Obiekty i funkcje wbudowane JS, które warto znać

JavaScript udostępnia wiele gotowych do użycia obiektów i różnorodnych funkcji wbudowanych, które bywają bardzo przydatne w praktyce, a zarazem często mogą pojawić się na egzaminie INF.03.

Okna dialogowe przeglądarki

JavaScript pozwala na komunikację z użytkownikiem przy pomocy tzw. okien dialogowych. Są to proste okna otwierane "nad" kartą przeglądarki (tzw. otwarcie modalne) i pełnią różne funkcje. Znamy je zapewne przede wszystkim z zajęć szkolnych, w praktyce webowej widujemy je już stosunkowo rzadko.

Okno alert()

Najprostsze z okien dialogowych to

`alert()`, które służy do wyświetlenia komunikatu użytkownikowi. To świetny sposób na przekazanie informacji, np. ostrzeżenia, powiadomienia o błędzie albo wyświetlenia wartości jakiejś zmiennej:

```
alert("Witaj świecie");
let liczba = 10;
alert("Wartość zmiennej liczba: " + liczba);
```

Okno prompt()

Czasami chcemy także, aby użytkownik wpisał w wyskakującym oknie jakąś wartość, np. swoje imię czy wiek albo wartość numeryczną. W takim przypadku używamy

`prompt()`:

```
let wynik = prompt("Jak masz na imię?");
```

Przeglądarka otworzy okienko modalne wyposażone w pole tekstowe do wpisania wartości. Wpisana wartość zostanie zwrócona i zapisana w stworzonej zmiennej

`wynik` – przypomina to trochę zapytanie użytkownika o wartość w programie konsolowym napisanym w *C++* albo *Python*.

Okno confirm()

Czasami potrzebujemy zapytać użytkownika (uzyskać potwierdzenie), czy na pewno chce czegoś dokonać – np. usunąć plik, opuścić stronę, zapisać zmiany. Do tego służy trzecie i ostatnie "wyskakujące" okienko

`confirm()`, które wyświetla monit z dwoma przyciskami: "OK" i "Anuluj". Zobaczmy przykład – załóżmy istnienie następującego formularza w HTML:

```
<form id="formularz">
  <label>Imię: <input type="text" name="imie"></label><br>
  <label>Nazwisko: <input type="text" name="nazwisko"></label><br>
  <input type="reset" value="Resetuj" onclick="return potwierdzReset();">
</form>
```

Natomiast skrypt JS zawierający nasze okienko wygląda tak:

```
function potwierdzReset() {
  return confirm("Czy na pewno chcesz wyczyścić formularz?");
}
```

Przycisk

`<input type="reset">` standardowo wyczyści zawartość wszystkich pól formularza. Nasza zdefiniowana funkcja `potwierdzReset()` wywołuje okno dialogowe z dodatkowym, własnym potwierdzeniem takiej operacji. Jeśli użytkownik kliknie “OK” funkcja zwróci wartość true i formularz zostanie wyczyszczony. Jeżeli jednak użytkownik wybierze “Anuluj”, to dzięki zwróceniu false przeglądarka anuluje domyślną akcję resetowania.

Zapewne zauważyliśmy też tym razem obecność dodatkowego słowa

`return` w wartości atrybutu `onclick` :

```
<input type="reset" value="Resetuj" onclick="return potwierdzReset();">
```

Zostało tam wstawione, ponieważ bez obecności

`return` wynik tej funkcji nie wpłynąłby na to, czy przeglądarka wykona domyślną akcję przycisku `type="reset"` , czyli czyszczenie zawartości pól formularza. Natomiast kiedy dodamy `return` , to właśnie wynik naszej funkcji zdecyduje o tym, czy domyślna akcja ma być kontynuowana (true) czy anulowana (false). Oczywiście gdybyśmy mieli do czynienia ze “zwykłym” przyciskiem, czyli `type="button"` to wówczas dodatkowy `return` nie byłby potrzebny, a akcję resetu formularza – uchwyconego dzięki np. atrybutowi `id` – moglibyśmy wywołać następująco:

```
document.getElementById("formularz").reset();
```

Czy “wyskakujące” okna znajdują jeszcze zastosowanie?

