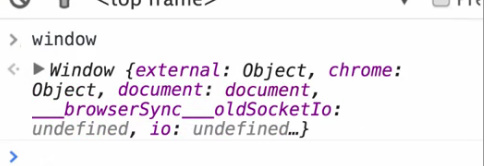
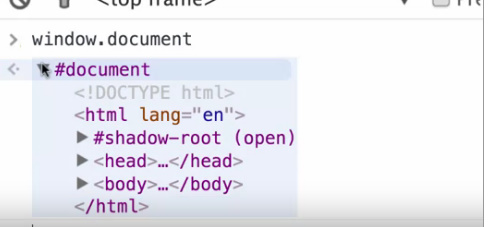
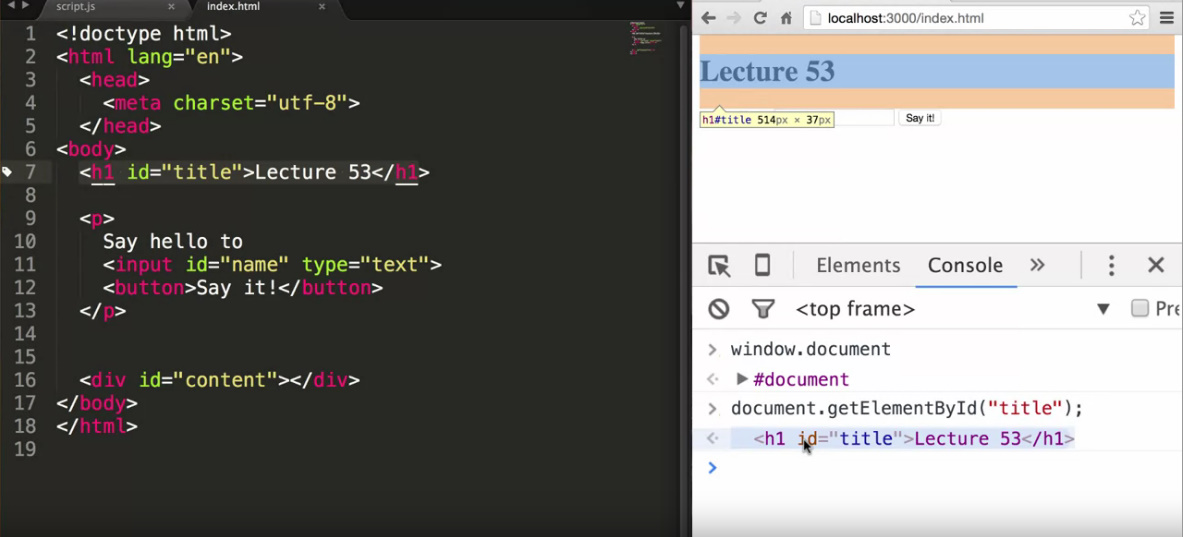
**DOM**

* **Document** – nume oferit intregii pagini HTML
* Document este un object din window object



****

* **deci rolul lui document object este de a ne oferi acces la elementele din pagina**
* document ne ofera posibilitatea de a obtine elementele dupa id-ul lor

****

* <script src=> </script> face ca codul din sursa sa fie copiat intre taguri
* nu e o idee buna de a pune codul js in <head>, deoarece la momentul executiei lui, adica in head, elementele paginii inca nu sunt generate,si daca acel js cod incearca sa obtina vreun elemente, nu are cum, caci nici un elemente in pagina nu e generat la executia scriptului
* Cel mai bine e de a pune scriptul la final de code
* Nu uitam ca HTML executa totul secvential
* document se afla si el in obiectul **HTMLDocument**

**Methods**

* **document.getElementById(...).value/textContent/innerHTML** – returneaza valoarea introdusa de user
* **document.getElementById(...).textContent = value –** insereaza o valoare in element, dar ca text, deci daca am avea o valoare ca <H2>Text</H2>, pai tagul va fi afisat asa cum il vedem noi
* **document.getElementById(...).innerHTML = value** insereazza valoarea ca cod **H**TML, deci putem folosi si tgauri
* **document.querySelector(selector)**.(**textContent,innerHTML)** – getElementById ne limiteaza la a selecta un element doar dupa id, dar querySelector ne permite sa folosim selectori din CSS, ca de ex .clasa sau div sau #id .Returneaza doar primul gastit

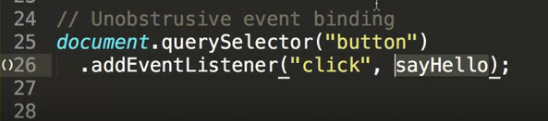
**Events**

* **EventHandlers –** sunt functii pe care le unim de metodde specifice, iar metodele sunt eventuri. De ex, avem onclick()
* **Metode de a administra evenimentele:**

1. **Folosind codul HTML**

* Events**:**
* onclick
* onblur – cand scriem de ex intr-un box text si iesim din el.
* ....

1. **Prin metoda addEventListener(„eveniment”,metoda)**



evenimente:

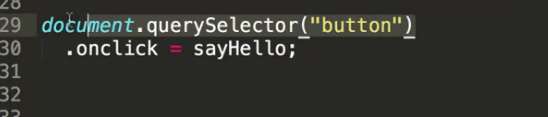
„click”

....

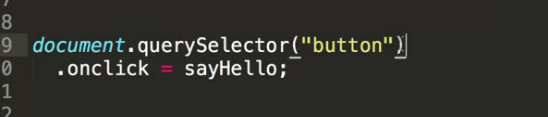
* asa nu mai trebuie sa folosim onclick si nici HTML, caci el nu e facut pentru asta ci pentru stilizare

1. **Prin proprietatile tuturor elementelor**

de ex document.querySelector(...).onclick = metoda



Obervam ca this din metoda ce administreaza evenimentul va fi acel element, ca button de ex, deoarece butonul e cel care creaza contextul si exeuta metoda:

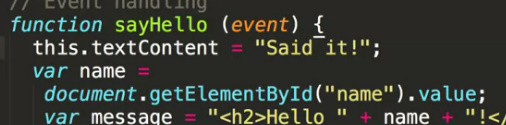






Avantajul la metodele 2 si 3 este:

1. ne ofera posibilitatea de a edita butoanele care au declansat eventul prin **this**:
2. Nu trebuie sa mai importam <script> la final de cod html si sa ne batem capul de ordinea lui, ci il putem chiar i <head>(citeste mai jos cum facem asta)
3. ascunde codul javascript mai bine



In poza de sus, cand butonul va fi apasat, el isi modifica textul

* E de dorit ca fiecare functie ce administeaza evenimente sa aiba un parametru event

**Problema cu continutul loaded la inceput**

* Nu e prea comod sa tot punem <script> la final de cod in pagina, dar daca il punem la inceput si setam evenimente la butoane, se intampla problema ca setam evenimente la elemente ce nu exista la momentul executarii scriptului, caci doar scriptul in < head> daca e pus se executa primul
* Pentru a rezolva problema asta, executam tot scriptul intr-o metoda ce se va asigura ca toate metodele noastre sa fie executate dupa ce intreaga pagina e incarcata

<script>

document.addEventListener("DOMContentLoaded",function() {

function hello(event) {

var name = document.getElementById("name").value;

document.querySelector("#hi").innerHTML = "<i>Salut " + name + "</i>";

}

document.getElementById("a").addEventListener("click", hello);

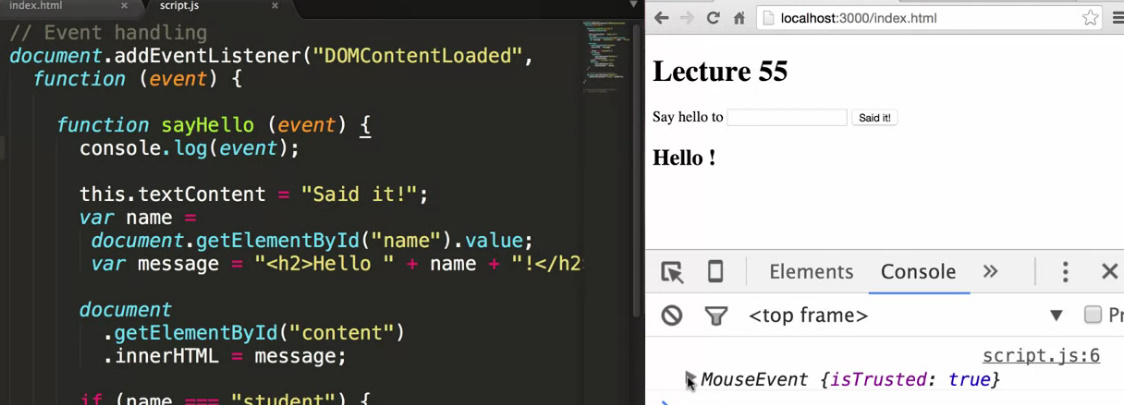
});

</script>

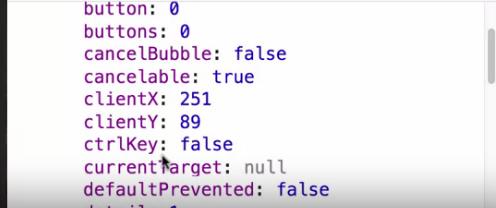
Mare grija la parametrul function () { }

**Event argument**

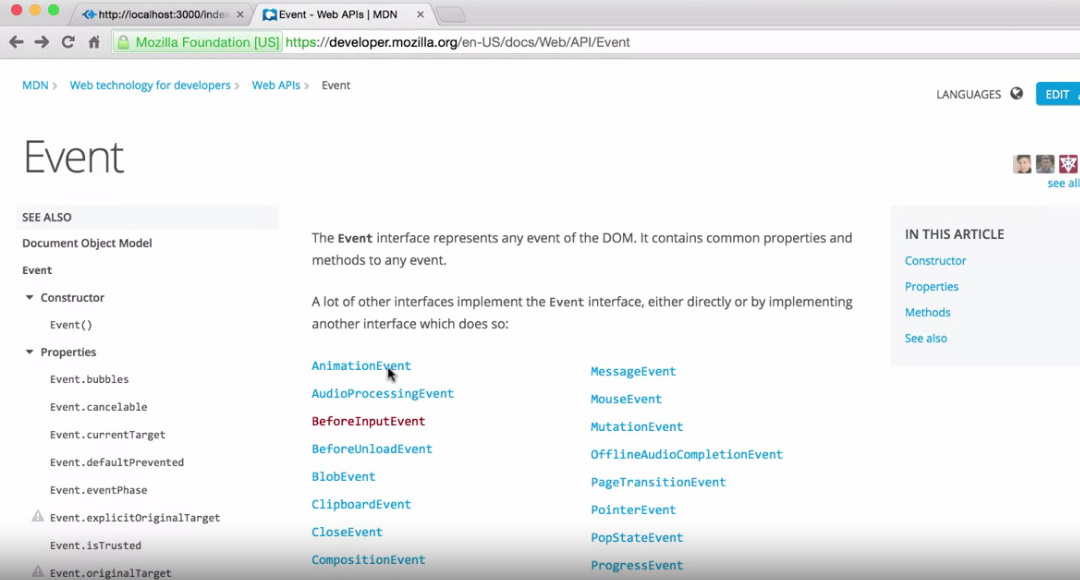
* Fiecare metoda ce gestioneaza un eveniment e recomandat sa aiba arg umentul event
* La fiecare declansare a unui eveniment, gen la apasarea unui buton, JS se va asigura sa trimita un obiect **event**



