Лабораторная работа №11
Объект JavaScript – Date.

1. **Цель**

   Изучить свойства, методы объекта **Date**, научиться применять их на практике.

2. **Методические указания**
   1. При изучении конструкций языка HTML можно использовать любой текстовый редактор. Для получения HTML файла, сохраняйте свои изменения как текстовые, для файла используйте расширение *.htm или *.html.
   2. Более опытные пользователи могут воспользоваться любым специализированным редактором HTML страниц (Macromedia Dreamweaver, HomeSite, FrontPage, AceHTML, Web Development Studio)
   3. Создаваемые файлы необходимо тестировать в основных браузерах Internet Explorer, Mazilla Firefox, Opera.
   4. Отлаженные файлы необходимо сохранять в отдельном каталоге.
   5. По окончанию работы сохраните все созданные файлы на своих носителях.

3. **Теоретические сведения**

   Для работы с датой и временем в JavaScript существует специальный объект - Date. Этот объект поддерживается практически всем версиями JavaScript, а зачит им можно пользоваться не оглядываясь на проблемы совместимости.

   Дата и время в объекте Date хранятся не в явном виде, а как и в большинстве языков программирования - в виде количества миллисекунд, прошедших с дня рождения Unix, т.е. с 0 часов 0 минут 1 января 1970 года. Отличительная особенность объекта Date - все диапазонные значения имеют индексы, начинающиеся с нуля. Это означает, что январь будет иметь индекс 0 (месяц №0), а декабрь будет не двенадцатым, а одиннадцатым месяцем. То же самое справедливо и для дней недели, часов, минут и пр.

   Создать объект Date очень просто:

```
// текущая дата-время
var date = new Date();

// дата-время из строки или числа
```

var date = new Date(дата);

// дата-время из отдельных значений
var date = new Date(год, месяц, день, час, минута, секунда, миллисекунда);

Объект Date обладает рядом очень полезных методов, позволяющих работать с отдельными компонентами даты-времени, а так же проводить проверку корректности и правильный вывод даты в заданном формате.

Объект Date обладает рядом очень полезных методов, позволяющих работать с отдельными компонентами даты-времени, а так же проводить проверку корректности и правильный вывод даты в заданном формате.

Методы получения компонентов даты-времени	
getFullYear	Возвращает год (например, 2012)
getYear	Возвращает год. Назначение метода getYear аналогично назначению getFullYear, однако данный метод является устаревшим и не рекомендуются к использованию, т.к. результаты его работы не однозначны: для диапазона дат от 1900 до 1999 года метод возвращает номер года в столетии (двузначный, например 77), а для дат за пределами этого диапазона - возвращается полное значение (четырёхзначное, например 2009).
getMonth	Возвращает месяц.
getDate	Возвращает день месяца (число в месяце).
getHours	Возвращает час.
getMinutes	Возвращает минуту.
getSeconds	Возвращает секунду.
getMilliseconds	Возвращает миллисекунду.
getDay	Возвращает номер деня недели.
getTime	Возвращает миллисекундное смещение,

	хранимое объектом.
Методы изменения компонентов даты-времени	
setFullYear	Устанавливает год.
setYear	Устанавливает год. Назначение метода setYear аналогично назначению setFullYear, однако данный метод является устаревшим и не рекомендуются к использованию (так же как и метод getYear).
setMonth	Устанавливает месяц.
setDate	Устанавливает дату в месяце (день месяца).
setHours	Устанавливает час.
setMinutes	Устанавливает минуту.
setSeconds	Устанавливает секунду.
setMilliseconds	Устанавливает миллисекунду.
setTime	Устанавливает миллисекундное смещение относительно 00:00:00 01.01.1970
Фунции форматирования и вывода даты-времени	
toString	Возвращает строковое представление даты и времени.
toUTCString	Возвращает строковое представление даты и времени с пересчётом на время UTC. Формат возвращаемой строки учитывает все интернет-стандарты.
toGMTString	Возвращает строковое представление даты и времени с пересчётом на время GMT (время по Гринвичу). Формат возвращаемой строки учитывает все интернет-стандарты.
toLocaleString	Аналог toString, но возвращает строковое представление даты и времени, отформатированное в соответствии с настройками локализации данного пользователя.
toTimeString	Возвращает строковое представление времени (строка содержит только время).
toDateString	Возвращает строковое представление даты (строка содержит только дату).

toLocaleTimeString	Аналог toTimeString, но возвращает строковое представление времени, отформатированное в соответствии с настройками локализации пользователя.
toLocaleDateString	Аналог toDateString, но возвращает строковое представление даты, отформатированное в соответствии с настройками локализации пользователя.
Дополнительные функции	
getTimezoneOffset	Возвращает смещение локального времени на компьютере пользователя относительно времени UTC. Смещение возвращается в минутах.
parse	Функция позволяет проверить корректность даты-времени, записанных в виде строки. Если строка корректная - сразу будет создан объект Date.

Так же объект Date содержит ряд методов для работы с UTC-датами. Эти функции полностью аналогичны уже рассмотреным, но содержат в имени префикс "UTC" и работают только с "универсальным" временем: getUTCSeconds, setUTCFullYear и т.д.

