## *Изпитна тема № 12:* **Интернет програмиране**

Мрежови протоколи (IP, TCP, UDP). Видове HTTP заявки (GET/POST/PUT/DELETE/PATCH). Клиент-сървърна комуникация. Основни тагове в HTML. Селектори и основни правила в CSS. Създаване на семантични страници. Създаване на адаптивно (responsive) оформление на стра-ници. Увод в Javascript – работа с обекти и събития. Принципи и манипулиране на DOM.

| **Критерии за оценяване на изпитна тема № 12** | **Максимален брой точки** |
|---|---|
| 1.Обяснява и диференцира различните протоколи. | 12 |
| 2. Дефинира понятието HTTP заявка, прави изводи за различните HTTP методи и избира метод за конкретна ситуация. | 18 |
| 3. Обяснява и представя графично клиент-сървърната комуникация. | 12 |
| 4.Различава смисъла на употребата и необходимостта от HTML, CSS и JavaScript. | 8 |
| 5. Дефинира и използва коректно HTML тагове. | 8 |
| 6. Задава свойства на HTML компонентите чрез CSS. | 6 |
| 7. Описва и демонстрира употребата на семантични елементи за създаване на семантична страница. | 8 |
| 8. Обяснява и демонстрира начините за създаване на адаптивен (responsive) дизайн. | 10 |
| 9. Работа с обекти и събития в JavaScript | 12 |
| 10. Демонстрира манипулирането на DOM. | 6 |
| **ОБЩ БРОЙ ТОЧКИ:** | **100** |

1. **Обяснява и дефинира различните протоколи(12 т.):**
   - IP

     Internet Protocol (IP) е протокол за комуникация, който стои в основата на интернет. Предназначението му е да позволи адресация на информацията, която се изпраща по мрежата. На всеки хост в мрежата се дава уникален адрес (наречен IP адрес). Когато се изпраща информация през мрежата, тя се разделя на малки пакети, наречени IP пакети. Към всеки пакет се прикрепя т. нар. хедър, който съдържа IP адреса на подателя и получателя и други служебни данни. С помощта на тези адреси компютрите, през които минава пакетът, решават какво да правят с него. IP адресите са два вида. ***IPv4*** адрес - използва 32 битово адресиране. 32 битовият адрес се записва като поредица от четири 8 битови числа (октети). Всеки един от тези октети може лесно да се преобразува в десетично число. ***IPv6*** адрес **-** използва 128 битов адрес. Представя се като осем 16 битови двоични числа. 16 битовите числа се записват в шестнадесетична бройна система и се разделят помежду си с двоеточия.

- TCP

  TCP (Transmission Control Protocol) е мрежов протокол за управление на обмена на информация, един от основните, използвани в интернет. Използвайки TCP, приложенията в мрежата могат да създават връзки (connections) едно с друго и чрез тях да обменят данни в пакети. Информацията, която трябва да бъде транспортирана, бива разделена на огромно множество от пакети, всеки от които съдържа достатъчно информация да бъде пренасочен към точната си дестинация. Надеждността на обмена се осигурява от контролни суми и сравнения между изпратените и пристигналите данни. Другата важна функция на протокола е да провери, че пакетите биват подредени в правилен ред по времето на пристигането си. Протоколът се използва съвместно с IP протокола, като обикновено ги наричат TCP/IP комплект от протоколи (на английски: protocol suite).
- UDP

  User Datagram Protocol (UDP) е минимален транспортен пакетен протокол. UDP не гарантира доставката на данните: не се пази информация за изпратено съобщение, заради което UDP понякога се превежда и като „Unreliable Datagram Protocol“. UDP добавя към IP пакета единствено мултиплексиране на приложения (чрез номерата на портовете) и проверка на целостта на данните (чрез контролна сума. В случай че е необходима надеждност на предаването, тя трябва да се реализира в потребителското приложение.

  Програмите, използващи UDP протокола трябва да реализират самостоятелно:
  · препредаване на загубени дейтаграми;
  · игнориране на повторения;
  · фрагментиране и обединяване на големи потоци данни.

