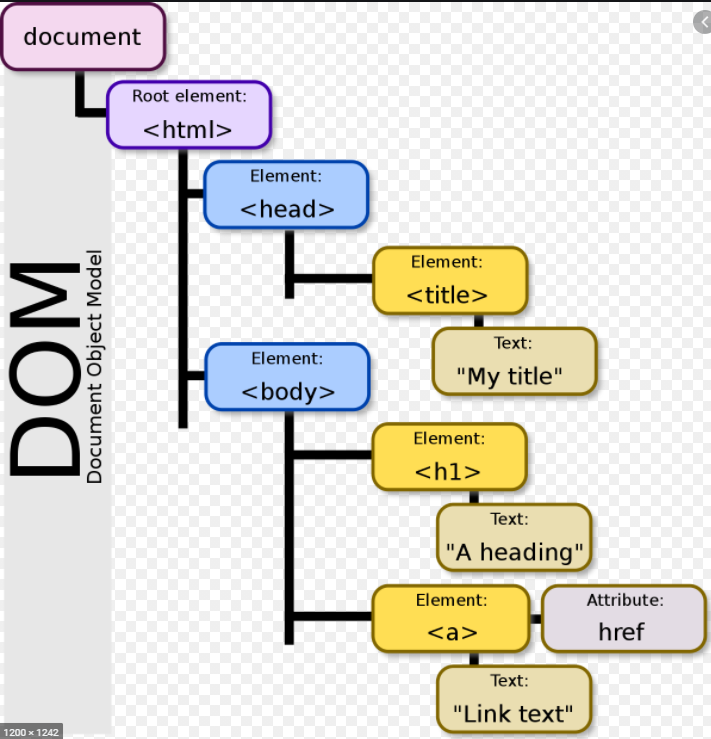


**HTML DOM:**

- HTML Document Object Model je objektově orientovaná reprezentace HTML dokumentu, DOM je API (rozhraní pro programování aplikací, jde o sbírku procedur, funkcí a tříd) umožňující přístup či modifikaci obsahu, struktury nebo stylu dokumentu či jeho částí

- původně měl každý webový prohlížeč své vlastní specifické rozhraní k manipulaci s HTML elementy pomocí JavaScriptu, DOM umožňuje dokumentu přístup dokumentu jako ke stromu což je zároveň datová struktura

- specifikace DOM jsou rozděleny do několika úrovní (DOM level 1-5), z nichž každá obsahuje povinné a volitelné moduly, k tomu aby nějaká aplikace mohla prohlásit že podporuje určitý DOM level

****

**Kontext a rozsah platnosti proměnné:**

- v JS existují tři druhy proměnných které se liší oborem platnosti (scope):

* Globální proměnné
* Lokální proměnné
* Proměnné s působností v rozsahu bloku (block-scoped)

- globální i lokální proměnné se deklarují klíčovým slovem **var**, obvykle se proměnné rovnou přiřadí nějaká hodnota jinak je výchozí hodnotou **undefined**



- pokud je proměnná deklarována mimo funkci jedná se o globální proměnnou která platí v celém skriptu, je-li deklarována uvnitř funkce jedná se o lokální proměnnou k níž existuje přístup pouze uvnitř této funkce nebo uvnitř funkcí deklarovaných v této funkci

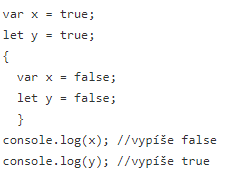
- globální i lokální proměnné mohou vznikat i bez explicitní deklarace

- kromě těchto proměnných se do JS později dostaly ještě proměnné a konstanty jejichž obor platnosti je v kódu vymezen složenými závorkami **{ }** a deklarují se klíčovými slovy **let a const**

- konstanty byly zavedeny kvůli bezpečnosti aby nemohly být přepsány cizím externím skriptem

- často bývá výhodné nebo dokonce potřebné aby proměnná platila pro danou funkci nebo jen nějaký cyklus či podmínku 🡪 kontext

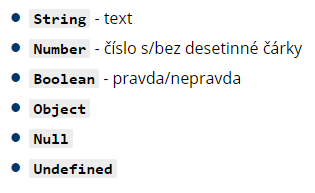
- takto je možné deklarovat množství stejnojmenných proměnných z nichž každá bude platná jen v daném kontextu



- při shodě jména globální a lokální proměnné dojde uvnitř funkce k uplatnění lokální proměnné a to i když je proměnná v algoritmu použita ještě před deklarací lokální proměnné

**Datové typy:**

- JS je dynamicky typovaný jazyk, plně nás odstiňuje od toho že proměnná má nějaký datový typ, základní datové typy jsou:





- u datového typu string se dá použít ještě několik speciálních funkcí:

