

## Лабораторна робота №2

**Тема:** Робота з датою та часом.

**Мета:** Навчитися використовувати дату і час для виводу інформації

### ТЕОРЕТИЧНІ ВІДОМОСТІ

Вбудований об'єкт: `Date`. Він містить дату і час, а також надає методи управління ними.

Наприклад, його можна використовувати для зберігання часу створення / зміни, для вимірювання часу або просто для виведення поточної дати.

#### Створення

Для створення нового об'єкта `Date` потрібно викликати конструктор `new Date()` з одним з таких аргументів:

`new Date()`

Без аргументів - створити об'єкт `Date` з поточними датою і часом:

```
let now = new Date();  
alert( now ); // показує поточну дату і час
```

#### `new Date(milliseconds)`

Створити об'єкт `Date` з часом, що дорівнює кількості мілісекунд (тисячна частка секунди), що пройшли з 1 січня 1970 року UTC + 0.

```
// 0 відповідає 01.01.1970 UTC+0  
let Jan01_1970 = new Date(0);  
alert( Jan01_1970 );  
  
// тепер додаємо 24 години і отримаємо 02.01.1970 UTC+0  
let Jan02_1970 = new Date(24 * 3600 * 1000);  
alert( Jan02_1970 );
```

Створити об'єкт `Date` з заданими компонентами в місцевому часовому поясі. Обов'язкові тільки перші два аргументи.

`Year` повинен складатися з чотирьох цифр: значення 2013 коректне, 98- немає.

`Month` починається з 0(січень) за 11(грудень).

Параметр `date` тут є день місяця. Якщо параметр не заданий, то приймається значення 1.

Якщо параметри `hours/minutes/seconds/ms` відсутні, їх значенням стає 0.

наприклад:

```
new Date(2011, 0, 1, 0, 0, 0, 0); // // 1 Jan 2011, 00:00:00
```

```
new Date(2011, 0, 1); // теж саме 0
```

Максимальна точність - 1 мс (до 1/1000 секунди):

```
let date = new Date(2011, 0, 1, 2, 3, 4, 567);  
alert( date ); // 1.01.2011, 02:03:04.567
```

### Отримання компонентів дати

Існують методи отримання року, місяця і т.д. з об'єкта `Date`:

[`getFullYear\(\)`](#)

Отримати рік (4 цифри)

[`getMonth\(\)`](#)

Отримати місяць, від 0 до 11 .

[`getDate\(\)`](#)

Отримати день місяця, від 1 до 31, що дещо суперечить назвою методу.

[`getHours\(\)`](#) , [`getMinutes\(\)`](#) , [`getSeconds\(\)`](#) , [`getMilliseconds\(\)`](#)

Отримати, відповідно, години, хвилини, секунди або мілісекунди.

Ніякого `getYear()`. тільки `getFullYear()`

Багато інтерпретаторів JavaScript реалізують нестандартний і застарілий метод `getYear()`, який інколи повертає рік у вигляді двох цифр. Будь ласка, обходьте його стороною. Якщо потрібно значення року, використовуйте `getFullYear()`.

Крім того, можна отримати певний день тижня:

[`getDay\(\)`](#)

Повернути день тижня від 0(неділя) до 6(субота). Незважаючи на те, що в ряді країн за перший день тижня прийнятий понеділок, в JavaScript початок тижня припадає на неділю.

**Всі перераховані вище методи повертають значення відповідно до місцевого часового поясу.**

Однак існують і їх UTC-варіанти, які повертають день, місяць, рік для тимчасової зони UTC + 0: `getUTCFullYear()` , `getUTCMonth()` , `getUTCDay()` . Для їх використання потрібно після "get" підставити "UTC".

Якщо ваш місцевий часовий пояс зміщений відносно UTC, то наступний код покаже різні години:

```
// поточна дата
```

```
let date = new Date();
```

```
// час
```

```
alert( date.getHours() );
```

```
// час в часовому поясі UTC+0 alert( date.getUTCHours() );
```

Крім вищенаведених методів, існують два особливих методу без UTC-варіанту:

`getTime()`

Для заданої дати повертає таймстамп - кількість мілісекунд, що пройшли з 1 січня 1970 року UTC + 0.

`getTimezoneOffset()`

Повертає різницю у хвилинах між місцевим часовим поясом і UTC:

## Встановлення компонентів дати

Наступні методи дозволяють встановити компоненти дати і часу:

- `setFullYear(year, [month], [date])`
- `setMonth(month, [date])`
- `setDate(date)`
- `setHours(hour, [min], [sec], [ms])`
- `setMinutes(min, [sec], [ms])`
- `setSeconds(sec, [ms])`
- `setMilliseconds(ms)`
- `setTime(milliseconds)` (Встановлює дату у вигляді цілого кількості мілісекунд, що пройшли з 01.01.1970 UTC)

У всіх цих методів, крім `setTime()`, є UTC-варіант, наприклад: `setUTCHours()`.

Як ми бачимо, деякі методи можуть встановлювати відразу кілька компонентів дати, наприклад: `setHours`. Якщо якась компонента не вказана, вона не змінюється.

приклад:

```
let today = new Date();

today.setHours(0);
alert(today);
today.setHours(0, 0, 0, 0);
alert(today);
```

## Перетворення до числа, різниця дат

Якщо об'єкт `Date` перетворити в число, то отримаємо тайм стамп за аналогією з `date.getTime()`:

```
let date = new Date();
```

```
alert(+date); // кількість мілісекунд, те саме що  
date.getTime()
```

Важливий побічний ефект: дати можна вичитати, в результаті отримуємо різницю в мілісекундах.

