**Final exam requirements**

The student, on their own, is able to:

1. Follow indentation and coding styleguides in their code
2. Use and set up linters (sima dependencia: a program futásához nélkülözhetetlen; dev-dependencia (develop): egyéb kiegészítő cuccok a programhoz. Enélkül is lefut. Ilyen az eslint is.)

kell előtte egy npm init. (package json miatt)

npm install eslint --save-dev vagy globalisan teszem fel: npm install -g eslint

eslint --init-et futtatni majd elvegezni a kovetkezo beallitasokat:

How would you like to configure ESLint? (Use arrow keys)

> Answer questions about your style

Use a popular style guide

> Google

Airbnb

Standard

Do you use React? No

What format do you want your config file to be in? (Use arrow keys)

JavaScript

YAML

JSON

1. Create variables, functions, objects in JavaScript based on a given specification

var,

let – csak a cikluson, belül érhető el az általa meghatározott változó,

const – ha változó, nem lehet módosítani az értékén. De ha objektumként határozok meg így valamit, akkor annak egyes értékeit változtathatom.

lehet a változó : integer, boolean, string, objektum, tömb

function - ismétlődő műveletek meghatározására.

function name(valami){

ide jön a valamigenerator

return

}

name()

Objektum: <http://gabor.molnar.es/blog/2013/05/05/javascript-programozas-jegyzet-objektumok/>

var students = {name: 'Rezso', age: 9.5, candies: 2},

1. Use conditions (if, switch) and loops (for, while):
   1. Filtering – tömbből ki kell valamit választani
   2. Finding elements or index
   3. Generating collections a collections egy array egy tombbol kivenni nehany elemet es uj tombot letrehozni
   4. Counting items in an object kaloz, cukor gyerek pelda
2. Determine the output of a logical expression (if, or, and) Határozd meg a kimenetét

true vagy false lesz az eredmeny

logical expression: and, or, not = !,

For example,

var status = (age >= 18) ? 'adult' : 'minor';

This statement assigns the value "adult" to the variable status if age is eighteen or more. És ha false akkor minor.

1. Determine the output of an immediately invoked function expression – meghatározni egy function kimenetét – revealing module pattern resz(functionban function)

cloesure!!

<https://addyosmani.com/resources/essentialjsdesignpatterns/book/#revealingmodulepatternjavascript>

1. Use higher order functions

lehetnek: split,

map – itt barmit csinalhatunk a tomb elemeivel, akar meg is valtoztathatjuk es igy csinalunk egy uj tombot;

filter – egy feltetlet vizsgalva (ez iga, vagy hamis) csinal egy uj tombot az igaz ertekekkel,

every – az allitas minden elemre igaz kell, hogy legyen – false vagy true erteket ad vissza

some-– az allitas csak nehany elemre kell, hogy igaz legyen – false vagy true erteket ad vissza

reduce-nem kell foreach, - 7. hét 3.nap

1. Explain their own code
2. Initialize a node package

npm init –> json file lesz

.gitignore file keszitese. beleirni: node\_modules

dev dependencia – nem feltetlenül kell a programod futásához. Pl tesztelés

tesztelés futtatása

1. Install packages via a package manager

npm install valami –save

Now install Express in the myapp directory and save it in the dependencies list. For example:

$ npm install express --save

To install Express temporarily and not add it to the dependencies list, omit the --save option:

$ npm install express

1. Write unit tests for functions and methods without mocking dependencies

sima teszteket kell mutatni – TESZTET KERESNI!! 8.hét 4.nap első 3 teszt

npm install tape --save-dev

tesztelendo file futtatasa: node 39\_test.js

1. Include and call MI EZ??? 3rd party libraries in their code (külső)

HOL ES MIERT EZT HASZNALOM??

script meghívása html tetején vagy require tape

library = előre definiált functionöket tartalmaz collectoins of functions and objects

Bootstrap, jquery, semantic UI, react, express, jasmin, tape, stb.

A nodejs nem library!