Reasumując – okna dialogowe w JavaScript to w dużej mierze relikty dawnych interfejsów systemowych – w praktyce lepszym rozwiązaniem jest korzystanie z kontrolek formularzy HTML, ponieważ modalne okna typu alert, prompt czy confirm mogą niepotrzebnie irytować użytkowników stron internetowych. Bywają one jednak czasami używane w zadaniach egzaminacyjnych, a ponadto okienko

`alert()` może nam posłużyć do szybkiego sprawdzenia wartości zmiennej albo poprawności podpięcia funkcji do obsługi zdarzenia.

Obiekt Math – matematyka w JS

JavaScript posiada wbudowany obiekt

`Math` , który oferuje wiele przydatnych funkcji matematycznych – wiele z nich może się nam przysłużyć w zadaniach egzaminacyjnych. Obiekt ten nie jest konstruktorem – nie tworzymy więc (jakoś nazwanej) jego instancji, tylko korzystamy z niego bezpośrednio w skryptach. Oto kilka najważniejszych metod, które do egzaminu na pewno warto znać:

- **Math.round(x)** – zaokrągla liczbę do najbliższej liczby całkowitej.

```
Math.round(4.3);  // 4
Math.round(4.6);  // 5
```

- **Math.floor(x)** – zaokrągla liczbę w dół (do najbliższej mniejszej lub równej).

```
Math.floor(5.9);   // 5
Math.floor(-2.1);  // -3
```

- **Math.ceil(x)** – zaokrągla liczbę w górę (do najbliższej większej lub równej).

```
Math.ceil(3.1);   // 4
Math.ceil(-3.1);  // -3
```

- **Math.abs(x)** – zwraca wartość bezwzględną liczby.

```
Math.abs(-7);  // 7
Math.abs(7);   // 7
```

- **Math.sqrt(x)** – pierwiastek kwadratowy.

```
Math.sqrt(25);   // 5
Math.sqrt(-1);   // NaN
```

- **Math.pow(x, y)** – potęguje liczbę x do potęgi y.

```
Math.pow(2, 3);   // 8
```

- **Math.max(...)** – największa liczba z podanych argumentów.

```
Math.max(3, 9, 1);  // 9
```

- **Math.min(...)** – najmniejsza liczba z podanych argumentów.

```
Math.min(3, 9, 1);  // 1
```

- **Math.random()** – losowa liczba z zakresu `<0; 1)`. Generowanie liczb losowych omówimy dokładniej w następnej części tej lekcji – to często występująca mechanika na egzaminach. Przykład podstawowy:

```
Math.random();  // np. 0.5321
```

- **Math.trunc(x)** – *ang. truncate = ucięcie, ścięcie* – metoda ucina część dziesiętną liczby (bez jej zaokrąglania):

```
Math.trunc(4.9);   // 4
Math.trunc(-4.9);  // -4
```

- **Math.sign(x)** – zwraca znak liczby jako wartość: 1, -1 lub 0.

```
Math.sign(-10);  // -1
Math.sign(0);    // 0
Math.sign(5);    // 1
```

- **Math.cbrt(x)** – pierwiastek trzeciego stopnia ($\sqrt[3]{x}$).

```
Math.cbrt(27);   // 3
```

- **Math.log(x)** – logarytm naturalny (log przy podstawie e).

```
Math.log(Math.E);  // 1
```

- **Math.exp(x)** – e do potęgi x (czyli e^x).

```
Math.exp(1);   // ~2.718
```

- **Math.sin(x), Math.cos(x), Math.tan(x), Math.asin(x), Math.acos(x), Math.atan(x)** – metody obliczania funkcji trygonometrycznych (nazwy metody mówią same za siebie), natomiast co warto wiedzieć – jednostką miary kąta x są radiany:

```
var x = 0.523599;  // Miara w radianach kąta 30 stopni
Math.sin(x);       // Sinus kąta 30 stopni to 0.5
```

Liczby losowe – więcej informacji o Math.random()

