## What's the output?

## function sayHi() {

## console.log(name);

## console.log(age);

## var name = "Lydia";

## let age = 21;

## }

## sayHi();

## A: Lydia and undefined

## B: Lydia and ReferenceError

## C: ReferenceError and 21

## D: undefined and ReferenceError

Name существует

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## What's the output?

## for (var i = 0; i < 3; i++) {

## setTimeout(() => console.log(i), 1);

## }

## for (let i = 0; i < 3; i++) {

## setTimeout(() => console.log(i), 1);

## }

## A: 0 1 2 and 0 1 2

## B: 0 1 2 and 3 3 3

## C: 3 3 3 and 0 1 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## What's the output?

## +true;

## !"Lydia";

## A: 1 and false

## B: false and NaN

## C: false and false

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## What's the output?

## const shape = {

## radius: 10,

## diameter() {

## return this.radius \* 2;

## },

## perimeter: () => 2 \* Math.PI \* this.radius

## };

## console.log(shape.diameter());

## console.log(shape.perimeter());

## A: 20 and 62.83185307179586

## B: 20 and NaN

## C: 20 and 63

## D: NaN and 63

У стрелочной функции нет "this".

## Which one is true?

## const bird = {

## size: "small"

## };

## const mouse = {

## name: "Mickey",

## small: true

## };

## A: mouse.bird.size is not valid

## B: mouse[bird.size] is not valid

## C: mouse[bird["size"]] is not valid

## D: All of them are valid

|  |  |
| --- | --- |
|  | [bird.size] И [bird["size"]] - одно и то же. |
|  |  |
| [size] - глобальная переменная. | |
|  |  |
| [bird["size"]] - тоже not valid. | |

## What's the output?

## let c = { greeting: "Hey!" };

## let d;

## d = c;

## c.greeting = "Hello";

## console.log(d.greeting);

## A: Hello

## B: Hey!

## C: undefined

## D: ReferenceError

## E: TypeError

## What's the output?

## let a = 3;

## let b = new Number(3);

## let c = 3;

## console.log(a == b);

## console.log(a === b);

## console.log(b === c);

## A: true false true

## B: false false true

## C: true false false

## D: false true true

|  |  |
| --- | --- |
|  | "===" |
| проверяется содержимое и типы данных. | |
| ссылка - всегда true.  "==" - сравнение только по значениям (содержимое), а не по типу переменных.  Сравнение объектов - по ссылке. Если одна и та же ссылка, то  isSame (только для ссылок),  isEqual.  2 примитива не м.б. isSame, они м.б. только isEqual.  Number(0) = true.  0, как примитив - это false.  От реализации интерфейса зависит - "==" или "===" - что быстрее работает.  Обычно сравнивают "===".  Число и String, как примитив, м. сравниваться "===". | |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## What happens when we do this?

## function bark() {

## console.log("Woof!");

## }

## bark.animal = "dog";

## A: Nothing, this is totally fine!

## B: SyntaxError. You cannot add properties to a function this way.

## C: "Woof" gets logged.

## D: ReferenceError

"SyntaxError" - почти никогда не бывает.

Ничего не будет.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## What's the output?

## Class Person(firstName, lastName) {

## this.firstName = firstName;

## this.lastName = lastName;

## }

## const lydia = new Person("Lydia", "Hallie");

## const sarah = Person("Sarah", "Smith");

## console.log(lydia);

## console.log(sarah);

## A: Person {firstName: "Lydia", lastName: "Hallie"} and undefined

## B: Person {firstName: "Lydia", lastName: "Hallie"} and Person {firstName: "Sarah", lastName: "Smith"}

## C: Person {firstName: "Lydia", lastName: "Hallie"} and {}

## D:Person {firstName: "Lydia", lastName: "Hallie"} and ReferenceError

Без new нельзя.

Д.б. "function Person(firstName, lastName) {

"

вместо

## " Class Person(firstName, lastName) {

".

## What's the output?

## function sum(a, b) {

## return a + b;

## }

## sum(1, "2");

## A: NaN

## B: TypeError

## C: "12"

## D: 3

конкатенация.