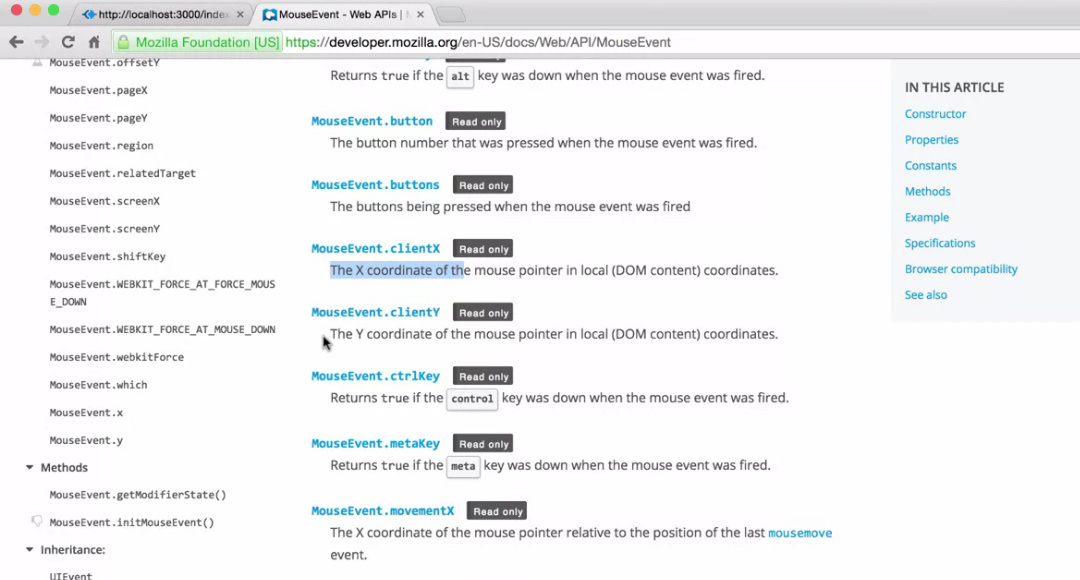
Iar acest MouseEvent contine multe date despre event:



* Pe langa MouseEvent, putem sa mai avem multe altele,ca KeyboardEvent care toate implementeaza interfata Event. Depinde cum il declansam



Fiecare din ele, dupa cum am vazut, are proprietati despre eveniment:

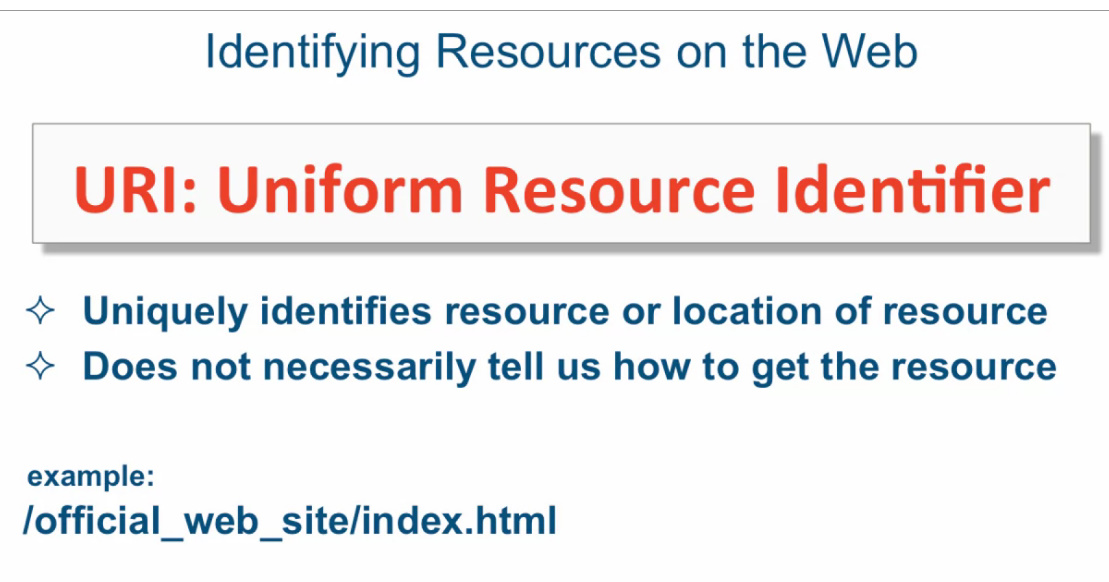


**HTTP**

* **HTTP –** protocol sau limbaj prin care clientul(browserul) comunica cu serverul
* **HTTP** nu memoreaza nici-un request sau response de dinainte.
* Clientul comunica cu serverul in urmatorul mod:

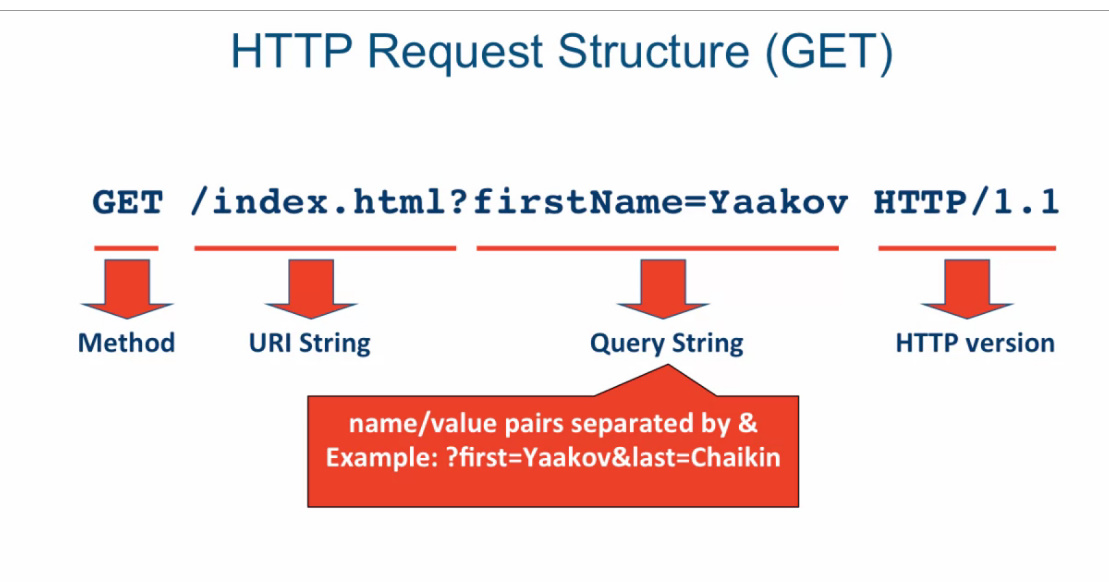
1. clientul stabileste o conexiune cu serverul
2. clientul trimite un **HTTP request** pentru o resursa
3. Serverul trimite un **HTTP response** la client
4. Clientul incide conexiunea cu serverul

* **URN –** Uniforme Resource Name
* **URN** se bazeaza pe faptul ca fiecare resursa se identifica printr-un nume/identificator propriu
* ****
* **URI –** Unifor Resource Identifier
* **URI –** se bazeaza pe numele resursei si pe locatia ei concreta

