

```
<script>
(function() {
  class Ultimate_Shopify_DataLayer {
    constructor() {
      window.dataLayer = window.dataLayer || [];

      // use a prefix of events name
      this.eventPrefix = "";

      //Keep the value false to get non-formatted product ID
      this.formattedItemId = true;

      // data schema
      this.dataSchema = {
        ecommerce: {
          show: true
        },
        dynamicRemarketing: {
          show: false,
          business_vertical: 'retail'
        }
      }

      // add to wishlist selectors
      this.addToWishListSelectors = {
        'addWishListIcon': "",
        'gridItemSelector': "",
        'productLinkSelector': 'a[href*="/products/"]'
      }

      // quick view selectors
      this.quickViewSelector = {
        'quickViewElement': "",
        'gridItemSelector': "",
        'productLinkSelector': 'a[href*="/products/"]'
      }

      // mini cart button selector
      this.miniCartButton = [
        'a[href="/cart"]',
      ];
      this.miniCartAppersOn = 'click';
    }
  }
  new Ultimate_Shopify_DataLayer();
})();
```

```

// begin checkout buttons/links selectors
this.beginCheckoutButtons = [
  'input[name="checkout"]',
  'button[name="checkout"]',
  'a[href="/checkout"]',
  '.additional-checkout-buttons',
];

// direct checkout button selector
this.shopifyDirectCheckoutButton = [
  '.shopify-payment-button'
]

//Keep the value true if Add to Cart redirects to the cart page
this.isAddToCartRedirect = false;

// keep the value false if cart items increment/decrement/remove refresh page
this.isAjaxCartIncrementDecrement = true;

// Caution: Do not modify anything below this line, as it may result in it not functioning
correctly.
this.cart = {{ cart | json }}
this.countryCode = "{{ shop.address.country_code }}";
this.storeURL = "{{ shop.secure_url }}";
localStorage.setItem('shopCountryCode', this.countryCode);
this.collectData();
this.itemsList = [];
}

updateCart() {
  fetch("/cart.js")
    .then((response) => response.json())
    .then((data) => {
      this.cart = data;
    });
}

debounce(delay) {
  let timeoutId;
  return function(func) {
    const context = this;
    const args = arguments;

```

```

clearTimeout(timeoutId);

timeoutId = setTimeout(function() {
    func.apply(context, args);
}, delay);
};
}

eventConsole(eventName, eventData) {
    const css1 = 'background: red; color: #fff; font-size: normal; border-radius: 3px 0 0 3px; padding: 3px 4px;';
    const css2 = 'background-color: blue; color: #fff; font-size: normal; border-radius: 0 3px 3px 0; padding: 3px 4px;';
    console.log('%cGTM DataLayer Event:%c' + eventName, css1, css2, eventData);
}

collectData() {
    this.customerData();
    this.ajaxRequestData();
    this.searchPageData();
    this.miniCartData();
    this.beginCheckoutData();

    {% if template contains 'cart' %}
        this.viewCartPageData();
    {% endif %}

    {% if template contains 'product' %}
        this.productSinglePage();
    {% endif %}

    {% if template contains 'collection' %}
        this.collectionsPageData();
    {% endif %}

    this.addToWishListData();
    this.quickViewData();
    this.selectItemData();
    this.formData();
    this.phoneClickData();
    this.emailClickData();
    this.loginRegisterData();
}