## What's the output?

## let number = 0;

## console.log(number++);

## console.log(++number);

## console.log(number);

## A: 1 1 2

## B: 1 2 2

## C: 0 2 2

## D: 0 1 2

"number++" - сначала выводится number.

## What's the output?

## function checkAge(data) {

## if (data === { age: 18 }) {

## console.log("You are an adult!");

## } else if (data == { age: 18 }) {

## console.log("You are still an adult.");

## } else {

## console.log(`Hmm.. You don't have an age I guess`);

## }

## }

## checkAge({ age: 18 });

## A: You are an adult!

## B: You are still an adult.

## C: Hmm.. You don't have an age I guess

"===" - это equals в Pyton.

2 разные ссылки на разные области памяти.

"==" и "===" - для ссылок, ссылки сравниваются.

Сравнятся через JSON.stringify.

JSON.stringify - точное сравнение.

Объекты сравниваются по ссылкам.

Раз разные ссылки, то разные объекты.

Иначе, если н. сравнить объекты не по ссылкам, а по значениям, то н. JSON.stringify.

## What's the output?

## function getAge() {

## "use strict";

## age = 21;

## console.log(age);

## }

## getAge();

## A: 21

## B: undefined

## C: ReferenceError

## D: TypeError

## What's value of sum?

## const sum = eval("10\*10+5");

## A: 105

## B: "105"

## C: TypeError

## D: "10\*10+5"

evaluation = оценка, вычисляет любое выражение. Результат выражения вернет.

## const sum = eval("10+'5'");

то "105".

## What's the output?

## var num = 8;

## var num = 10;

## console.log(num);

## A: 8

## B: 10

## C: SyntaxError

## D: ReferenceError

var перезаписывает.

## What's the output?

## const obj = { a: "one", b: "two", a: "three" };

## console.log(obj);

## A: { a: "one", b: "two" }

## B: { b: "two", a: "three" }

## C: { a: "three", b: "two" }

## D: SyntaxError

## What's the output?

## const foo = () => console.log("First");

## const bar = () => setTimeout(() => console.log("Second"));

## const baz = () => console.log("Third");

## bar();

## foo();

## baz();

## A: First Second Third

## B: First Third Second

## C: Second First Third

## D: Second Third First

bar не выполняется, только дает состояние. Установили событие, а браузер б. выполнять в момент, когда браузер перейдет в состояние ожидание.

## What's the output?

## !!null;

## !!"";

## !!1;

## A: false true false

## B: false false true

## C: false true true

## D: true true false

null - false.

## What does this return?

## [..."Lydia"];

## A: ["L", "y", "d", "i", "a"]

## B: ["Lydia"]

## C: [[], "Lydia"]

## D: [["L", "y", "d", "i", "a"]]

## [..."Lydia"]; - массив, разбивает на части. Строка - множество символов, то spread разобьет на множество элементов.

"..." - spread.

## 20. What does this return?

## const firstPromise = new Promise((res, rej) => {

## setTimeout(res, 500, "one"); //the method res will be called in 500 milleseconds with argument “one”

## });

## const secondPromise = new Promise((res, rej) => {

## setTimeout(res, 100, "two");

## });

## Promise.race([firstPromise, secondPromise]).then(res => console.log(res));

## //race – the method of Promise that gets several promises and performs an one that is resolved first

## A: "one"

## B: "two"

## C: "two" "one"

## D: "one" "two"

Пропускаем.

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## What's the output?

## let person = { name: "Lydia" };

## const members = [person];

## person = null;

## console.log(members);

## A: null

## B: [null]

## C: [{}]

## D: [{ name: "Lydia" }]

Spread для массива разбивает на отдельные элементы.

Char нет в JS.

Работает:

const [a, b] = [..."ab"];

Не работает:

const [a, b] = ..."ab";

const [a, b] = [..."ab"];

const ar = [];

ar.push(..."ab");

const n = new Number(0);

const c = n;

n = null;

const l = 2;

const b = l;

destructuring строки нельзя делать.

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## What's the output?

## const person = {

## name: "Lydia",

## age: 21

## };

## for (const item in person) { //iterates keys of an object

## console.log(item);

## }

## A: { name: "Lydia" }, { age: 21 }

## B: "name", "age"

## C: "Lydia", 21

## D: ["name", "Lydia"], ["age", 21]

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## What's the output?

