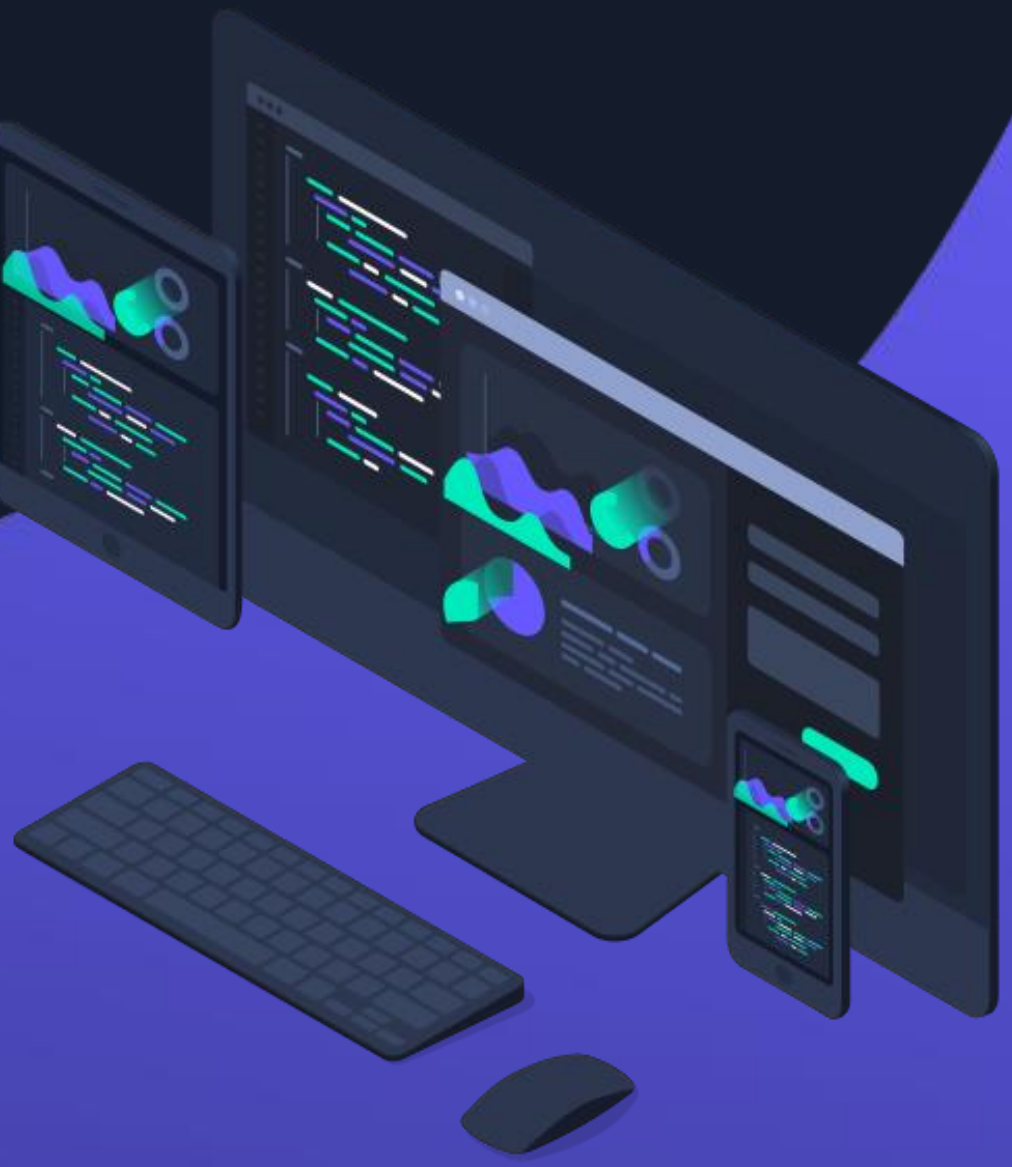
ITEA

ONLINE



I T E A
ONLINE

На сегодняшнем занятии:

1. Объекты
2. Создание объекта
3. Создание объекта и свойств в нем
4. Удаление свойства
5. Свойство из нескольких слов
6. Проверка существования свойства «in»
7. Цикл «for…in»
8. Оператор объединения ??
9. Опциональная цепочка '?.'
10. DOM-дерево
11. document.getElementById
12. getElementsBy*
13. querySelector

Объекты

Объекты используются для хранения коллекций различных значений и более сложных сущностей. В JavaScript объекты используются очень часто, это одна из основ языка.