```

```

//logged-in customer data
customerData() {
  const currentUser = {};
  {% if customer %}
    currentUser.id = "{{ customer.id }}";
    currentUser.first_name = "{{ customer.first_name }}";
    currentUser.last_name = "{{ customer.last_name }}";
    currentUser.full_name = "{{ customer.name }}";
    currentUser.email = "{{ customer.email }}";
    currentUser.phone = "{{ customer.default_address.phone }}";

    {% if customer.default_address %}
      currentUser.address = {
        address_summary: "{{ customer.default_address.summary }}",
        address1: "{{ customer.default_address.address1 }}",
        address2: "{{ customer.default_address.address2 }}",
        city: "{{ customer.default_address.city }}",
        street: "{{ customer.default_address.street }}",
        zip: "{{ customer.default_address.zip }}",
        company: "{{ customer.default_address.company }}",
        country: "{{ customer.default_address.country.name }}",
        countryCode: "{{ customer.default_address.country_code }}",
        province: "{{ customer.default_address.province }}"
      };
    {% endif %}
  {% endif %}

  if (currentUser.email) {
    currentUser.hash_email = "{{ customer.email | sha256 }}"
  }

  if (currentUser.phone) {
    currentUser.hash_phone = "{{ customer.phone | sha256 }}"
  }

  window.dataLayer = window.dataLayer || [];
  dataLayer.push({
    customer: currentUser
  });
}

// add_to_cart, remove_from_cart, search
ajaxRequestData() {
  const self = this;

```

```

// handle non-ajax add to cart
if(this.isAddToCartRedirect) {
  document.addEventListener('submit', function(event) {
    const addToCartForm = event.target.closest('form[action="/cart/add"]');
    if(addToCartForm) {
      event.preventDefault();

      const formData = new FormData(addToCartForm);

      fetch(window.Shopify.routes.root + 'cart/add.js', {
        method: 'POST',
        body: formData
      })
      .then(response => {
        window.location.href = "{{ routes.cart_url }}";
      })
      .catch((error) => {
        console.error('Error:', error);
      });
    }
  });
}

// fetch
let originalFetch = window.fetch;
let debounce = this.debounce(800);

window.fetch = function () {
  return originalFetch.apply(this, arguments).then((response) => {
    if (response.ok) {
      let cloneResponse = response.clone();
      let requestURL = arguments[0]['url'] || arguments[0];

      if(/.*\VsearchV?.*\?.*q=.*\./test(requestURL) &&
!requestURL.includes('&requestFrom=uldt')) {
        const queryString = requestURL.split('?')[1];
        const urlParams = new URLSearchParams(queryString);
        const search_term = urlParams.get("q");

        debounce(function() {
          fetch(`${self.storeURL}/search/suggest.json?q=${search_term}&resources[type]=product&requestFrom=uldt`)

```

```

.then(res => res.json())
.then(function(data) {
  const products = data.resources.results.products;
  if(products.length) {
    const fetchRequests = products.map(product =>
      fetch(`${self.storeURL}/${product.url.split('?')[0]}.js`)
        .then(response => response.json())
        .catch(error => console.error('Error fetching:', error))
    );

    Promise.all(fetchRequests)
      .then(products => {
        const items = products.map((product) => {
          return {
            product_id: product.id,
            product_title: product.title,
            variant_id: product.variants[0].id,
            variant_title: product.variants[0].title,
            vendor: product.vendor,
            total_discount: 0,
            final_price: product.price_min,
            product_type: product.type,
            quantity: 1
          }
        });

        self.ecommerceDataLayer('search', {search_term, items});
      })
    }else {
      self.ecommerceDataLayer('search', {search_term, items: []});
    }
  });
});
}
else if (requestURL.includes("/cart/add")) {
  cloneResponse.text().then((text) => {
    let data = JSON.parse(text);

    if(data.items && Array.isArray(data.items)) {
      data.items.forEach(function(item) {
        self.ecommerceDataLayer('add_to_cart', {items: [item]});
      })
    } else {
      self.ecommerceDataLayer('add_to_cart', {items: [data]});
    }
  });
}