Рассмотрим пример работы с датами:

```
<script type="text/javascript">
        var tmp = new Date();
        var arrNames = new Array('Сегодня', 'Завтра',
                'Через 1 год 6 месяцев 15 дней');
        var arrNormal = new Array();
        var arrLocal = new Array();
        var arrUTC = new Array();
 // сейчас
        arrLocal[0] = tmp.toLocaleString();
        arrNormal[0] = tmp.toString();
        arrUTC[0] = tmp.toUTCString();
  // завтра
        tmp.setDate(tmp.getDate() + 1);
        arrLocal[1] = tmp.toLocaleString();
        arrNormal[1] = tmp.toString();
        arrUTC[1] = tmp.toUTCString();
```

```
// через 1 год 6 месяцев 15 дней
        tmp.setFullYear(tmp.getFullYear() + 1);
        tmp.setMonth(tmp.getMonth() + 6);
        tmp.setDate(tmp.getDate() + 15);
        arrLocal[2] = tmp.toLocaleString();
        arrNormal[2] = tmp.toString();
        arrUTC[2] = tmp.toUTCString();
// выводим результаты в таблицу
        document.write('<table border="1"><tr>'
                                        + '<td>Дата</td>'
                                        + '<td>Локализованная</td>'
                                        + '<td>Не локализованная</td>'
                                        + '<td>UTC</td></tr>');

        for (i = 0; i < 3; i++)
        {
                document.write('<tr>'
                        + '<td>' + arrNames[i] + '</td>'
                        + '<td>' + arrLocal[i] + '</td>'
                        + '<td>' + arrNormal[i] + '</td>'
                        + '<td>' + arrUTC[i] + '</td>'
                        + '</tr>'
                );
        }

        document.write('</table>');
</script>
```

Рассмотрим теперь скрипт, создающий на экране изображение работающих часов:

```
<html>
<head>

<script Language="JavaScript">
<!-- hide

var timeStr, dateStr;

function clock() {
      now= new Date();
```

```
        // врем\я
        hours= now.getHours();
        minutes= now.getMinutes();
        seconds= now.getSeconds();
        timeStr= "" + hours;
        timeStr+= ((minutes < 10) ? ":0" : ":") + minutes;
        timeStr+= ((seconds < 10) ? ":0" : ":") + seconds;
        document.clock.time.value = timeStr;

        // дата
        date= now.getDate();
        month= now.getMonth()+1;
        year= now.getYear();
        dateStr= "" + month;
        dateStr+= ((date < 10) ? "/0" : "/") + date;
        dateStr+= "/" + year;
        document.clock.date.value = dateStr;

        Timer= setTimeout("clock()",1000);
}

// -->
</script>
</head>

<body onLoad="clock()">

<form name="clock">
 Время:
  <input type="text" name="time" size="8"  value=""><br>
 Дата:
  <input type="text" name="date" size="8" value="">
</form>

</body>
</html>
```

Здесь для ежесекундной коррекции времени и даты мы пользуемся методом setTimeout(). Фактически это сводится к кому, что мы каждую секунду создаем новый объект Date, занося туда текущее время. Можно видеть, что функции clock() вызываются программой обработки

события onLoad, помещенной в тэг <body>. В разделе body нашей HTML-страницы имеется два элемента формы для ввода текста. Функция clock() записывает в оба эти элемента в корректном формате текущие время и дату. Для этой цели используются две строки *timeStr* и *dateStr*. Как мы уже упомянули ранее, существует проблема с индикацией, когда количество минут меньше 10 - в данном скрипте эта проблема решается с помощью следующей строки:

```
timeStr+= ((minutes < 10) ? ":0" : ":") + minutes;
```

Как видим, количество минут заносится в строку *timeStr*. Если у нас менее 10 минут, то мы еще должны приписать спереди 0. Для Вас эта строка в скрипте может показаться немного странной, и ее можно было бы переписать в более знакомом Вам виде:

```
if (minutes < 10) timeStr+= ":0" + minutes else timeStr+= ":" + minutes;
```

Как видите, представление даты существенно отличается в зависимости от используемого формата. Поэтому при работе с датой-временем надо придерживаться нескольких простых правил:

1. По возможности пользоваться UTC или GMT-форматами. Особенно это важно при создании распределённых решений (например, клиентов платёжных систем). Использование общего опорного времени даст вам гарантии (пусть и не стопроцентные), что и вы и ваш удалённый партнёр будете одинаково интерпретировать получаемые данные.
2. Локализованные дату и время имеет смысл использовать только при выводе их пользователю. Во всех остальных случаях от локализованных данных лучше отказаться.
3. Если всё же приходится использовать локальные дату и время - не забывайте учитывать смещение локального времени относительно опорного (UTC или GMT).

Следование этим правилам избавит вас от большинства логических ошибок и недочётов, а значит сделает ваш код более стабильным и качественным.

4. Содержание отчёта

Отчёт должен содержать название и цель. Ход работы все свойства, методы. Выполненный вариант. Выводы.

5. Контрольные вопросы

1. Что делает метод getHours.
2. Что делает метод getDay.
3. Что делает метод getTime.
4. Объясните разницу между getDate и setDate.
5. Что делает метод setMonth.
6. Для чего нужны UTC или GMT-форматами

Варианты заданий:

№ варианта	Задание
1	Написать программу, которая показывает число рабочих дней прошедших до Рождества.
2	Написать программу, которая рассчитывает разницу дат, возвращает количеству дней.
3	Написать программу, которая рассчитывает разницу дат, возвращает количеству минут.
4	Написать программу, которая возвращает для определенной даты день недели.
5	Написать программу, которая рассчитывает сколько человеку лет.
6	Написать программу, которая вычисляет дату пасхи.
7	Написать программу, которая вычисляет номер месяца по его имени.
8	Написать программу, которая возвращает имя дня недели.
9	Написать программу, которая возвращает название квартала по дате.
10	Написать программу, которая рассчитывает разницу во времени.