  В локалните мрежи грешките при предаване на информация са пренебрежимо малки. Използването на UDP протокол ще генерира по-малък трафик. Обикновено, при избора на UDP протокол се търси не намаляване на трафика, а по-скоро намаляване на времето за доставка и отговор. Например, при използване на UDP при предаване на глас, загубата на сегмент ще доведе до дефект – пукане. Използването на TCP при същото приложение ще доведе до забавяне – накъсване на звука, сериозно забавяне времето за отговор.

**2. Дефинира понятието HTTP заявка, прави изводи за различните HTTP методи и избира метод за конкретна ситуация(18 т.):**

HTTP(hypertext transfer protocol) е мрежов протокол за пренос на информация в Интранет мрежи и World Wide Web, първоначално създаден като средство за публикуване на HTML страници. Има две версии на HTTP, версия HTTP / 1.0 и най-новата версия HTTP / 1.1. Промяната, извършена в ревизията е главно във връзката за всяка транзакция със заявка и отговор. В предишната му версия е необходима отделна връзка. В по-късната версия връзката може да се използва многократно.

В HTTP протокола се използват понятия като клиент (обикновено това са Web-браузърите - т.е. самите приложения, а не физически хостовете в мрежата) и сървър (това са Web-сървърите - т.е. самите приложения, а не хостовете в мрежата).

HTTP определя 8 различни методи на заявки:

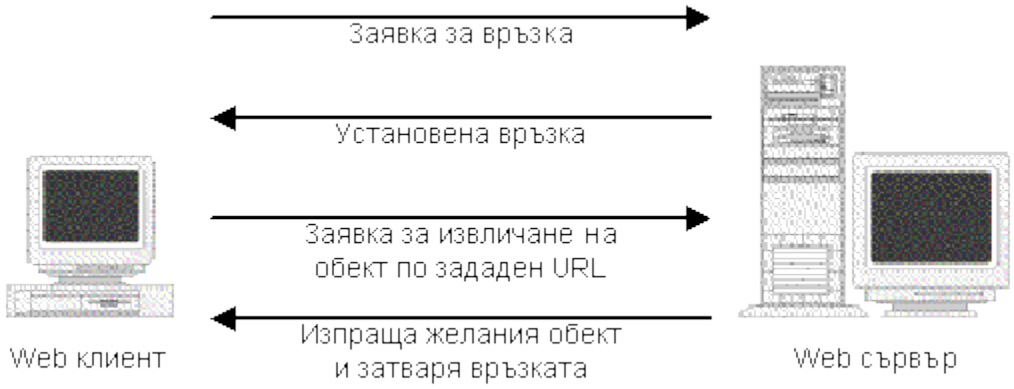
- GET – с него клиентът прави заявка за ресурс, зададен чрез URL. Могат да се изпращат и ограничено количество данни, закодирани директно в самия URL (отделени чрез въпросителен знак);
- POST – позволява клиентът да изпрати данни на сървъра. Тази заявка обикновено се генерира при изпращането на уеб формуляр, а данните могат да бъдат: текст, написан от потребителя във формуляра; файл на клиентския компютър и др.;
- PUT – качва файл, който в бъдеще ще отговаря на посочения URL.
- DELETE – изтрива посочения ресурс;
- PATCH – изисква целевият ресурс да промени състоянието си в съответствие с частичната актуализация, дефинирана в представянето, приложено в заявката. Това може да спести честотна лента чрез актуализиране на част от файл или документ, без да се налага да го прехвърляте изцяло;
- TRACE – сървърът връща получената заявка със статус OK. Позволява да се провери в какъв вид пристига заявката при сървъра и дали (и как) е била модифицирана по трасето от междинни прокси сървъри.
- OPTIONS – сървърът трябва да отговори с поддържаните от него клиентски методи, съответстващи на зададения URL, или с поддържаните от сървъра методи като цяло, ако е зададено * вместо URL.
- CONNECT – използва се при комуникация през прокси.