```

```

    }
    self.updateCart();
  });
}else if(requestURL.includes("/cart/change") || requestURL.includes("/cart/update")) {

  cloneResponse.text().then((text) => {

    let newCart = JSON.parse(text);
    let newCartItems = newCart.items;
    let oldCartItems = self.cart.items;

    for(let i = 0; i < oldCartItems.length; i++) {
      let item = oldCartItems[i];
      let newItem = newCartItems.find(newItems => newItems.id === item.id);

      if(newItem) {

        if(newItem.quantity > item.quantity) {
          // cart item increment
          let quantity = (newItem.quantity - item.quantity);
          let updatedItem = {...item, quantity}
          self.ecommerceDataLayer('add_to_cart', {items: [updatedItem]});
          self.updateCart();

        }else if(newItem.quantity < item.quantity) {
          // cart item decrement
          let quantity = (item.quantity - newItem.quantity);
          let updatedItem = {...item, quantity}
          self.ecommerceDataLayer('remove_from_cart', {items: [updatedItem]});
          self.updateCart();
        }

      }

    }else {
      self.ecommerceDataLayer('remove_from_cart', {items: [item]});
      self.updateCart();
    }
  });
}

return response;

```

```

    });
  }
  // end fetch

  //xhr
  var origXMLHttpRequest = XMLHttpRequest;
  XMLHttpRequest = function() {
    var requestURL;

    var xhr = new origXMLHttpRequest();
    var origOpen = xhr.open;
    var origSend = xhr.send;

    // Override the `open` function.
    xhr.open = function(method, url) {
      requestURL = url;
      return origOpen.apply(this, arguments);
    };

    xhr.send = function() {

      // Only proceed if the request URL matches what we're looking for.
      if (requestURL.includes("/cart/add") || requestURL.includes("/cart/change") ||
        /\.\/searchV?.*\?.*q=.\+/.test(requestURL)) {

        xhr.addEventListener('load', function() {
          if (xhr.readyState === 4) {
            if (xhr.status >= 200 && xhr.status < 400) {

              if(/\.\/searchV?.*\?.*q=.\+/.test(requestURL) &&
!requestURL.includes('&requestFrom=uldt')) {
                const queryString = requestURL.split('?')[1];
                const urlParams = new URLSearchParams(queryString);
                const search_term = urlParams.get("q");

                debounce(function() {

fetch(`${self.storeURL}/search/suggest.json?q=${search_term}&resources[type]=product&requestFrom=uldt`)

                    .then(res => res.json())
                    .then(function(data) {
                      const products = data.resources.results.products;

```

```

    if(products.length) {
      const fetchRequests = products.map(product =>
        fetch(`${self.storeURL}/${product.url.split('?')[0]}.js`)
          .then(response => response.json())
          .catch(error => console.error('Error fetching:', error))
      );

      Promise.all(fetchRequests)
        .then(products => {
          const items = products.map((product) => {
            return {
              product_id: product.id,
              product_title: product.title,
              variant_id: product.variants[0].id,
              variant_title: product.variants[0].title,
              vendor: product.vendor,
              total_discount: 0,
              final_price: product.price_min,
              product_type: product.type,
              quantity: 1
            }
          });

          self.ecommerceDataLayer('search', {search_term, items});
        })
    } else {
      self.ecommerceDataLayer('search', {search_term, items: []});
    }
  });
});

}

else if(requestURL.includes("/cart/add")) {
  const data = JSON.parse(xhr.responseText);

  if(data.items && Array.isArray(data.items)) {
    data.items.forEach(function(item) {
      self.ecommerceDataLayer('add_to_cart', {items: [item]});
    })
  } else {
    self.ecommerceDataLayer('add_to_cart', {items: [data]});
  }
  self.updateCart();
}

```

```

    }else if(requestURL.includes("/cart/change")) {

        const newCart = JSON.parse(xhr.responseText);
        const newCartItems = newCart.items;
        let oldCartItems = self.cart.items;

        for(let i = 0; i < oldCartItems.length; i++) {
            let item = oldCartItems[i];
            let newItem = newCartItems.find(newItems => newItems.id === item.id);

            if(newItem) {
                if(newItem.quantity > item.quantity) {
                    // cart item increment
                    let quantity = (newItem.quantity - item.quantity);
                    let updatedItem = {...item, quantity}
                    self.ecommerceDataLayer('add_to_cart', {items: [updatedItem]});
                    self.updateCart();

                }else if(newItem.quantity < item.quantity) {
                    // cart item decrement
                    let quantity = (item.quantity - newItem.quantity);
                    let updatedItem = {...item, quantity}
                    self.ecommerceDataLayer('remove_from_cart', {items: [updatedItem]});
                    self.updateCart();
                }

            }else {
                self.ecommerceDataLayer('remove_from_cart', {items: [item]});
                self.updateCart();
            }
        }
    }
});
}

return origSend.apply(this, arguments);
};

return xhr;