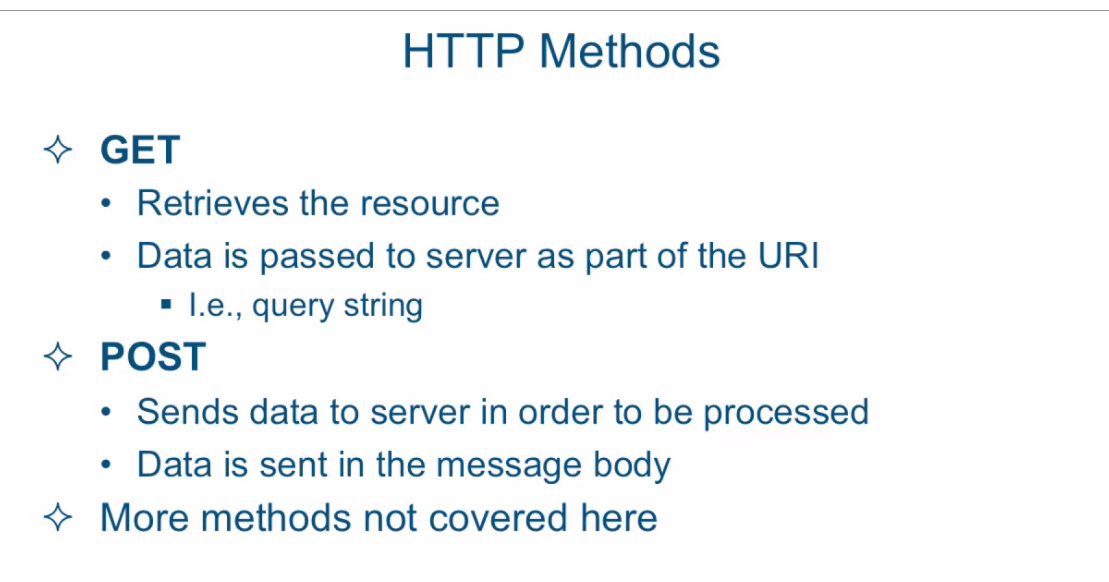
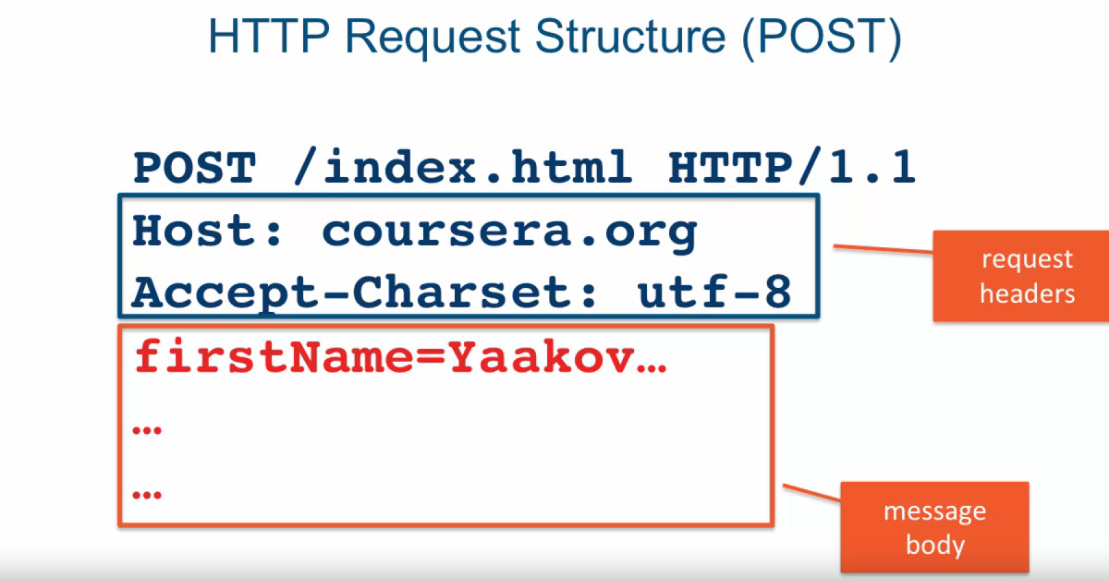
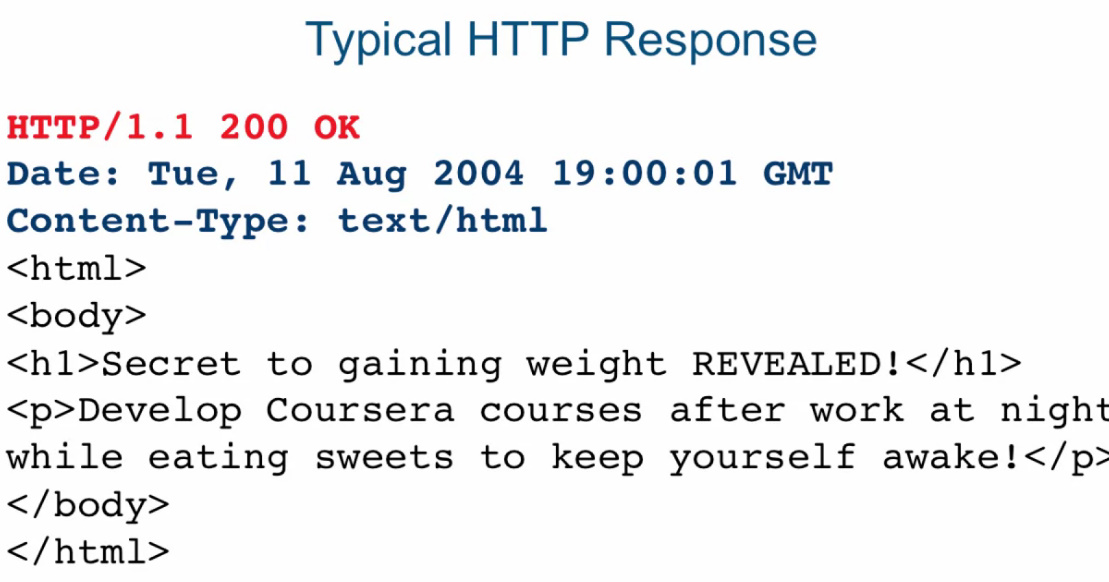
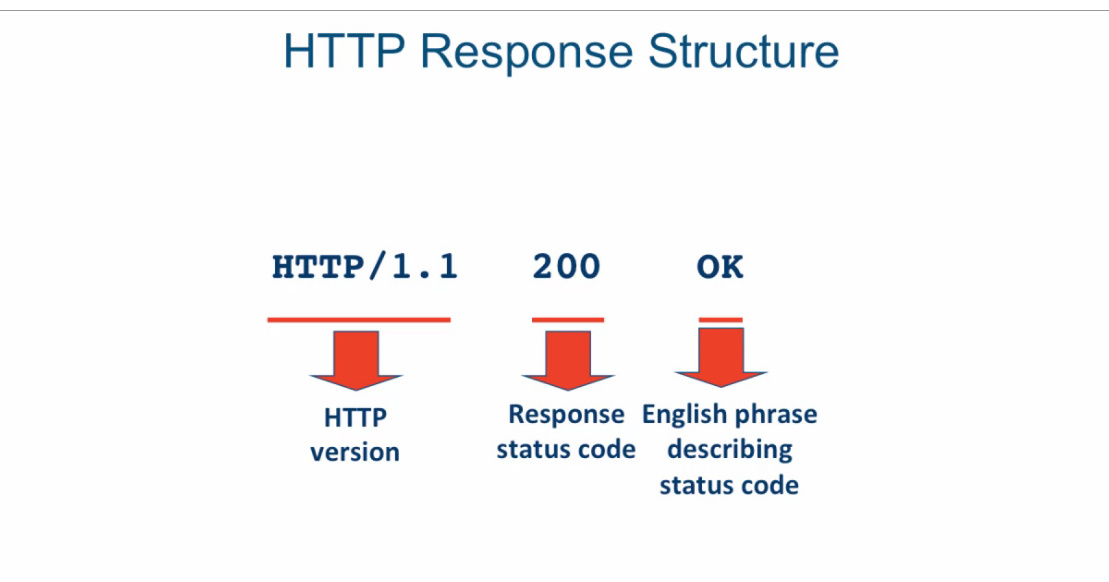
****

