Válaszolj az alábbi kérdésekre minél részletesebben!

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

<span> - általános inline tároló elem. Általában, akkor használjuk, mikor egy szövegben külön akarunk egy résznek style-t megadni (pl. félkövér, dőltbetű) vagy hasonló attribútum értékük van (pl. lang). Akkor érdemes használni, ha már más szemantikai elemet már nem tudunk.

<div> - általános tároló elem. Blokk szintű. Összefoghatunk vele több inline vagy más blokk elemeket, hasonlóan formázás céljából vagy pl. egy bekezdést, egy dokumentum bizonyos részét elérhetővé tegyük a Javascript számára. Szemantikailag ez is inkább semleges.

<article> - Egy önálló, szemantikailag összefüggő részét tartalmazza a dokumentumnak. Lehet egy cikk, blogbejegyzés, forum poszt stb.

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

A cookie-kat szerver oldalról olvassuk, míg a localStorage-ot főleg kliens oldalról. A localStorage automatikusan nem küldődik el a szervernek minden HTTP requestnél és a szerver sem tudja direkt olvasni. A cookie-knak van lejárati idejük (expiration date), a localStorage-nak nincs.

Méretük:

Cookie: 4,096 byte, 50 db/domain, összesen 3000db.

LocalStorage: nagyobb méret pl. 5Mb a Google Chrome estében.

A projektfeladatunk során (Zerda Reader Frontend) mi a localStorage-ot használtuk a felhasználót azonosító token tárolására, amit belépéskor kap. Kilépéskor töröljük a localStorage-ot:

<https://github.com/greenfox-zerda-lasers/zerda-reader-frontend/blob/master/app/login/login.controller.js>

<https://github.com/greenfox-zerda-lasers/zerda-reader-frontend/blob/master/app/navbar/navbar.controller.js>

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

GET – adatot kérünk a szervertől.

POST – adatot küldünk a szervernek.

PUT – adatot küldünk, amivel a backenden módosítani tudunk egy korábbit.

DELETE – adat törlésére használjuk a szerveren.

Több HTTP request is van, de mi ezeket használtuk a tanulmányaink során.

Itt van egy példa, amikor egy Todo appnál használtam:

<https://github.com/greenfox-zerda-lasers/gygabor/blob/master/week-09/day-3/mysqlproba/todo/todo.js>

További HTTP metódusok, amikről hallottam: OPTIONS, HEAD, CONNECT.

1. Mi lesz az ’x’ értéke és miért?

var a = b = x = 0;

x = ++a + b++;

x = 1, mert postfix inkrementálásnál (b++), előbb az eredeti értéket adja hozzá a többi értékhez (jelen esetben ++a-hoz, ami ekkor már 1), majd utána inkrementálja a b-t. A prefix inkrementálás (++a) előbb inkrementál, majd az új értéket adja át.

1. Mi kerül az x változóba és miért?

var x = (a = 3);

x = 3. Előbb az a-nak adok értéket a 3-at, majd az a értékét átadjuk x-nek.

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’

’bar’

’foo’

A setTimeout miatt aszinkronitás lép fel. A program lefut kiíja a ’baz’-t, majd meghívja a foo függvényt, 1 másodperc és a bar függvényt 0.5 másodperc „késleltetéssel”. Emiatt, előbb fog lefutni a bar függvény.

1. Mi a különbség a két változó között?

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

var v2 = $(’#inputText’);

Mindkettő az ’inputText’ id-jű DOM elemet menti el változóba. A getElementById csak a globális document objektumra használható.

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

var data = {

’person.name’: ’John Smith’

};

console.log(data['person.name']);

Object.keys(data).map(function(objectKey) {

console.log(data[objectKey]);

});

1. Mit ír ki és miért?

var callbacks = [];

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

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

}

callbacks[3]();

5;

A for ciklus öt függvényt hoz létre a callback array-en belül, melyek kiírják az i-t. Ami jelen esetben 5, mivel mikor meghívjuk a callback 3. függvényét (callbacks[3]();) a for ciklus lefutott és az i értéke 5. A prefix inkrementálás miatt nem 4 az i értéke, hanem 5.

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);

8

Mert az y = b-vel nem értéket adunk át, hanem egyenlővé tesszük a b objektummal az y-ont, így mikor megváltoztatjuk y értéké, b-é is változik.

1. Mi a különbség a két CSS selector között?

.element .symbol {}

a symbol class nevű elem az element class nevű elem alá tartozik:

<div class="element">

<div class="symbol”>

</div>

</div>

.element.large .symbol {}

a symbol class nevű elem az element class nevű elem alá tartozik, amelynek van még egy large classneve is:

<div class="element large">

<div class="symbol”>

</div>

</div>

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;

}

Kék. A ’ Hello World!’ div elem a form-square class nevű elem child-ja. Konkrétan meg van adva, hogy a form-square diven belül található divek kékek, míg a második esetben általánosan van meghatározva a ’seven-col’ class nevű elemek színe, így az első az specifikusabb. Ha át akarnánk színezni, így kéne megadnunk:

div.form-square > .seven-col {

color: red;

}

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 ☺