```

```

    };
    //end xhr
}

// search event from search page
searchPageData() {
    const self = this;
    let pageUrl = window.location.href;

    if(/.+\/search\?.*\&q=+.+/.test(pageUrl)) {
        const queryString = pageUrl.split('?')[1];
        const urlParams = new URLSearchParams(queryString);
        const search_term = urlParams.get("q");

        fetch(`${shop.secure_url
    }}/search/suggest.json?q=${search_term}&resources[type]=product&requestFrom=uldt`)
        .then(res => res.json())
        .then(function(data) {
            const products = data.resources.results.products;
            if(products.length) {
                const fetchRequests = products.map(product =>
                    fetch(`${self.storeURL}/${product.url.split('?')[0]}.js`)
                    .then(response => response.json())
                    .catch(error => console.error('Error fetching:', error))
                );
                Promise.all(fetchRequests)
                .then(products => {
                    const items = products.map((product) => {
                        return {
                            product_id: product.id,
                            product_title: product.title,
                            variant_id: product.variants[0].id,
                            variant_title: product.variants[0].title,
                            vendor: product.vendor,
                            total_discount: 0,
                            final_price: product.price_min,
                            product_type: product.type,
                            quantity: 1
                        }
                    }
                ));

                self.ecommerceDataLayer('search', {search_term, items});
            }
        })
    } else {

```

```

        self.ecommerceDataLayer('search', {search_term, items: []});
    }
    });
}
}

// view_cart
miniCartData() {
  if(this.miniCartButton.length) {
    let self = this;
    if(this.miniCartAppersOn === 'hover') {
      this.miniCartAppersOn = 'mouseenter';
    }
    this.miniCartButton.forEach((selector) => {
      let miniCartButtons = document.querySelectorAll(selector);
      miniCartButtons.forEach((miniCartButton) => {
        miniCartButton.addEventListener(self.miniCartAppersOn, () => {
          self.ecommerceDataLayer('view_cart', self.cart);
        });
      })
    });
  }
}

// begin_checkout
beginCheckoutData() {
  let self = this;
  document.addEventListener('pointerdown', (event) => {
    let targetElement = event.target.closest(self.beginCheckoutButtons.join(', '));
    if(targetElement) {
      self.ecommerceDataLayer('begin_checkout', self.cart);
    }
  });
}

// view_cart, add_to_cart, remove_from_cart
viewCartPageData() {

  this.ecommerceDataLayer('view_cart', this.cart);

  //if cart quantity chagne reload page
  if(!this.isAjaxCartIncrementDecrement) {
    const self = this;
    document.addEventListener('pointerdown', (event) => {

```

```

const target = event.target.closest('a[href*="/cart/change?"]');
if(target) {
  const linkUrl = target.getAttribute('href');
  const queryString = linkUrl.split("?")[1];
  const urlParams = new URLSearchParams(queryString);
  const newQuantity = urlParams.get("quantity");
  const line = urlParams.get("line");
  const cart_id = urlParams.get("id");

  if(newQuantity && (line || cart_id)) {
    let item = line ? {...self.cart.items[line - 1]} : self.cart.items.find(item => item.key ===
cart_id);

    let event = 'add_to_cart';
    if(newQuantity < item.quantity) {
      event = 'remove_from_cart';
    }

    let quantity = Math.abs(newQuantity - item.quantity);
    item['quantity'] = quantity;

    self.ecommerceDataLayer(event, {items: [item]});
  }
}
});
}
}

productSinglePage() {
  {% if template contains 'product' %}
  const item = {
    product_id: {{ product.id | json }},
    variant_id: {{ product.selected_or_first_available_variant.id }},
    product_title: {{ product.title | json }},
    line_level_total_discount: 0,
    vendor: {{ product.vendor | json }},
    sku: {{ product.selected_or_first_available_variant.sku | json }},
    product_type: {{ product.type | json }},
    item_list_id: {{ product.collections[0].id | json }},
    item_list_name: {{ product.collections[0].title | json }},
    {% if product.selected_or_first_available_variant.title != "Default Title" %}
    variant_title: {{ product.selected_or_first_available_variant.title | json }},
    {% endif %}
  }