## 3. Обяснява и представя графично клиент-сървърната комуникация(12 т.):

Основният модел за обмен на информация в Интернет е модела “клиент/сървър”.

Клиент-сървър е тип мрежова архитектура, която отделя клиента от сървъра и най-често се използва в компютърни мрежи. Всеки клиент или сървър, свързан с мрежата, може също така да бъде използван като възел. Най-елементарната типова клиент-сървър архитектура се състои от две части – от едната страна е сървърът, от другата страна е клиентът Постигането на определен резултат при използване винаги е свързано с комуникация между двете страни;

WWW е Интернет услуга, в основата на която стои също модела клиент/сървър. Комуникацията между Web клиента и Web сървъра се осъществява чрез използване на протокола HTTP. **HTTP** (**H**yper **T**ext **T**ransfer **P**rotocol) служи за обмен на документи между сървър и клиент, и е част от протоколния стек TCP/IP за управление на поток от данни в Интернет. Всъщност протокола HTTP функционира на базата на проста схема от тип “въпрос–отговор". Клиентът изпраща заявка към сървъра, на която сървърът отговаря.




Освен при HTTP подобна схема на комуникация се прилага и при други протоколи – например FTP (File Transfer Protocol). Общо казано Web сървърът ще изпраща поисканите от клиентите заявки и файлове. Разликата между това дали ще използваме FTP или HTTP сървър се състои в значително по–богатата функционалност на HTTP сървъра. Ако изградим нашия сайт с Web страници под форма на хипермедийни документи, може да сложим освен текста и асоциирани с него графични, звукови или видео компоненти. Тогава, при условие, че клиент е отправил заявка към сайта ни, като резултат ще му бъдат изпратени всички елементи на документа, т.е всички съставни компоненти ще влязат в документа-резултат. Освен това според естеството на заявката и средствата за нейната обработка, от сървъра към клиента може да се изпращат и генерирани динамично данни (CGI интерфейс, Java аплет или Active X контрола).

**4. Различава смисъла на употребата и необходимостта от HTML, CSS и JavaScript(8 т.):**

- **HTML** – използва се за оформлението на уеб сайтове. HTML (HyperText Markup Language) не е програмен език, а описание на това как искаме да изглежда съдържанието, на структурата на самото съдържание. Изработването на HTML документа става с помощта на така наречените тагове. Чрез тях се създават и се определя разположението на отделните елементи в уеб страницата като заглавия, таблици, форми, бутони, текстови полета, текст, изображения, видеа и др.HTML е текст, които ние подаваме на браузъра, а той от своя страна го интерпретира като последователност от команди/инструкциите, които изпълнява и по този начин подрежда обектите в страницата. Основната функция на HTML е да определи структурата на документа от нейното съдържание като текст и мултимедиа. Със CSS и JavaScript вече може да създадем стилове на елементите от документа като: цвят, размер, яркост и много други.
- **CSS** – изгражда дизайна на уеб сайтове; CSS (Cascading Style Sheets) представлява скриптов език за описание на стилове.Чрез CSS се задават различни атрибути на

вече използваните тагове в HTML документа като цвят, размери, фон, разположение, прозрачност и много, много други.

CSS си има и своите стилови правила за изписване. Използва селектори, чрез които се избира върху кой елемент и таг да бъде приложен дадения стил.