## console.log(3 + 4 + "5");

## A: "345"

## B: "75"

## C: 12

## D: "12"

console.log(3 + 4 \* "5");

23

## 24.What's the output?

## const a = {};

## const b = { key: "b" };

## const c = { key: "c" };

## //Any object key should be a string ; if an object doesn’t have the method

## //toString , JS will convert the object to [object Object]

## a[b] = 123;

## a[c] = 456;

## console.log(a[b]);

## A: 123

## B: 456

## C: undefined

## D: ReferenceError

Ключ в объекте - только строка.

a[b] - доступ к полю b у объекта a.

const a = {};

const b = { key: "b", toString: function() {return this.key}};

const c = { key: "c" };

//Any object key should be a string ; if an object doesn’t have the method

//toString , JS will convert the object to [object Object]

a[b] = 123;

a[c] = 456;

console.log(a[b]);

123

const a = {};

const b = { key: "b", toString: function() {return this.key + 9}};

const c = { key: "c" };

//Any object key should be a string ; if an object doesn’t have the method

//toString , JS will convert the object to [object Object]

a.b = 123;

a[c] = 456;

console.log(a[b]);

undefined

Зарплата - строка, если не сделано преобразование в число.

## 25. What's the output?

## const numbers = [1, 2, 3];

## numbers[10] = 11;

## console.log(numbers.length);

## A: 11

## B: 4

## C: Error

## 26. What's the value of num?

## const num = parseInt("7\*6");

## A: 42

## B: "42"

## C: 7

## D: NaN

Обрабатывается только первое число.

let a = parseInt("k", 32);

console.log(a);

## 27. What's the output?

## function getInfo(member, year) {

## member.name = "Lydia";

## year = "1998";

## }

## const person = { name: "Sarah" };

## const birthYear = "1997";

## getInfo(person, birthYear);

## console.log(person, birthYear);

## A: { name: "Lydia" }, "1997"

## B: { name: "Sarah" }, "1998"

## C: { name: "Lydia" }, "1998"

## D: { name: "Sarah" }, "1997"

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 28. What's the output?

## function greeting() {

## throw "Hello world!";

## }

## function sayHi() {

## try {

## const data = greeting();

## console.log("It worked!", data);

## } catch (e) {

## console.log("Oh no an error:", e);

## }

## }

## sayHi();

## A: It worked! Hello world!

## B: Oh no an error: undefined

## C: SyntaxError: can only throw Error objects

## D: Oh no an error: Hello world!

## 29. What's the output?

## const numbers = [1, 2, 3, 4, 5];

## const [y] = numbers;

## console.log(y);

## A: [[1, 2, 3, 4, 5]]

## B: [1, 2, 3, 4, 5]

## C: 1

## D: [1]

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 30. What's the output?

## const user = { name: "Lydia", age: 21 };

## const admin = { admin: true, ...user };

## console.log(admin);

## A: { admin: true, user: { name: "Lydia", age: 21 } }

## B: { admin: true, name: "Lydia", age: 21 }

## C: { admin: true, user: ["Lydia", 21] }

## D: { admin: true }

C "..." - ссылка.

## 31. What's the output?

## const settings = {

## username: "lydiahallie",

## level: 19,

## health: 90

## };

## const data = JSON.stringify(settings, ["level", "health"]);//considered only // specified keys

## console.log(data);

## A: "{"level":19, "health":90}"

## B: "{"username": "lydiahallie"}"

## C: "["level", "health"]"

## D: "{"username": "lydiahallie", "level":19, "health":90}"

## 32. What's the output?

## let num = 10;

## const increaseNumber = () => num++;

## const increasePassedNumber = number => number++;

## const num1 = increaseNumber();

## const num2 = increasePassedNumber(num1);

## console.log(num1);

## console.log(num2);

## A: 10, 10

## B: 10, 11

## C: 11, 11

## D: 11, 12

## const increasePassedNumber = number => number++;

то же, что (для стрелочной функции ставим обязательно "()", если нет параметров; а если параметры есть, то их можно ставить, а можно не ставить в "()"):