```

```

    final_price: {{ product.selected_or_first_available_variant.price }},
    quantity: 1
  };

const variants = {{ product.variants | json }}
this.ecommerceDataLayer('view_item', {items: [item]});

if(this.shopifyDirectCheckoutButton.length) {
  let self = this;
  document.addEventListener('pointerdown', (event) => {
    let target = event.target;
    let checkoutButton = event.target.closest(this.shopifyDirectCheckoutButton.join(', '));

    if(checkoutButton && (variants || self.quickViewVariants)) {

      let checkoutForm = checkoutButton.closest('form[action="/cart/add"]');
      if(checkoutForm) {

        let variant_id = null;
        let variantInput = checkoutForm.querySelector('input[name="id"]');
        let variantIdFromURL = new
URLSearchParams(window.location.search).get('variant');
        let firstVariantId = item.variant_id;

        if(varientInput) {
          variant_id = parseInt(varientInput.value);
        }else if(varientIdFromURL) {
          variant_id = varientIdFromURL;
        }else if(firstVariantId) {
          variant_id = firstVariantId;
        }

        if(variant_id) {
          variant_id = parseInt(variant_id);

          let quantity = 1;
          let quantitySelector = checkoutForm.getAttribute('id');
          if(quantitySelector) {
            let quantityInput =
document.querySelector('input[name="quantity"][form="'+quantitySelector+'"]');
            if(quantityInput) {
              quantity = +quantityInput.value;
            }
          }
        }
      }
    }
  });
}

```

```

    if(variant_id) {
      let variant = variants.find(item => item.id === +variant_id);
      if(variant && item) {
        variant_id
        item['variant_id'] = variant_id;
        item['variant_title'] = variant.title;
        item['final_price'] = variant.price;
        item['quantity'] = quantity;

        self.ecommerceDataLayer('add_to_cart', {items: [item]});
        self.ecommerceDataLayer('begin_checkout', {items: [item]});
      }else if(self.quickViewedItem) {
        let variant = self.quickViewVariants.find(item => item.id === +variant_id);
        if(variant) {
          self.quickViewedItem['variant_id'] = variant_id;
          self.quickViewedItem['variant_title'] = variant.title;
          self.quickViewedItem['final_price'] = parseFloat(variant.price) * 100;
          self.quickViewedItem['quantity'] = quantity;

          self.ecommerceDataLayer('add_to_cart', {items: [self.quickViewedItem]});
          self.ecommerceDataLayer('begin_checkout', {items:
[self.quickViewedItem]});

          }
        }
      }
    }
  }
});
}

{% endif %}
}

```

```

collectionsPageData() {
  var ecommerce = {
    'items': [
      {% for product in collection.products %}
      {
        'product_id': {{ product.id | json }},
        'variant_id': {{ product.selected_or_first_available_variant.id | json }},

```

```

        'vendor': {{ product.vendor | json }},
        'sku': {{ product.selected_or_first_available_variant.sku | json }},
        'total_discount': 0,
        'variant_title': {{ product.selected_or_first_available_variant.title | json }},
        'product_title': {{ product.title | json }},
        'final_price': Number({{ product.price }}),
        'product_type': {{ product.type | json }},
        'item_list_id': {{ collection.id | json }},
        'item_list_name': {{ collection.title | json }},
        'url': {{product.url | json}},
        'quantity': 1
    },
    {% endfor %}
]
};