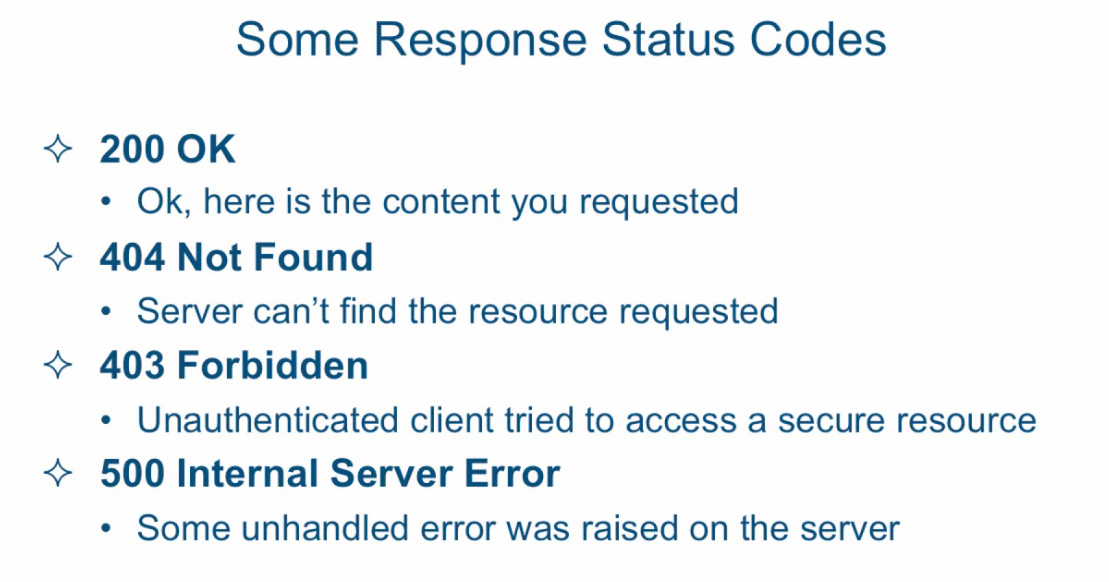
* **URL –** uniform resource locator
* **URL** se bazeaza pe adresa resursei , ca a siteului