Добра практика е само структурата и съдържанието да са в HTML документа, а стилизирането да е в отделен CSS файл. Връзката между HTML и CSS файловете може да се направи по няколко начина:

external CSS файл, свързан с HTML файла чрез тага link;
inline CSS, в самия HTML елемент чрез тага style;
дирекотно изписване на свойствата и стойностите им като атрибути към даден таг;

- **JavaScript** – използва се за добавяне на някакъв вид поведение на HTML елементите и математически операции в уеб сайтове. Предназначението му е да улесни динамичната обработка на HTML документи при клиента. Скрипт кодът се включва като част от HTML кода. Изпълнението на скрипта става след интерпретация от навигатора на клиента при изтегляне на страницата от сървъра. Позволява на страницата да реагира на действията на посетителите, дава възможност за използване на специални ефекти (визуални и др.). JavaScript не се нуждае от компилатор и е по-снизходителен в някои области като синтаксиса. JavaScript е обектно-базиран скриптов език от страна на клиента, който се използва, за да се направят по-динамични Web страници. Обектно-базиран означава, че могат да се използват елементи като обекти; от страна на клиента означава, че JavaScript се изпълнява от софтуера, който използва посетителя, а не Web сървъра на сайта, обслужващ тази страница; скриптов език означава, че не изисква компилиране преди изпълнение.

5. **Дефинира и използва коректно HTML тагове(8 т.):**
   - **<html>** - Указва на браузъра, че това е HTML документ. Отбелязва началото и края на документа и съдържа всички други негови елементи (с изключение на <!DOCTYPE> елемента);
   - **<head>** - Съдържа заглавието на документа, и може да съдържа стилове, скриптове, енкодинг и т.н.;
   - **<body>** - Съдържа форматиране видимо за потребителя – текст, хиперлинк, картинки, таблици, бутони, параграфи и т.н.;
   - **<!DOCTYPE>** - Декларира се първи, още преди <html> тага. Валидира документа. <!DOCTYPE> не е HTML таг. Той е инструкция за уеб браузъра – указва HTML версията, на която е написана страницата;
   - **<b> или ** - Удебелява текста;
   - <i> или  - Задава курсив/наклон на текста;
   - **<u>** - Подчертава текста;
   - **<sub>** - Дефинира текст под черта;
   - **<sup>** - Дефинира текст над черта;
   - **<br>** - Указва нов ред. Няма таг за край;
   - **<a>** - Указва линк към друга страница. Най-важният атрибут на този таг е **href.** Той посочва URL адреса, към който сочи линка;

- **<img>** - Дефинира картинка в HTML страницата. Има два задължителни атрибута: **src** и alt. Атрибутът src указва URL адреса на картинката, alt – указва алтернативен текст на картинката, а hight и width – указват съответно височината и ширината на картинката в пиксели;
- Таговете от **<h1>** до **<h6>** дефинират заглавия в HTML документа. <h1> дефинира най-важното заглавие. <h6> дефинира най-маловажното заглавие;
- **<p>** - указва параграф. Браузърите автоматично добавят по един празен ред преди и след текста, маркиран като параграф;
- Таблици - Дефинират се с тага **<table>**. Таблицата е разделена на редове, чрез тага **<tr>** („table row“), а всеки ред е разделен на клетки с данни (чрез тага **<td>**, „table data“). Всяка клетка може да съдържа текст, линкове, картинки, списъци, форми, други таблици и т.н. Атрибутът border задава рамка на таблицата;
- Подредените списъци започват с тага **<ol>**, а всеки елемент на списъка – с тага **<li>**;
- Неподредени списъци - При тези списъци номерацията не е с цифри или букви, а с кръгли точки. Всеки списък започва с тага **<ul>**, а всеки елемент на списъка – с тага <li>;

6. **Задава свойства на HTML компонентите чрез CSS(6 т.):**
   - Задаването на свойства на HTML компонентите се осъществява чрез атрибутът style написан вътре в HTML таг, тагът <style>…</style> или отделен файл със CSS код с разширение .css, който се линква в <head> тага.

