Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_Е.В.Паскал  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ |

|  |  |
| --- | --- |
| Специальность:  2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина: «Скриптовые языки программирования» |

**Лабораторная работа № 21**

**Инструкционно-технологическая карта**

Тема: Глобальный объект node.js и js. Работа с браузерной и серверной средой.

Цель:

Время выполнения: 2 часа

# Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.
2. Реализовать алгоритм решения задачи.
3. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
4. Ответить на контрольные вопросы.

# Краткие теоретические сведения

Глобальный объект предоставляет переменные и функции, доступные в любом месте программы. По умолчанию это те, что встроены в язык или среду исполнения.

В браузере он называется **window**, в **Node.js – global**, в другой среде исполнения может называться иначе.

Недавно **globalThis** был добавлен в язык как стандартизированное имя для глобального объекта, которое должно поддерживаться в любом окружении. В некоторых браузерах, например в старой версии **Edge** который не был на **Chromium**, **globalThis** ещё не поддерживается, но легко реализуется с помощью полифила.

Далее будет использоваться **window**, полагая, что среда – браузер. Если скрипт может выполняться и в другом окружении, лучше будет **globalThis**.

Ко всем свойствам глобального объекта можно обращаться напрямую:

alert("Привет");

// это то же самое, что и

window.alert("Привет");

В браузере глобальные функции и переменные, объявленные с помощью **var** (не **let/const**!), становятся свойствами глобального объекта:

var gVar = 5;

alert(window.gVar); // 5 (становится свойством глобального объекта)

Такое поведение поддерживается для совместимости. В современных проектах, использующих **JavaScript** модули, такого не происходит.

Если бы была объявлена переменную при помощи **let**, то такого бы не произошло:

let gLet = 5;

alert(window.gLet); // undefined (не становится свойством глобального объекта)

Если свойство настолько важное, что его необходимо сделать его доступным для всей программы, это свойство следует записать в глобальный объект напрямую:

// сделать информацию о текущем пользователе глобальной, для предоставления доступа всем скриптам

window.currentUser = {

    name: "John"

};

// где угодно в коде

alert(currentUser.name); // John

// или, если у нас есть локальная переменная с именем "currentUser",

// получим её из window явно (безопасно!)

alert(window.currentUser.name); // John

При этом обычно не рекомендуется использовать глобальные переменные. Следует применять их как можно реже. Дизайн кода, при котором функция получает входные параметры и выдаёт определённый результат, чище, надёжнее и удобнее для тестирования, чем когда используются внешние, а тем более глобальные переменные.

## Использование для полифилов

Глобальный объект можно использовать, чтобы проверить поддержку современных возможностей языка.

Например, проверить наличие встроенного объекта **Promise** (такая поддержка отсутствует в очень старых браузерах):

if (!window.Promise) {

    alert("Ваш браузер очень старый!");

}

Если нет (скажем, используется старый браузер), можно создать полифил: добавить функции, которые не поддерживаются окружением, но существуют в современном стандарте.

if (!window.Promise) {

    window.Promise = ... // собственная реализация современной возможности языка

}

## Глобальные объекты node.js

Также как и window, Global позволяет использовать какие-то встроенные свойства и методы из любого места в коде.

#### \_\_filename

Указывает имя файла сценария, выполняемого в данный момент. Он выведет абсолютный путь к расположению файла, и он может не совпадать с именем файла, указанным в параметре командной строки. Если в модуле, возвращаемое значение – это путь к файлу модуля.

#### \_\_dirname

Указывает каталог, в котором находится исполняемый в данный момент сценарий.

#### setTimeout(cb, ms)

Глобальная функция выполняет указанную функцию (cb) через указанное количество миллисекунд (мс). : **SetTimeout** () выполняет указанную функцию только один раз.

Возвращает значение дескриптора, представляющее таймер.

#### clearTimeout(t)

Глобальная функция используется для остановки таймера, ранее созданного **setTimeout** (). Параметры t – это таймер, созданный функцией **setTimeout** ().

#### setInterval(cb, ms)

Глобальная функция выполняет указанную функцию (cb) через указанное количество миллисекунд (мс).

Возвращает значение дескриптора, представляющее таймер. можно использовать **clearInterval**(t)Функция сброса таймера.

Метод **setInterval** () будет продолжать вызывать функцию до тех пор, пока не будет вызван **clearInterval** () или окно не будет закрыто.

#### console

Консоль используется для обеспечения стандартного вывода консоли. Это инструмент отладки, предоставляемый движком **JScript** Internet Explorer, который постепенно стал стандартом реализации браузера.

Node.js следует этому стандарту и предоставляет консольные объекты, согласующиеся с обычным поведением для вывода символов в стандартный поток вывода (**stdout**) или стандартный поток ошибок (**stderr**).

#### console.log ()

Выводит символы в стандартный поток вывода и заканчивается символом новой строки. **console.log** получает несколько параметров, если параметр только один, выводится строковая форма этого параметра. Если параметров несколько, они будут выведены в формате, аналогичном формату команды **printf ()** языка Си.

# Контрольные вопросы

1. Что такое глобальный объект?
2. Как обратиться к глобальному объекту в браузере?
3. Перечислите глобальные объекты node.js

# Задания

1. Создайте массив, запишите его в глобальный объект window и выведите в консоль.
2. Через диалоговое окно спросите у пользователя число от 100 до 1000, после чего прокрутите страницу вниз на полученное число.
3. Измените размер окна на случайный.
4. Создайте несколько html страниц, наполните их данными. Реализуйте навигацию по получившимся страницам без использования гиперссылок.

# Литература

**Диков, А.В.** Клиентские технологии веб программирования: JavaScript и DOM: учебное пособие / А.В. Диков. – СПб: Лань, 2020 – 124 с.

**Читанамбри, Кирупа.** Изучаем React / Кирупа Читанамбри – 2-е изд. – М.: Эксмо, 2019. 368 с.

**Хавербеке, Марейн.** Выразительный JavaScript. Современное веб-программирование. / Марейн Хавербеке – 3-е изд. – СПб.: Питер, 2019. – 480 с.:

Преподаватель Рогалевич А.В.

Рассмотрено на заседании цикловой

комиссии программного обеспечения

информационных технологий №10

Протокол № \_\_\_ от \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Председатель ЦК\_\_\_\_\_\_\_\_ В.Ю.Михалевич