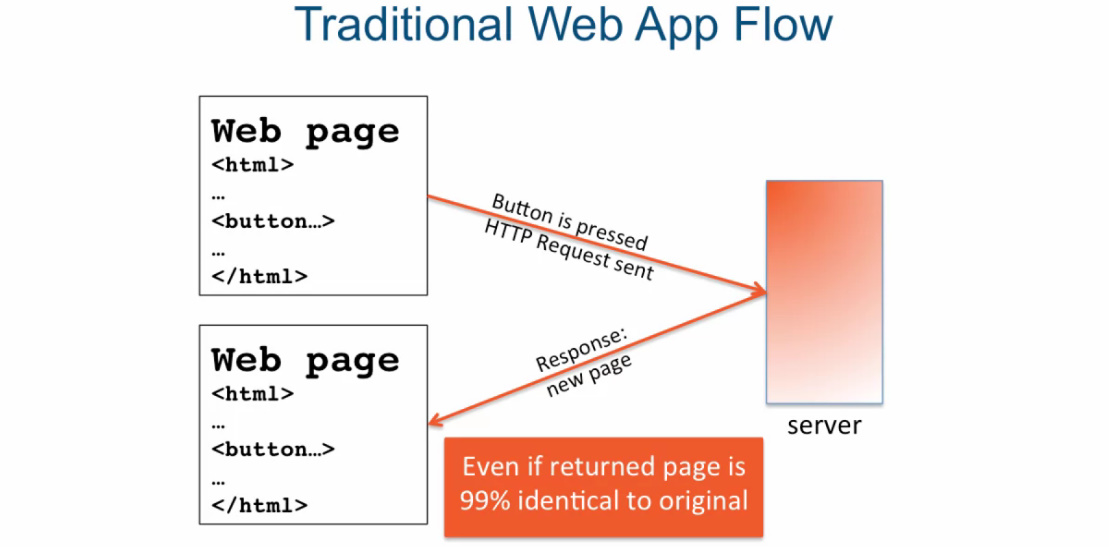
* ****

Asa trimite browserul un get request

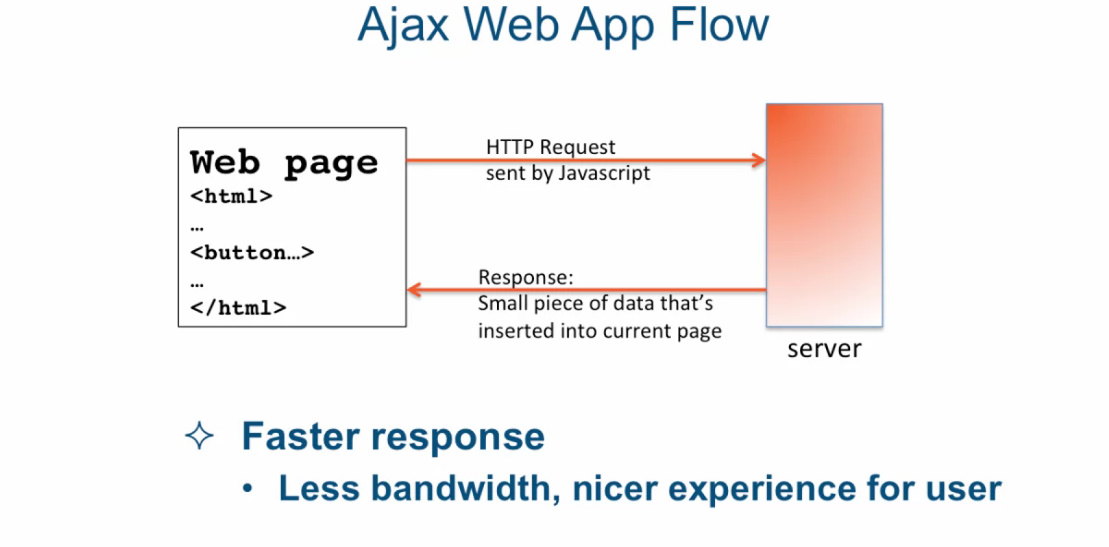
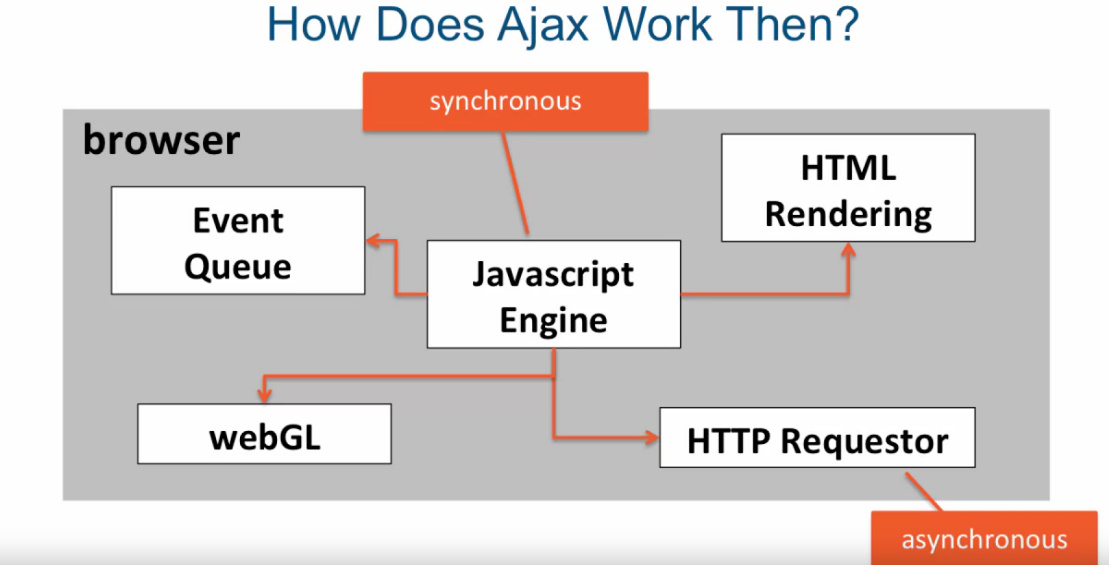
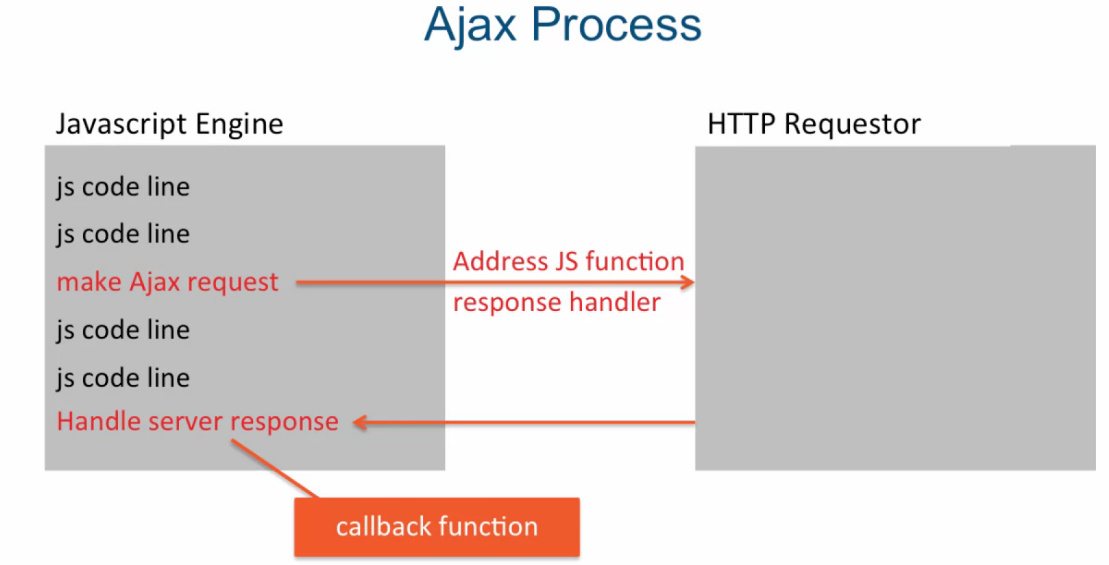
* 
* 
* deci la POST datele nu sunt trimise in URL, dar in message body
* 
* 