     Примери:
     **Стилове във външен CSS файл**
     <head>
     <link rel="stylesheet" type="text/css" href="stil1.css" />
     </head>
     CSS файлът съдържа само CSS команди (няма HTML тагове), например:
     h1 {color:red}
     p {text-indent:1em;}
     th {background-color:#DD2599}

     **Указване на стил в главата на HTML (вътрешен стил)**
     Когато се създава уникален стил, който ще бъде използван само за една страница, той се указва като вътрешен стил: в главата на HTML документа (head). За целта се използват таговете <style> и </style>, между които се разполагат CSS командите са стил.
     <head>
     <style>
     h1 {color:red}
     p {text-indent:1em;}
     th {background-color:#DD2599}
     </style>
     </head>
     **Стил в самия HTML елемент(in-line)**

Ако се зададе специфичен стил за един конкретен елемент може да се направи по следния начин:
<p style="text-indent:1em; color:green">

**7. Описва и демонстрира употребата на семантични елементи за създаване на семантична страница(8 т.):**

Семантичният HTML5 не е предназначен за потребителите на сайта, а за търсещите машини и ботове, които обикалят интернет пространството и за начина, по които те „виждат" различните сайтове. Семантичният HTML помага на търсачките да се ориентират в сайта. За тази цел се използват семантични HTML елементи наричани още семантични тагове, които указват на търсачките дали дадено съдържание или ключова дума например е важна, дали определено словосъчетание е заглавие и пр.

Семантични са таговете:

- **<header>** – представлява заглавие на елемент или група от елементи. Тага <header> позволява използването на няколко заглавия в един документ за разлика от възможността която предоставя HTML4;
- **<nav>** – дефинира навигацията на сайта. Той е предназначен за големи блокове от навигационни връзки. Не всички връзки трябва да са в <nav> елемента, а само главното навигационно меню;
- **<article>** – представлява предмет/статия. Съдържанието в <article> трябва да има смисъл само по себе си и да може да се разпространява независимо и отделно от останалата част на сайта. Използва се най-често за публикация във форум; публикация в блог; новини; коментари и др.;
- **<section>** е тематично групирано съдържание, обикновено със заглавие. Този елемент позволява влагане на нова сегмантична структура във вече съществуващ елемент;
- **<aside>** – той представя съдържание което е странично от основното съдържание на страницата. Съдържанието в <aside> се счита за неважно и може да бъде пропуснато, ако сайта се възпроизвежда на устройство с малък екран като телефон;
- **<footer>** – може да съдържа информация за автора на страницата; навигация на сайта; авторските права върху използваните материали и връзки към други (подобни) публикации;

**8. Обяснява и демонстрира начините за създаване на адаптивен (responsive) дизайн(10 т.):**

- Адаптивният уеб дизайн се изгражда като на CSS свойството width на даден HTML таг се зададе стойност в проценти или auto. Така този дизайн няма да позволява излизането на елементите извън предела на границите на устройството. При адаптивния дизайна могат да бъдат налични няколко оформления на сайта с определени фиксирани размери за различните устройства.
  Пример:

```
@media only screen and (min-width:750px) {
        /*CODE*/
}
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
* {

@media only screen and (max-width: 720px) {
 /* For mobile phones: */
 .menu, .main, .right {
  width: 100%;
 }
}
</style>
```

„Responsive дизайн“ реагира на промени в дължината на екрана на браузъра, пренастройвайки позициите на елементите, така че те да се вместят в наличното пространство. Казано иначе – респонсивният дизайн спомага за визуализацията на елементите на един сайт според предоставеното му място в прозореца на браузъра. Ако се отвори сайт, който е проектиран респонсивно и се промени размера на прозореца на браузъра – съдържанието ще се размести динамично, за да отговори на новия размер.

Responsive дизайнът разчита на динамична промяна на позициите на елементите, в зависимост от свободното място. За разлика от него, адаптивният дизайн борави с множество предварително фиксирани изгледи. Когато сайтът засече наличното място, той избира изгледа, който да извика. Адаптивният дизайн е по-статичен, тъй като ако повторим горния пример за десктоп, при промяна на размера на прозореца на браузъра, изгледът няма да се промени – сайтът вече е заредил определен шаблон и този шаблон не подлежи на динамично пренареждане.