* 1. Using libraries on the frontend

bootstrap, jquery, semantic UI,

npm install semantic-ui --save

semantic.json-t hoz letre

// Import Semantic UI modules (styles):

require('./node\_modules/semantic-ui/dist/semantic.min.css');

require('./node\_modules/semantic-ui/dist/components/accordion.css');

require('./node\_modules/semantic-ui/dist/components/popup.css');

* 1. Using libraries on the backend

express,

Now install Express in the myapp directory and save it in the dependencies list. For example:

$ npm install express --save

To install Express temporarily and not add it to the dependencies list, omit the --save option:

$ npm install express

var express = require('express');

var app = express();

app.use(bodyParser.json());

app.use(express.static('client'));

app.post('/exam/', function(req, res) {

1. Use frameworks and know its components and processes

framework = a frémwörk futtatja az én kódomat, struktúrát ad neki, collections of libraries and patterns

Angular

components: controller, factory, directive,

router, stb hasznalata

process: data binding – {{ez is az}}

1. Use async structures:

AJAx hivas, eventek, settimeout, setinterval – bizonyos időnként újra lefuttatja önmagát

* 1. Writing functions that are taking and calling callbacks
  2. Determining the calling order in nested callbacks
  3. Using timers

1. Use git:
   1. Commit, add, push, pull
   2. Resolving a simple merge conflict – HEAD az ujabb cuccok kerulnek a headbe
   3. Creating a pull request – egy új branch pullolása, összemergelése a master ággal
2. Use the command line:
   1. Traversing directories – mappaváltás - cd
   2. Moving mv – mit hova

cp -r mit hova másolás –r rekurzió: mappa és tartalma, file r nélkül is mehet

creating, fájlt: touch, könyvtárat: mkdir

deleting files = remove rm

* 1. Passing command line arguments to commands pl.: ls –l; ls-alt; npm init valami –save, ezek a kotojeles cuccok

1. Use databases:
   1. Implement database calls with JavaScript – SELECT FROM, INSERT INTO
   2. Write a simple SQL query for a single table:
      1. Listing – ez a select, Deleting, Updating, Creating rows with specific values Tibi TODO - <https://github.com/greenfox-zerda-lasers/syllabus/tree/master/solutions-reviews/todo-app>
2. Write backend code:
   1. Implement an HTTP endpoint – server.js –ben
   2. Set headers on HTTP responses – json cucc
3. Explain REST concepts – representational state transfer – a wikin

Kommunikacios architektura. REST api

<http://rest.elkstein.org/2008/02/what-is-rest.html>

A **REST** (**Representational State Transfer**) egy szoftverarchitektúra típus, [elosztott](https://hu.wikipedia.org/wiki/Elosztott_sz%C3%A1m%C3%ADt%C3%A1sok) kapcsolat (loose coupling), nagy internet alapú rendszerek számára, amilyen például a [világháló](https://hu.wikipedia.org/wiki/Vil%C3%A1gh%C3%A1l%C3%B3).

## **Koncepció[**[**szerkesztés**](https://hu.wikipedia.org/w/index.php?title=REST&action=edit&section=2)**]**

Egy REST típusú architektúra kliensekből és szerverekből áll. A kliensek kéréseket indítanak a szerverek felé; a szerverek kéréseket dolgoznak fel és a megfelelő választ küldik vissza. A kérések és a válaszok erőforrás-reprezentációk szállítása köré épülnek. Az erőforrás lényegében bármilyen koherens és értelmesen címezhető koncepció lehet. Egy erőforrás-reprezentáció általában egy dokumentum, mely rögzíti az erőforrás jelenlegi vagy kívánt állapotát.

Bármely adott pillanatban egy kliens vagy állapotok közötti átmenetben van, vagy "nyugalmi" állapotban. A nyugalmi állapotban lévő kliens képes interakcióra a felhasználójával, de nem hoz létre terhelést és nem fogyaszt tárolót a szervereken vagy a hálózaton.

Ha a kliens készen áll az átmenetre egy új állapotba, akkor elkezdi küldeni a kéréseit a szerverekhez. Míg legalább egy olyan kérés van, amelyre nem érkezett válasz, a kliens átmeneti állapotban marad. Egyes erőforrás-reprezentációk hivatkozásokat tartalmaznak további erőforrásokra, amelyeket a kliens felhasználhat új állapotba történő átmenetkor.[[7]](https://hu.wikipedia.org/wiki/REST#cite_note-7)

A REST eredetileg a [HTTP](https://hu.wikipedia.org/wiki/HTTP) keretein belül lett leírva, de nem korlátozódik erre a protokollra. Egy "RESTful" architektúra más [alkalmazási rétegbeli](https://hu.wikipedia.org/wiki/Alkalmaz%C3%A1si_r%C3%A9teg) protokollra is épülhet, amennyiben az már rendelkezik értelmes erőforrás-reprezentáció átvitelhez szükséges gazdag és egységes szókinccsel. A "RESTful" alkalmazások maximálisan kihasználják a választott hálózati protokoll már létező, jól kialakított interfészeit és egyéb beépített képességeit, és minimalizálják új alkalmazás-specifikus jellemzők bevezetését.