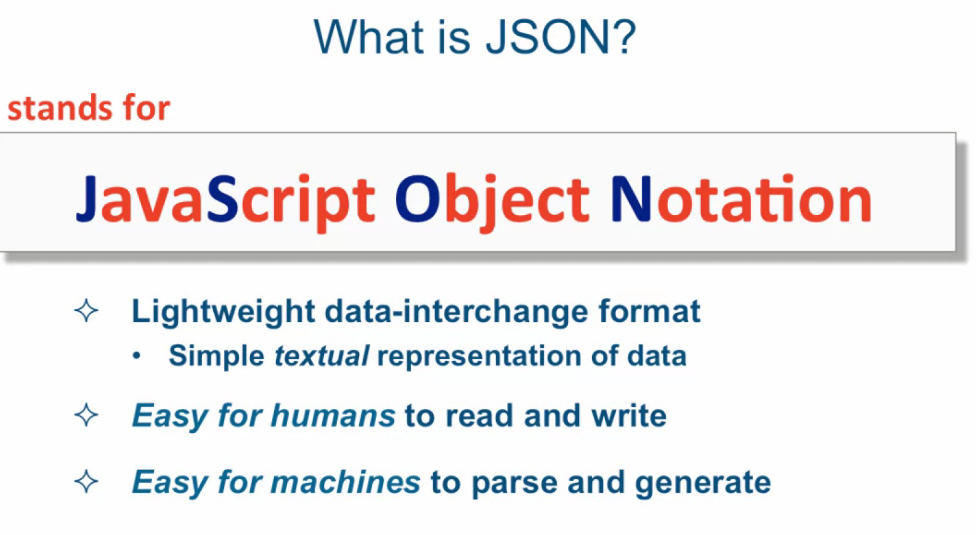
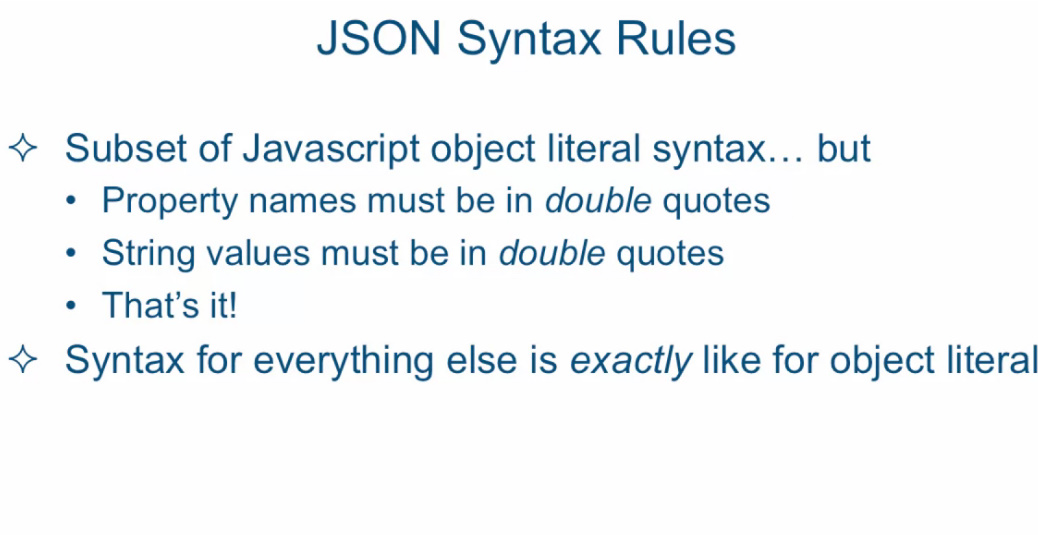
**AJAX**

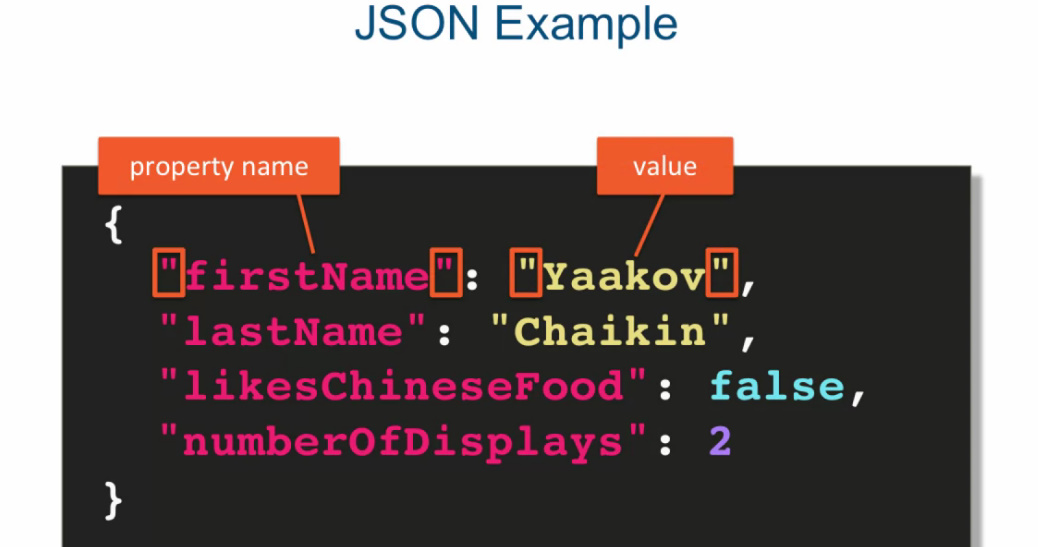
* **AJAX** – Asynchronous JavaScript And XML
* **Ajax** a inceput prin a folosi XML, dar putine aplicatii mai folosesc XML acum. In schimb, se foloseste mai mult HTML sau JSON
* ****