Jak już wiemy,

`Math.random()` w JS służy do generowania liczb pseudolosowych. Jest bardzo przydatna mechanika na egzaminie INF.03 do zadań takich jak losowanie liczb, symulacje, wybór losowego elementu etc. Domyślnie metoda zwraca liczbę zmiennoprzecinkową z zakresu `<0; 1)`, jednak często zachodzi potrzeba zrealizowania w skrypcie losowania z zupełnie innego zakresu. Przedstawmy zatem teraz kilka popularnych przypadków:

1. Losowanie liczby całkowitej z przedziału od 0 do n–1

```
let losowa = Math.floor(Math.random() * 10); // liczba od 0 do 9
```

2. Losowanie liczby całkowitej z przedziału od a do b (włącznie z b)

```
let a = 5;
let b = 12;
let losowa = Math.floor(Math.random() * (b - a + 1)) + a;
```

3. Losowanie liczby zmiennoprzecinkowej z przedziału a do b (bez b)

```
let a = 2.5;
let b = 5.0;
let losowa = Math.random() * (b - a) + a;
```

4. Wylosuj czterocyfrowy kod PIN

```
let pin = Math.floor(1000 + Math.random() * 9000); // od 1000 do 9999
```

5. Rzut kostką sześciościenną (1–6)

```
let kostka = Math.floor(Math.random() * 6) + 1;
```

6. Losowy wybór elementu (jego indeksu) z tablicy

```
let kolory = ["czerwony", "zielony", "niebieski"];
let losowyIndeks = Math.floor(Math.random() * kolory.length);
let kolor = kolory[losowyIndeks];
```

Pytanie jakie przy tej okazji rodzi się w głowie brzmi: czy istnieją inne sposoby generowania liczb pseudolosowych w JS? Otóż

`Math.random()` jest jedynym **wbudowanym** sposobem generowania liczb losowych. Jeśli potrzebujemy bardziej zaawansowanej kontroli nad losowością w swoich własnych projektach (poza egzaminem, bo tam przecież nie mamy dostępu do internetu), to zawsze możemy użyć bibliotek zewnętrznych, takich jak:

- seedrandom – generowanie liczb z tzw. kontrolowanym ziarniem,
- random-js – alternatywny silnik losujący inspirowany losowaniem znanym z *C++11*,
- chance.js – losowanie imion, miast, numerów PESEL itp.

Co jeszcze ważne –

`Math.random()` nie jest bezpieczny kryptograficznie, stąd nie należy go używać do generowania haseł, tokenów ani kodów potwierdzających do logowania w systemach produkcyjnych.

Obiekt Date – praca z datą i czasem

W JavaScript obiekt

`Date` (jak sama nazwa wskazuje) służy do pracy z datą i czasem. Umożliwia nam pobieranie bieżącej daty, godziny, a także wykonywanie operacji na czasie – co bywa czasem przydatne na egzaminie INF.03, np. przy sprawdzaniu wieku użytkownika czy wyświetlaniu aktualnej daty na stronie.

W praktyce, najczęściej tworzymy instancję obiektu

`Date` bez parametrów – wtedy przyjmuje ona aktualną datę i czas:

```
let teraz = new Date();
alert(teraz);
```

Możemy też jednak podać konkretną datę (lub także opcjonalnie czas) jako argument w postaci tekstowej, inicjując w ten sposób obiekt konkretną datą:

```
let data = new Date("2024-05-12");
```

Obiekt

`Date` oferuje wiele metod odczytu konkretnych, interesujących nas części daty bądź czasu:

- `getFullYear()` – zwraca rok (np. 2025)
- `getMonth()` – zwraca miesiąc od 0 do 11 (styczeń = 0)
- `getDate()` – zwraca dzień miesiąca (1–31)
- `getDay()` – zwraca dzień tygodnia (0 = niedziela, 6 = sobota)
- `getHours()`, `getMinutes()`, `getSeconds()` – czas: godziny, minuty, sekundy

Przykładowy kod do szybkiego przetestowania tych metod:

```
let dzis = new Date();
alert("Rok: " + dzis.getFullYear());
alert("Miesiąc: " + (dzis.getMonth() + 1));
alert("Dzień: " + dzis.getDate());
alert("Godzina: " + dzis.getHours() + ":" + dzis.getMinutes());
```