### A HTTP példája**[[szerkesztés](https://hu.wikipedia.org/w/index.php?title=REST&action=edit&section=3" \o "Szakasz szerkesztése: A HTTP példája)]**

A HTTP nagyon gazdag szókinccsel rendelkezik igék (vagy "metódusok"), [URI](https://hu.wikipedia.org/wiki/URI" \o "URI)-k, média típusok, kérés- és feleletkódok stb. szempontjából. Egy REST alkalmazás a HTTP protokoll meglévő tulajdonságait használja és így lehetővé teszi a proxyknak és az átjáróknak, hogy együttműködjenek az alkalmazással (például gyorsítótárazás vagy biztonsági funkciók formájában).

1. Write frontend code:
   1. Create layout in CSS and HTML based on a given specification
   2. Select any HTML element using CSS selectors
   3. Handle events- pl click
   4. Dynamically create elements – létrehozni html cuccot js-en keresztül- képnézegető képei
   5. Change attributes and properties on HTML elements using JavaScript

document.getElementsByTagName("H1")[0].setAttribute("class", "democlass"); innertext, innerhtml

* 1. Initiate HTTP requests XML http request

1. Use build tools – pl webpack
2. Deploy application to hosting service – Heroku, github pages is jo, vagy gomix

Github pages – settings – source

Alapbol csak a mastert, vagy ezen belule egy /docs mappat lehetbeallitani, de ha egy branchet gh-pages-nek hivok, akkor az is valaszthato lesz.

<https://tpracser.github.io/fedex-helga-pracsi-bence/>

1. Differentiate between backend and frontend functionality
2. Refactor code:
   1. Create meaningful names for functions and variables
   2. Split bigger functions to smaller ones
   3. Separate functions to different files
   4. Point out code smells –ezek a total rossz kodok, pl kiscica nevu, 200 soros kod, duplikacio, stb.
3. Name 3 design patterns

[JavaScript Design Patterns](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#designpatternsjavascript)

* [Constructor Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#constructorpatternjavascript)
* [Module Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#modulepatternjavascript)
* [Revealing Module Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#revealingmodulepatternjavascript)
* [Singleton Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#singletonpatternjavascript)
* [Observer Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#observerpatternjavascript)
* [Mediator Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#mediatorpatternjavascript)
* [Prototype Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#prototypepatternjavascript)
* [Command Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#commandpatternjavascript)
* [Facade Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#facadepatternjavascript)
* [Factory Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#factorypatternjavascript)
* [Mixin Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#mixinpatternjavascript)
* [Decorator Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#decoratorpatternjavascript)
* [Flyweight Pattern](https://addyosmani.com/resources/essentialjsdesignpatterns/book/#detailflyweight)

1. Understand how the internet works
   1. Name the parts of an URL

**URL (Uniform Resource Locator)**: {magyar fordításban: egységes (információ)forrás (meg)határozó} Szabványos jelölésrendszer különböző típusú (Web, gopher, newsgroup, FTP stb.) Internet információforrások helyének meghatározására. Az ~ címek általában a forrás típusát, a szolgáltató gép vagy a szolgáltatás nevét, és esetleg egy állomány vagy alkönyvtár elérési útvonalát tartalmazzák.

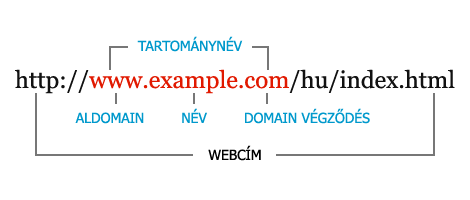
Az **URL** vagy más néven **webcím** (mely a ***U****niform****R****esource****L****ocator* [egységes erőforrás-azonosító] rövidítése), az [interneten](https://hu.wikipedia.org/wiki/Internet) megtalálható bizonyos erőforrások (például szövegek, képek) szabványosított címe.

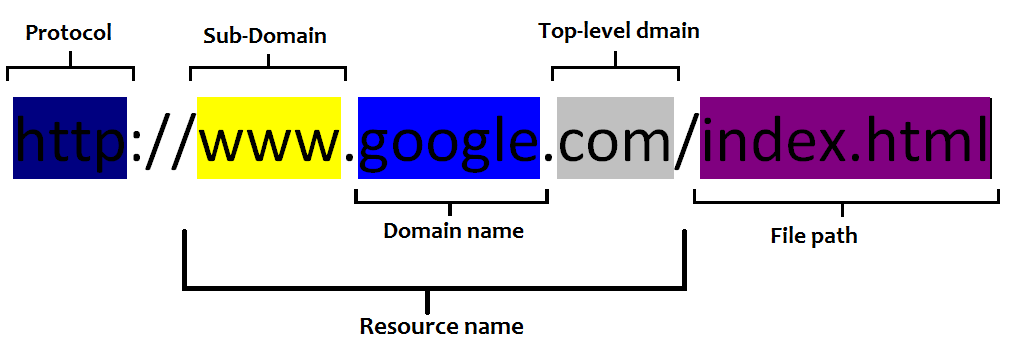
## URL részei

[**Az URL három részből áll**](https://hu.wikipedia.org/wiki/URL):