Объект «Автомобиль»

Свойства:

- Цвет:
- Вес;
- Максимальная скорость;
- Объем двигателя.

Методы:

- Работа двигателя;
- Поворот колес;
- Система охлаждения салона;
- Блокировка дверей.

ITEA ONLINE

Создание объекта

Объект может быть создан с помощью фигурных скобок `{…}` с необязательным списком свойств. Свойство – это пара «*ключ: значение*», где *ключ* – это строка (также называемая «именем свойства»), а *значение* может быть чем угодно.

```
let user = new Object();  // синтаксис "конструктор объекта"
let user = {};            // синтаксис "литерал объекта"
```

Обычно используют вариант с фигурными скобками `{...}`. Такое объявление называют *литералом объекта* или *литеральной нотацией*.

ITEA ONLINE

Создание объекта и свойств в нем

При использовании литерального синтаксиса `{...}` мы сразу можем поместить в объект несколько свойств в виде пар «*ключ: значение*»:

```
let car = {
   color: "white",
   weight: 2.3,
   engine: 4.8,
   startEngine: function() { console.log("Engine start"); },
   blockDoors: function() { console.log("Block doors"); }
};

car.startEngine(); // Engine start
console.log(car.color); // white
```

ITEA ONLINE

Удаление свойства

Для удаления свойства мы можем использовать оператор `delete`

```
let car = {
   color: "white",
   weight: 2.3,
   engine: 4.8,
   startEngine: function() { console.log("Engine start"); },
   blockDoors: function() { console.log("Block doors"); }
};

delete car.color;          // Удаляем свойство color
console.log(car.color);    // undefined
delete car.color;          // Удаляем свойство color еще раз
                           // ошибки не будет
```

ITEA ONLINE

Свойство из нескольких слов

Имя свойства может состоять из нескольких слов, но тогда оно должно быть заключено в кавычки:

```
let car = {
   color: "white",
   weight: 2.3,
   "max speed": 200,
   engine: 4.8,
   startEngine: function() { console.log("Engine start"); },
   blockDoors: function() { console.log("Block doors"); }
};
car["max speed"] = 300;

console.log(car["max speed"]);     // 300
delete car["max speed"];/          // Удаляем свойство
```

I T E A ONLINE

Проверка существования свойства «in»

В отличие от многих других языков, особенность JavaScript-объектов в том, что можно получить доступ к любому свойству. Даже если свойства не существует – ошибки не будет!

```
let car = {
   color: "white",
   weight: 2.3,
   "max speed": 200
};

console.log("max" in car);         // false
console.log("color" in car);       // true
```

Обратите внимание, что слева от оператора `in` должно быть имя свойства. Обычно это строка в кавычках.

I T E A
ONLINE

Цикл «for...in»

Для перебора всех свойств объекта используется цикл `for..in`. Этот цикл отличается от изученного ранее цикла `for(;;)`.

```
let car = {
   color: "white",
   weight: 2.3,
   "max speed": 200
};

for (let prop in car) {
   console.log(prop);       // color, weight, max speed
   console.log(car[prop]); // white, 2.3, 200
}
```

ITEA ONLINE

Оператор объединения ??

Оператор объединения с `null`/`undefined` представляет собой два вопросительных знака `??`.

Результат выражения `a ?? b` будет следующим:

- `a`, если значение `a` определено,
- `b`, если значение `a` не определено.

То есть оператор `??` возвращает первый аргумент, если он не `null`/`undefined`, иначе второй.

Вот как можно переписать выражение `result = a ?? b`, используя уже знакомые нам операторы:

```
result = (a !== null && a !== undefined) ? a : b;
```

ITEA ONLINE

Опциональная цепочка '?.'

Опциональная цепочка `?.` — это безопасный способ доступа к свойствам вложенных объектов, даже если какое-либо из промежуточных свойств не существует.

Например, рассмотрим объекты для пользователей `user`. У большинства пользователей есть адрес `user.address` с улицей `user.address.street`, но некоторые адрес не указали.
В этом случае при попытке получить свойство `user.address.street` будет ошибка:

```
let user = {}; // пользователь без свойства address
console.log(user.address.street); // ошибка!

console.log(user?.address?.street); // null //undefined (без ошибки)
```

ITEA ONLINE

Опциональная цепочка '?.'