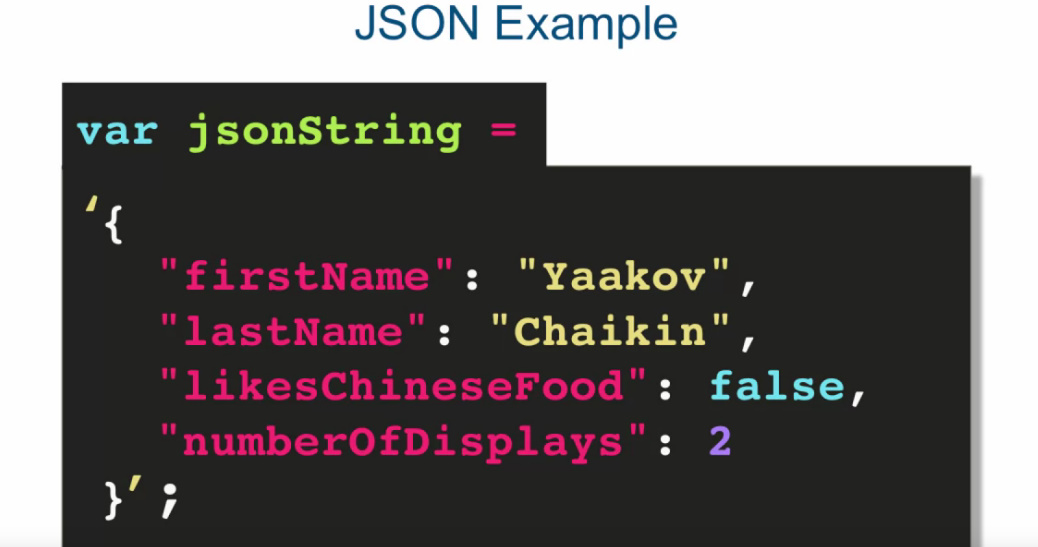
Deci, in aplicatiile traditionale, un request facut de vreun element din pagina, va duce la aceea ca se va trimite un request cu HTTP si serverul va returna aceeasi pagina, doar ca ceva modificat la ea, putin modificat deobicei

* 
* la Ajax request HTTP e facut de JS
* Ajax este folosit cand JS face requesturi
* 
* JS cand face un request, ea foloseste un obiect special pentru a se duce la HTTP Requester
* 

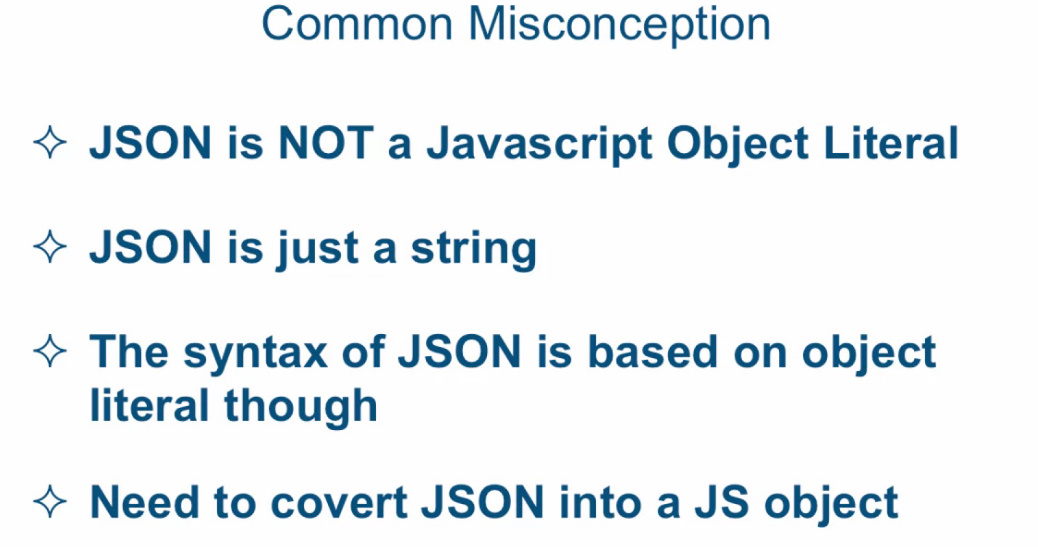
**JSON**

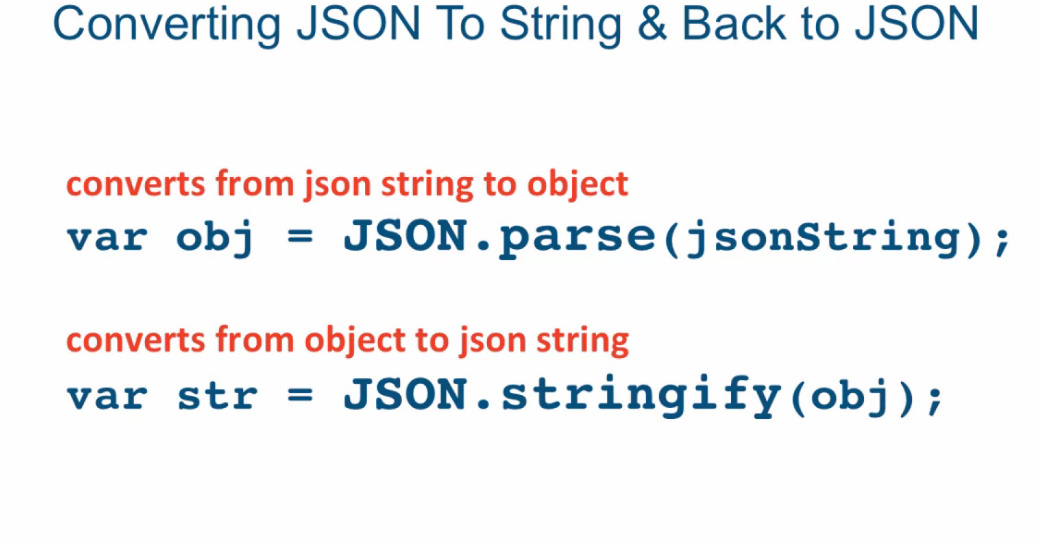
* 
* 





In JS JSON va fi un simplu String, nu un obiect!





* JSON este un format foarte bun pentru a transfera datele de la server la client si invers
* JSON nu e JS Literal!E un string asemanator doar
* ‚’ nu sunt permise in JSON!

**JQUERY**

* $(functie() {.... }) – daca in $() punem o funtie, Jquery il va echivala cu

document.addEventListener(„DOMContentLoaded”);

$(„”).blur(function(event) {... }) – cand vede ca drep parametru i se ofera un string, devine un query selector. Adica, e echivalent cu document.querySelector(„”).addEventListener(„blur”,function(event) { ...} )