Przykład: obliczanie wieku użytkownika

Załóżmy, że użytkownik podał swój rok urodzenia. Dzięki metodzie

`getFullYear()` łatwo możemy teraz obliczyć jego aktualny wiek w latach:

```
let rokUrodzenia = 2005;
let teraz = new Date();
let wiek = teraz.getFullYear() - rokUrodzenia;
console.log("Użytkownik ma " + wiek + " lat.");
```

Przykład: formatowanie daty

Czasami chcemy wyświetlić w skrypcie datę w formacie “windowsowym”: DD.MM.RRRR zamiast bazodanowym RRRR-MM-DD. Możemy tego łatwo dokonać metodą

`toLocaleDateString()`:

```
let dzisiaj = new Date();
console.log(dzisiaj.toLocaleDateString()); // np. "30.04.2025"
```

Czas wykonania kodu w milisekundach

Obiekt

`Date` umożliwia także porównywanie czasu przez pobranie liczby milisekund od 1 stycznia 1970 (*tzw. epoka Unixa*) – co pozwoli nam zmierzyć czas wykonania jakichś operacji czy algorytmów (np. sortowanie liczb):

```
let start = new Date();
// jakaś operacja lub funkcja której czas chcemy zmierzyć
let koniec = new Date();
let roznica = koniec.getTime() - start.getTime();
console.log("Czas wykonania: " + roznica + " ms");
```

Metody ustawiające wartości w obiekcie Date:

Oprócz pobierania wartości dat metodami odczytującymi (które nazywamy *getterami*), czasem przydatne mogą się okazać także tzw. *settery*, czyli funkcje ustawiające wartości:

- `setFullYear(rok)` – ustawia rok
- `setMonth(miesiąc)` – ustawia miesiąc (pamiętajmy, że zakres to 0–11)
- `setDate(dzień)` – ustawia dzień miesiąca

Przykład ustawienia wartości metodami:

```
let d = new Date();
d.setFullYear(2000);
d.setMonth(0); // styczeń
d.setDate(1);
console.log(d.toLocaleDateString()); // "01.01.2000"
```

Inne przydatne obiekty wbudowane

- **String** – pozwala na operacje na tekstach – jednak to opisaliśmy dokładnie w lekcji: Skrypty przetwarzające napisy.
- **Array** – tablice i metody z nimi związane – o tym opowiedzieliśmy w lekcji: Tablice (kolekcje) w JS.
- **console** – obiekt używany jako podręczna pomoc w debugowaniu: `console.log()`, `console.error()` oraz `console.table()`, czyli wyświetlanie danych w formie czytelnej tabeli w konsoli przeglądarki.

Podsumowanie

Warto dobrze opanować podstawowe obiekty – z dużym prawdopodobieństwem mogą pojawić się na egzaminie INF.03. Przedstawmy wnioski z lekcji:

- **Okna dialogowe** (modalne): `alert()`, `prompt()` i `confirm()` służą do komunikacji z użytkownikiem – są proste, ale rzadko stosowane w nowoczesnych aplikacjach.
  - `alert()` – wyświetla komunikat (np. do debugowania lub ostrzeżenia).
  - `prompt()` – umożliwia pobranie wartości od użytkownika i zapisanie jej do zmiennej.
  - `confirm()` – pozwala użytkownikowi potwierdzić lub anulować działanie (zwraca `true` lub `false`).

- **Obiekt Math** zawiera wiele przydatnych metod matematycznych (np. `Math.round()`, `Math.sqrt()`, `Math.random()`).
- `Math.random()` generuje liczby pseudolosowe z zakresu `<0; 1)`; lecz można go użyć do losowania np. PIN-u, koloru, wartości całkowitej z zadanego przedziału.
- **Obiekt Date** służy do operacji na czasie i dacie – umożliwia pobieranie bieżącej daty, godziny, obliczanie wieku czy mierzenie czasu wykonania operacji.
- Do odczytu daty służą gettery, np. `getFullYear()`, `getMonth()`, `getDate()`; do ustawiania – settery, np. `setFullYear()`.
- Metoda `toLocaleDateString()` umożliwia formatowanie daty w sposób znany z Windows (np. “30.04.2025”).
- **Obiekt console** umożliwia debugowanie: `console.log()`, `console.error()` oraz `console.table()` (czytelna tabelka z danymi).