## const increasePassedNumber = (number) => number++;

## 33. What's the output?

## [1, 2, 3, 4].reduce((x, y) => console.log(x, y));

## A: 1 2 and 3 3 and 6 4

## B: 1 2 and 2 3 and 3 4

## C: 1 undefined and 2 undefined and 3 undefined and 4 undefined

## D: 1 2 and undefined 3 and undefined 4

## 34. What's the output?

## function addToList(item, list) {

## return list.push(item);

## }

## const result = addToList("apple", ["banana"]);

## console.log(result);

## A: ['apple', 'banana']

## B: 2

## C: true

## D: undefined

sort и изменяет массив, и возвращает ссылку на тот же массив.

## 35. What is the output?

## const list = [1 + 2, 1 \* 2, 1 / 2]

## console.log(list)

## A: ["1 + 2", "1 \* 2", "1 / 2"]

## B: ["12", 2, 0.5]

## C: [3, 2, 0.5]

## D: [1, 1, 1]

## 36. What is the output?

## function sayHi(name) {

## return `Hi there, ${name}`

## }

## console.log(sayHi())

## A: Hi there,

## B: Hi there, undefined

## C: Hi there, null

## D: ReferenceError

## 37. What's the output?

## console.log("I want pizza"[0])

## A: """

## B: "I"

## C: SyntaxError

## D: undefined

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 38. What is the output?

## function checkAge(age) {

## if (age < 18) {

## const message = "Sorry, you're too young."

## } else {

## const message = "Yay! You're old enough!"

## }

## return message

## }

## console.log(checkAge(21))

## A: "Sorry, you're too young."

## B: "Yay! You're old enough!"

## C: ReferenceError

## D: undefined

## 39. What's the output?

## function sum(num1, num2 = num1) {

## console.log(num1 + num2)

## }

## sum(10)

## A: NaN

## B: 20

## C: ReferenceError

## D: undefined

## 40. What's the output?

## let newList = [1, 2, 3].push(4)

## console.log(newList.push(5))

## A: [1, 2, 3, 4, 5]

## B: [1, 2, 3, 5]

## C: [1, 2, 3, 4]

## D: Error

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 41. When you click the paragraph, what's the logged output?

## <div onclick="console.log('div')">

## <p onclick="console.log('p')">

## Click here!

## </p>

## </div>

## A: p div

## B: div p

## C: p

## D: div

## 42. What's the output?

## const person = { name: "Lydia" };

## function sayHi(age) {

## return `${this.name} is ${age}`;

## }

## console.log(sayHi.call(person, 21));

## console.log(sayHi.bind(person, 21));

## A: undefined is 21 Lydia is 21

## B: function function

## C: Lydia is 21 Lydia is 21

## D: Lydia is 21 function

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 43. What's the output?

## function sayHi() {

## return (() => 0)();

## }

## console.log(typeof sayHi());

## A: "object"

## B: "number"

## C: "function"

## D: "undefined"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

typeOf(функция) = Object.

typeOf(undefined) = undefined.

typeOf(NaN) = number.

## 44. Which of these values are falsy?

## 0;

## new Number(0);

## ("");

## (" ");

## new Boolean(false);

## undefined;

## A: 0, '', undefined

## B: 0, new Number(0), '', new Boolean(false), undefined

## C: 0, '', new Boolean(false), undefined

## D: All of them are falsy

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 45. What's the output?

## (() => {

## let x, y;

## try {

## throw new Error();

## } catch (x) {

## (x = 1), (y = 2);

## console.log(x);

## }

## console.log(x);

## console.log(y);

## })();

## A: 1 undefined 2

## B: undefined undefined undefined

## C: 1 1 2

## D: 1 undefined undefined

## \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

x в catch и x вне catch - разные.

## 46. What's the output?

## [[0, 1], [2, 3]].reduce(

## (acc, cur) => {

## return acc.concat(cur);

## },

## [1, 2]

## );

## A: [0, 1, 2, 3, 1, 2]

## B: [6, 1, 2]

## C: [1, 2, 0, 1, 2, 3]

## D: [1, 2, 6]