Цей прийом можна використовувати для вимірювання часу:

```
let start = new Date(); // починаємо відлік  
  
// виконуємо певні дії  
for (let i = 0; i < 100000; i++) {  
    let doSomething = i * i * i;  
}  
  
let end = new Date(); // закінчуємо відлік дії  
  
alert( `Цикл оброблено за ${end - start} мілісекунд` );
```

## [Date.now \(\)](#)

Якщо потрібно просто виміряти час, об'єкт `Date` нам не потрібен.

Існує особливий метод `Date.now()`, який повертає поточну мітку часу.

Семантично він еквівалентний `new Date().getTime()`, однак метод не створює проміжний об'єкт `Date`. Так що цей спосіб працює швидше і не навантажує збирач сміття.

Даний метод використовується з міркувань зручності або коли важливо швидкодію, наприклад, при розробці ігор на JavaScript або інших спеціалізованих додатків.

Ймовірно, попередній приклад краще переписати так:

```
let start = Date.now(); // кількість мілісекунд з 1 січня  
1970 року  
  
// виконуємо певні дії  
for (let i = 0; i < 100000; i++) {  
    let doSomething = i * i * i;  
}
```

```
let end = Date.now(); // закінчуємо відлік часу

alert( `Цикл оброблено за ${end - start} мілісекунд` );

// різниця числа, а не дати
```

### Розбір рядка з датою

Метод `Date.parse(str)` зчитує дату з рядка.

Формат рядка повинен бути наступним: `YYYY-MM-DDTHH:mm:ss.sssZ`, де:

`YYYY-MM-DD` - це дата: рік-місяць-день.

Символ "T" використовується як роздільник.

`HH:mm:ss.sss` - час: години, хвилини, секунди і мілісекунди.

Необов'язкова частина `'Z'` означає часовий пояс в форматі `+hh:mm`. Якщо вказати просто букву `Z`, то отримаємо UTC + 0.

Можливі й більш короткі варіанти, наприклад, `YYYY-MM-DD` або `YYYY-MM`, або навіть `YYYY`.

Виклик `Date.parse(str)` обробляє рядок в заданому форматі і повертає таймстамп (кількість мілісекунд з 1 січня 1970 року UTC + 0). Якщо формат неправильний, повертається `NaN`.

наприклад:

```
let ms = Date.parse('2012-01-26T13:51:50.417-07:00');

alert(ms); // 1327611110417 (таймстамп)
```

Можна тут же створити об'єкт `new Date` з таймстамп:

```
let date = new Date( Date.parse('2012-01-26T13:51:50.417-07:00') );

alert(date);
```

### ОТЖЕ

Дата і час в JavaScript представлені об'єктом `Date`. Не можна створити «тільки дату» або «тільки час»: об'єкти `Date` завжди містять і те, і інше.

Рахунок місяців починається з нуля (так, січень - це нульовий місяць).

Дні тижня в `getDay()` також відраховуються з нуля, що відповідає неділі.

Об'єкт `Date` самостійно коригується при введенні значень, що виходять за рамки допустимих. Це корисно для додавання / віднімання днів / місяців / тижнів.

Дати можна вчитати, і різниця повертається в мілісекундах. Так відбувається, тому що при перетворенні в число об'єкт `Date` стає таймстамп.

Використовуйте `Date.now()` для швидкого отримання поточного часу в форматі таймстамп.

Врахуйте, що, на відміну від деяких інших систем, в JavaScript таймстамп в мілісекундах, а не в секундах.

## **ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ**

1. Виведіть на екран поточні **день, місяць і рік** в форматі **'рік-місяць-день'** .
2. Виведіть на екран **поточний місяць словом** , по-російськи.

**Рішення:** створимо масив місяців **months** , потім отримаємо номер поточного місяця за допомогою **getMonth** , і виведемо місяць словом, звернувшись до елементу масиву **months** з ключем, рівним номеру поточного місяця, ось так - **months [month]** :

### **Робота з new Date**

3. Виведіть на екран **поточний день** .
4. Виведіть на екран **поточний місяць** .
5. Виведіть на екран **поточний рік** .
6. Виведіть на екран поточну дату-час в форматі **'12: 59: 59 31.12.2014 '** . Для вирішення цього завдання **напишіть функцію** , яка буде додавати **0** перед днями і місяцями, які складаються з однієї цифри (з **1.9.2014** зробить **01.09.2014** ).

### **Робота з getDay**

*Для вирішення завдань даного блоку вам знадобляться наступні методи: getDay .*

7. Виведіть на екран **номер поточного дня тижня** . Показати рішення.
8. Виведіть на екран **поточний день тижня** (словом, українською). Створіть для цього допоміжну функцію **showDay** , яка параметром приймає число, а повертає **день тижня українською** .
9. Дізнайтеся, який був **7-го січня 2015 року** .

### **Робота з getTime**

*Для вирішення завдань даного блоку вам знадобляться наступні методи: getTime .*

10. Виведіть на екран **кількість хвилин з 1-го січня 1970 року до теперішнього моменту часу**.

### **Робота з Date.parse**

*Для вирішення завдань даного блоку вам знадобляться наступні методи: Date.parse .*

11. Виведіть на екран **кількість годин** , що минув між **1 березня 1988 року** і поточним моментом за допомогою **Date.parse** .

### **Різниця між датами**

12. Виведіть на екран **кількість секунд з початку дня до теперішнього моменту часу**.
13. Виведіть на екран **кількість секунд, яке залишилося до кінця дня** .
14. Створіть **prompt**, в який користувач вводить дату свого народження в форматі **'2014-12-31'** (з конкретним роком). За втрати фокусу виведіть скільки днів залишилося до його дня народження. Скористайтесь методом **Date.parse**.