1. **Protokoll**, ami megmondja hogyan csatlakozzunk a weboldalhoz. Ez normál esetben http (szabad szöveg) vagy https (titkosított kapcsolat).
2. [**Domain név**](http://profiseo.hu/domain-nev-jelentese/), amihez csatlakozni szeretnénk.
3. **Konkrét tartalom**







Egy tipikus, egyszerű webcím így néz ki:

<https://hu.wikipedia.org:80/wiki>

Ennek részei:

* A *http* a használandó **[protokoll](https://hu.wikipedia.org/wiki/Protokoll_(informatika)" \o "Protokoll (informatika))**. A protokoll neve után kettőspont (:) írandó.
* A *hu.wikipedia.org* a célgép **[tartományneve](https://hu.wikipedia.org/wiki/Tartom%C3%A1nyn%C3%A9v" \o "Tartománynév)**. Ez elé két perjel (//) írandó.
* A *80* a célgép azon hálózati **[portszáma](https://hu.wikipedia.org/w/index.php?title=Portsz%C3%A1m&action=edit&redlink=1" \o "Portszám (a lap nem létezik))**, amin kérésünket várja; ez elé kettőspont (:) írandó. Ezt a részt gyakran teljesen elhagyhatjuk, például esetünkben a [http](https://hu.wikipedia.org/wiki/Http) protokoll alapértelmezett portszáma a 80.
* A */wiki* a kért **[elérési út](https://hu.wikipedia.org/w/index.php?title=El%C3%A9r%C3%A9si_%C3%BAt&action=edit&redlink=1" \o "Elérési út (a lap nem létezik))** a célgépen. Ez a rész mindig a perjellel (/) kezdődik.

A legtöbb böngésző nem is igényli, hogy a „http://” részt begépeljük egy [weblap](https://hu.wikipedia.org/wiki/Weblap) eléréséhez, hiszen az esetek döntő többségében úgyis ezt használjuk. Egyszerűen begépelhetjük a lap címét, például: „hu.wikipedia.org/wiki/Bit”. A főlap megtekintéséhez általában elég a tartomány nevét beírni, például „hu.wikipedia.org”.

A webcímek egyéb részeket is tartalmazhatnak, [http](https://hu.wikipedia.org/wiki/Http) esetében például az elérési út után, egy kérdőjel (?) mögé helyezve keresési kérdés szerepelhet, ami egy *get* metódusú [HTML űrlapból](https://hu.wikipedia.org/w/index.php?title=HTML_%C5%B1rlap&action=edit&redlink=1) származik. Az elérési út után, attól egy kettős kereszttel (#) elválasztva szerepelhet a [hiperszöveg](https://hu.wikipedia.org/wiki/Hipersz%C3%B6veg" \o "Hiperszöveg) egy részére hivatkozó azonosító. Ez az azonosító nem része a webcímnek, de gyakran szerepel vele kapcsolatban.

* 1. Name a few protocols/technologies in connection with network communication

ip, dns, http, url reszei, .com,.hu – mi az, tc/ip,

Az **IP-cím** **(Internet Protocol-cím)** egy egyedi [hálózati azonosító](https://hu.wikipedia.org/w/index.php?title=H%C3%A1l%C3%B3zati_azonos%C3%ADt%C3%B3&action=edit&redlink=1), amelyet az [internetprotokoll](https://hu.wikipedia.org/wiki/Internetprotokoll) segítségével kommunikáló számítógépek egymás azonosítására használnak. Minden, az [internetre](https://hu.wikipedia.org/wiki/Internet) kapcsolt számítógépnek van IP-címe, de egy-egy konkrét cím nem kötődik feltétlenül egy-egy géphez: egyes gépeknek több címük is lehet (ilyenkor a különböző címek rendszerint a számítógép különböző hálózati eszközeit azonosítják), vagy több gép osztozhat egy címen (például [NAT](https://hu.wikipedia.org/wiki/H%C3%A1l%C3%B3zati_c%C3%ADmford%C3%ADt%C3%A1s) vagy [proxy](https://hu.wikipedia.org/wiki/Proxy_szerver) használata esetén), vagy a gép IP-címe rendszeresen változhat (ez különösen a lakossági [internetszolgáltatón](https://hu.wikipedia.org/wiki/Internetszolg%C3%A1ltat%C3%B3) keresztül kapcsolódó otthoni számítógépekre jellemző).

