Válaszolj az alábbi kérdésekre minél részletesebben!

1. Mi a különbség a <span>, <div>, és az <article> elemek között?

**->**  span: inline container, nincs szemantikai jelentése, főként formázásra használják,  
 **->**  div: blokk szintű container, nincs szemantikai jelentése, főként formázásra használják,  
 **->**  article: blokk szintű container, szemantikai jelentéssel bír, önmagában is helytálló dokumentum részt tartalmaz

1. Milyen különbségeket és hasonlóságokat ismersz a cookie-k és a localStorage között?

**->** hasonlóságok: mindkettő a userre és/vagy a sessionre vonatkozó adatokat tartalmaz,

**->** különbség: localStorage-ban tárolt adat nem kerül visszaküldésre a szerverre, a kliens oldalon kerül felhasználásra, míg a cookie minden http request során elküldésre kerül (a headerben)

1. Milyen http metódusokat ismersz és melyiket milyen célra használják?

**->** get: adat lekérése, letöltésére való,  
 **->** post: adat szerverre történő feltöltésére való,  
 **->** put: ugyancsak adat feltöltésére és tárolására való,  
 **->**  delete: adat törlésére való,  
 **->** options: a szerver által támogatott metódusok lekérésére való

1. Mi lesz az ’x’ értéke és miért?

var a = b = x = 0;

x = ++a + b++;  
   
 **->** Az x értéke 1 lesz, mert az 'a' változó a művelet előtt kerül inkrementálásra, míg a 'b' csak utána.

1. Mi kerül az x változóba és miért?

var x = (a = 3);

**->** Az 'x' változóba 3 kerül, 'a' változó értékadása közben a visszatérési érték 3, így az kerül az x változóba is.

1. Mit ír ki és miért?

var bar = function() { console.log("bar"); }

var foo = function() { console.log("foo"); }

console.log("baz");

setTimeout(foo, 1000);

setTimeout(bar, 500);

// baz **->** első log

// bar **->** a foo és a bar köz egy időben indulnak el, mivel a bar 500 ms időeltolással logol, ez lesz a 2.

// foo **->** a foo 1000 ms (1 mp) időeltolással logol, ez a 3.

1. Mi a különbség a két változó között?

var v1 = document.getElementById(’inputText’);

var v2 = $(’#inputText’);

**->** v1 alapértelmezett js selector, az id-val rendelkező elemre mutató referenciával tér vissza,

**->** v2 jquery selector,  jQuery objektummal tér vissza

1. Mely mód(ok)on lehet elérni a „John Smith” szöveget tartalmazó változót?

var data = {

’person.name’: ’John Smith’

};

**->** data['person.name'];

**->** Object.values(data)[0];

1. Mit ír ki és miért?

var callbacks = [];

for (var i = 0; i < 5; ++i) {

callbacks.push(function() { console.log(i); });

}

callbacks[3]();

**->** Meghívásra kerül a callbacksben tárolt függvény, de mivel annak az 'i' változó referenciája lett átadva (ami a lefutás után már 5), így 5-t ír ki.

1. Mit ír ki és miért?

var a = 2, b = { c: 4 };  
var x = a, y = b;

x = 5;  
y.c = 6;

console.log(a + b.c);

**->** Az 'y' változónak referenciaként meg lett adva a 'b' objektum, így mikor felül lett definiálva az 'y' objektum 'c' tulajdonsága, a 'b' objektum 'c' tulajdonsága is megváltozott (az 'y' és a 'b' ugyanazt a referenciát tartalmazza), így 2 + 6 = 8-at ír ki.

1. Mi a különbség a két CSS selector között?

.element .symbol {}

**->** az element osztályú elemek összes child elemei között kiválasztja a symbol osztályú elemeket.

.element.large .symbol {}

**->** az element osztályú elemek közül a large osztályú elemek (mindkét osztályt tartalmazó elemek) összes child elemei között kiválasztja a symbol osztályú elemeket.

1. Milyen színű lesz a felirat és miért?

<div class="form-square">

<div class="seven-col">

Hello World!

</div>

</div>

div.form-square > div {

color: blue;

}

.seven-col {

color: red;

}

**->** CSS specificity miatt **kék** lesz: az első esetben 2 tag selector-t és 1 class selector-t tartalmaz a CSS selector, míg a második esetben csak egy class selectort, így az első nagyobb súlyú és felülírja a másodikat.

1. Adatbázis tervezési feladat

Tervezd meg egy webáruház adatbázis sémáját – relációs adatbázisban (táblákban gondolkodj)!

Termékeket szeretnénk tárolni úgy, hogy az adminok a felületen vehessék fel a termékeket. A termékhez bármennyi, különböző típusú attribútum tartozhat. Új terméktípus, vagy új termék-attribútum felvételéhez ne kelljen sémát módosítani!

Példa a termékekre és attribútumaikra:

Dell Vostro X500  
- memória: 16gb   
- processzor magok száma: 4   
- bevezetés ideje: 2015-12-31

Braun A1800 hajszárító- szín: fekete, fehér  
- erősség: 1800W  
- hideglevegő fújás: igen   
- garanciális hónapok száma: 12

A cél tehát olyan adatstruktúrát kialakítani, hogy ha ezentúl notebookoknál a kijelző felbontását is tárolni szeretnénk, akkor ehhez ne kelljen az adattáblák struktúráját módosítani. Ha több alternatív megoldást is találsz, írhatsz többet is. Az adatstruktúrát bármilyen formában leírhatod – csak értsük meg ☺

**->** Törekednék a minél jobban normalizál adattáblákra, csak az adott termékekre jellemző tulajdonságokat hagynám a táblában, pl: ha csak notebookokat árul a webshop, és nincsenek kategóriák, akkor benne hagynám a memóriát a notebook adattáblában, viszont ha vannak kategóriák (pl: subnotebook, asztali PC stb) akkor kialakítanék egy memória adattáblát is.

Példaadatáblák az adott példatermékek alapján:

Notebookok:

- id - int (primary key)

- típus - varchar,

- processzor - varchar *(nem raknám külön táblába, mert nagyon különböznek),*

- bevezetés - dátum,

- garancia - varchar/int

Memória:

- id - int (primary key)

- típus - varchar,

- kapacitás - int

- termék - termék\_id (foreign key), *//1 termék több típusú memóriával is elérhető lehet*

Szín:

- id - int (primary key)

- szín - varchar,

- termék - termék\_id (foreign key), *//1 terméknek több elérhető színe lehet*

Hajszárítók:

- id - int (primary key)

- típus - varchar,

- erősség - int,

- bevezetés - dátum,

- garancia - varchar/int,

- hideglevegő fújás: bool

*// kicsit tartok tőle, hogy félreértettem a feladatot, de ilyesmire gondoltam most így elsőre* ☺