**9. Работа с обекти и събития в JavaScript.**

**Модифицира код спрямо конкретна задача(12 т.):**

- Обектите в JavaScript са променливи, които могат да съдържат много стойности. Обектът е начин на моделиране на нещо реално

Пример:
```
const person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```
Обектът **person** има свойствата **firstName, lastName, id** . Достъпът до свойствата на обекта се извършва поредством оператора точка “.”
Методите са функции, които извършват различни операции със свойствата на обектите.
В JavaScript има предварително зададени обекти, а има възможност и за съсздаване на собствени. Някои от вградените обекти са :

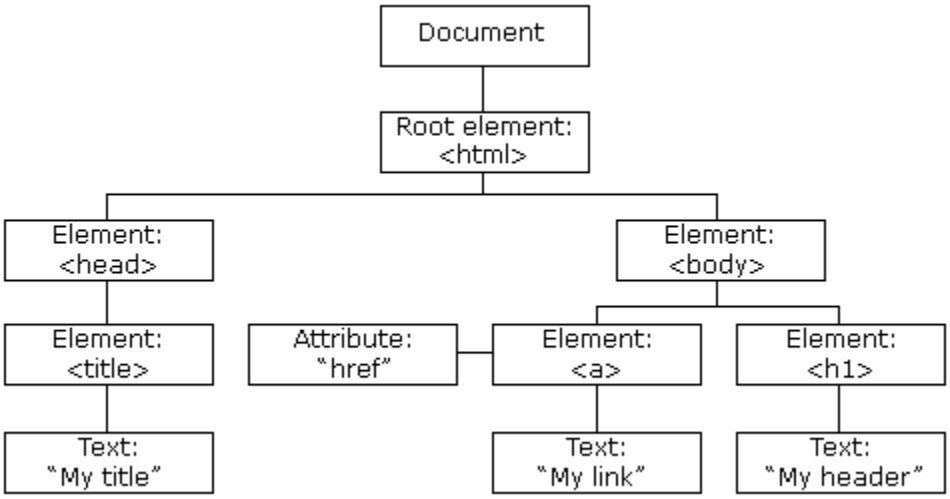
- Navigator – със свойства appName(съсдържа типа на браузъра, който използва потребителя), appVersion(версията на браузъра), language (език, на който е преведен браузъра) и методи javaEnabled(), plugins.refresh();
- Document – свойства linkColor, bgColor, images и методи open(), close(), write();
- Window – свойства closed, name, location и методи alert(), confirm(), print(), prompt()

- Събитията са предварително дефинирани думи, които се използват за обработване на дадено събитие в web страницата. Събитие се сучва, когато потребителят извърши действия например с мишка или клавиатура. Когато JavaScript се използва в HTML страници, JavaScript може да "реагира" на тези събития. Такива събития са:
  - onchange - HTML елемент е променен;
  - onclick - Потребителят щраква върху HTML елемент;

- onmouseover - Потребителят премества мишката върху HTML елемент;
- onmouseout - Потребителят премества мишката от HTML елемент;
- onkeydown - Потребителят натиска клавиш на клавиатурата;
- onload - Браузърът приключи със зареждането на страницата;

10.Демонстрира манипулирането на DOM(6 т.):

- DOM (от английски Document Object Model) представлява модел на обектите в една уеб страница.




- Тези обекти могат да бъдат манипулирани посредством JavaScript(тоест да бъдат променяни), като с JavaScript могат да се направят следните неща:
  - JavaScript може да промени всички HTML елементи на страницата
  - JavaScript може да промени всички HTML атрибути на страницата
  - JavaScript може да промени всички CSS стилове в страницата
  - JavaScript може да премахне съществуващите HTML елементи и атрибути
  - JavaScript може да добавя нови HTML елементи и атрибути
  - JavaScript може да реагира на всички съществуващи HTML събития в страницата
  - JavaScript може да създава нови HTML събития в страницата
- Пример за това е променянето на текста в даден параграф:

```
<html>
<body>

<p id="p1">Здравейте!</p>

<script>
document.getElementById("p1").innerHTML = "Довиждане!";
</script>

</body>
</html>
```