**IP (Internet Protocol)**: Egy internet hálózaton az adatcsomagok összeállításáért és továbbításáért felelős protokoll.

**IP address**: {magyarul: IP cím} Egy Internetbe kapcsolt számítógép (host) hálózati azonosító száma. Rendszerint 4 decimális számból áll, melyeket pontok választanak el.

A **Domain Name System** **(DNS)**, azaz a **tartománynévrendszer** egy hierarchikus, nagymértékben elosztott elnevezési rendszer számítógépek, szolgáltatások, illetve az [internetre](https://hu.wikipedia.org/wiki/Internet) vagy egy [magánhálózatra](https://hu.wikipedia.org/wiki/Mag%C3%A1nh%C3%A1l%C3%B3zat) kötött bármilyen erőforrás számára. A részt vevő entitások számára kiosztott [tartománynevekhez](https://hu.wikipedia.org/wiki/Tartom%C3%A1nyn%C3%A9v) (doménekhez) különböző információkat társít. Legfontosabb funkciójaként az emberek számára értelmes tartományneveket a hálózati eszközök számára érthető numerikus azonosítókká „fordítja le”, „oldja fel”, melyek segítségével ezeket az eszközöket meg lehet találni, meg lehet címezni a hálózaton.

Az internetes tartománynevek további célja az egyszerűsítés, egy doménnevet (pl. www.example.com) sokkal könnyebb megjegyezni, mint egy IP-címet, mint 208.77.188.166 (IPv4) vagy 2001:db8:1f70::999:de8:7648:6e8 (IPv6). A felhasználók így megjegyezhetik a számukra jelentést hordozó [web-](https://hu.wikipedia.org/wiki/Webc%C3%ADm) (URL) és [emailcímeket](https://hu.wikipedia.org/wiki/Email" \o "Email), anélkül, hogy tudnák, a számítógép valójában hogyan éri el ezeket.

## **DNS, mint Informatikai fogalom**

Azok a szerverek, amelyek számon tartják az egyes IP címekhez tartozó számítógépneveket, illetve biztosítják ezek oda-vissza fordítását, mert a gépek csak a numerikus IP címeket tudják kezelni, az emberek viszont könnyebben megjegyzik a neveket. A DNS lehet Domain Name System jelentésű is, ebben az értelemben a hálózatba kötött számítógépek azonosítóinak elosztott adatbázisa.

**DNS (Domain Name System)**: Tartománynév rendszer, tartományneveket IP címekké átalakító Internetes szolgáltatás.

**domain**: {magyarul: címtartomány} Az Internethez csatlakozó gépek hierarchikus címrendszerének valamelyik szintje. A legfelső szintű ~ gyakran az országot jelzi, az alatta levő pedig a gép tulajdonosát stb.

**DOM**: A dokumentumobjektum-modell egy API a HTML és az XML dokumentumokhoz. A dokumentum szerkezetét modellezi és lehetővé teszi annak tartalmi és vizuális változtatását. Lényegében összeköttetést képez a weblapok és a script vagy prog­ramo­zási nyelvek között.

**HTML (HyperText Markup Language)**: {magyar fordításban: hipertext jelölő nyelv} A World-Wide Web hipertext oldalainak leírására szolgáló szabványos nyelv, az SGML egyik egyszerűsített majd továbbfejlesztett változata. A ~ nyelven írt állományokat (jellemzően) HTTP szerverekkel szolgáltatják és Web-böngészőkkel jelenítik meg.

**HTTP (Hypertext Transfer Protocol)**: {magyar fordításban: hipertext átviteli protokoll} HTML és a World-Wide Webnél használt egyéb formátumú állományok továbbítására kidolgozott eljárás az Interneten. A Web szerverek és kliensek közötti kommunikációnál használt adatcsere szabvány.

**HTTPS**: (Hyper Text Transfer Protocol Secure) Szabványos internet adattovábbítási protokoll, az adatforgalom mindkét irányú titkosításával.

**TCP/IP (Transmission Control Protocol/Internet Protocol)**: {magyar fordításban: átvitelvezérlő protokoll/internet protokoll} Az Internet világhálózatnál (is) használt adatátviteli eljárásgyűjtemény. A hálózatba kötött gépek egymás közötti kommunikációjának szabályait írja le.