В веб-разработке нам бывает нужно получить данные об HTML-элементе, который иногда может отсутствовать на странице:

```
// Произойдёт ошибка, если querySelector(...) равен null.
let html = document.querySelector('.my-element').innerHTML;
```

До появления `?.` в языке для решения подобных проблем использовался оператор `&&`.

```
let user = {}; // пользователь без адреса
console.log( user && user.address && user.address.street ); // undefined (без ошибки)
```

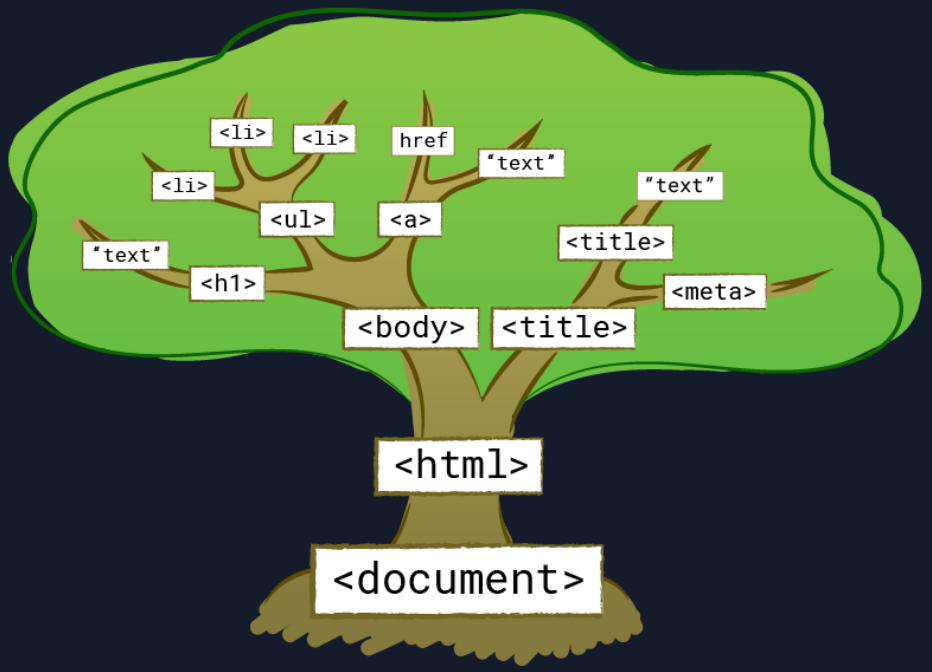
Использование логического `&&` со всей цепочкой свойств гарантирует, что все они существуют (а если нет – вычисление прекращается), но это довольно длинная и громоздкая конструкция.

DOM-дерево

Основой HTML-документа являются теги.
В соответствии с объектной моделью документа *(«Document Object Model»*, коротко DOM), каждый HTML-тег является объектом. Вложенные теги являются «детьми» родительского элемента. Текст, который находится внутри тега, также является объектом.

Все эти объекты доступны при помощи JavaScript, мы можем использовать их для изменения страницы.

Например, `document.body` – объект для тега `<body>`.



ITEA ONLINE

document.getElementById

Если у элемента есть атрибут id, то мы можем получить его вызовом `document.getElementById(id)`, где бы он ни находился.

```
<div id="elem">
  <div id="elem-content">Element</div>
</div>

<script>
  // получить элемент
  let elem = document.getElementById('elem');

  // сделать его фон красным
  elem.style.background = 'red';
</script>
```

ITEA ONLINE

getElementsBy*

На данный момент, они скорее исторические, так как `querySelector` более чем эффективен.

Здесь мы рассмотрим их для полноты картины, также вы можете встретить их в старом коде.

- `elem.getElementsByTagName(tag)` ищет элементы с данным тегом и возвращает их коллекцию.
- `elem.getElementsByClassName(className)` возвращает элементы, которые имеют данный CSS-класс.
- `document.getElementsByName(name)` возвращает элементы с заданным атрибутом name. Очень редко используется.

ITEA ONLINE

querySelector

Возвращает первый элемент документа, который соответствует указанному селектору или группе селекторов. Если совпадений не найдено, возвращает значение `null`.

В этом примере, нам вернется первый элемент в документе с классом `"myclass"`:

```
let el = document.querySelector(".myclass");
```

Селекторы, передаваемые в querySelector, могут быть использованы и для точного поиска, как показано в примере ниже. В данном примере возвращается элемент `<input name="login"/>`, находящийся в `<div class="user-panel main">`:

```
let el = document.querySelector("div.user-panel.main input[name=login]");
```

ITEA ONLINE

Q&A

ITEA

ONLINE