```

```

this.itemsList = ecommerce.items;
ecommerce['item_list_id'] = {{ collection.id | json }}
ecommerce['item_list_name'] = {{ collection.title | json }}

this.ecommerceDataLayer('view_item_list', ecommerce);
}

```

```

// add to wishlist
addToWishListData() {
    if(this.addToWishListSelectors && this.addToWishListSelectors.addWishListIcon) {
        const self = this;
        document.addEventListener('pointerdown', (event) => {
            let target = event.target;

            if(target.closest(self.addToWishListSelectors.addWishListIcon)) {
                let pageURL = window.location.href.replace(/^\?.+/, "");
                let requestURL = undefined;

                if(/\/products\/[^\/]+$/i.test(pageURL)) {
                    requestURL = pageURL;
                } else if(self.addToWishListSelectors.gridItemSelector &&
self.addToWishListSelectors.productLinkSelector) {
                    let itemElement = target.closest(self.addToWishListSelectors.gridItemSelector);
                    if(itemElement) {
                        let linkElement =
itemElement.querySelector(self.addToWishListSelectors.productLinkSelector);
                        if(linkElement) {

```

```

        let link = linkElement.getAttribute('href').replace(/\?.+\/g, "");
        if(link && /\productsV[/^]+$/i.test(link)) {
            requestURL = link;
        }
    }
}
}

if(requestURL) {
    fetch(requestURL + '.json')
        .then(res => res.json())
        .then(result => {
            let data = result.product;
            if(data) {
                let dataLayerData = {
                    product_id: data.id,
                    variant_id: data.variants[0].id,
                    product_title: data.title,
                    quantity: 1,
                    final_price: parseFloat(data.variants[0].price) * 100,
                    total_discount: 0,
                    product_type: data.product_type,
                    vendor: data.vendor,
                    variant_title: (data.variants[0].title !== 'Default Title') ? data.variants[0].title :
undefined,
                    sku: data.variants[0].sku,
                }

                self.ecommerceDataLayer('add_to_wishlist', {items: [dataLayerData]});
            }
        });
    }
}
});
}
}

quickViewData() {
    if(this.quickViewSelector.quickViewElement && this.quickViewSelector.gridItemSelector
&& this.quickViewSelector.productLinkSelector) {
        const self = this;
        document.addEventListener('pointerdown', (event) => {
            let target = event.target;
            if(target.closest(self.quickViewSelector.quickViewElement)) {

```

```

let requestURL = undefined;
let itemElement = target.closest(this.quickViewSelector.gridItemSelector );

if(itemElement) {
  let linkElement =
itemElement.querySelector(self.quickViewSelector.productLinkSelector);
  if(linkElement) {
    let link = linkElement.getAttribute('href').replace(/\?.+/, "");
    if(link && /\productsV[/^/]+$/i.test(link)) {
      requestURL = link;
    }
  }
}

if(requestURL) {
  fetch(requestURL + '.json')
    .then(res => res.json())
    .then(result => {
      let data = result.product;
      if(data) {
        let dataLayerData = {
          product_id: data.id,
          variant_id: data.variants[0].id,
          product_title: data.title,
          quantity: 1,
          final_price: parseFloat(data.variants[0].price) * 100,
          total_discount: 0,
          product_type: data.product_type,
          vendor: data.vendor,
          variant_title: (data.variants[0].title !== 'Default Title') ? data.variants[0].title :
undefined,
          sku: data.variants[0].sku,
        }

        self.ecommerceDataLayer('view_item', {items: [dataLayerData]});
        self.quickViewVariants = data.variants;
        self.quickViewedItem = dataLayerData;
      }
    });
}
});

{% unless template contains 'product' %}

```

```

if(this.shopifyDirectCheckoutButton.length) {
  let self = this;
  document.addEventListener('pointerdown', (event) => {
    let target = event.target;
    let checkoutButton = event.target.closest(this.shopifyDirectCheckoutButton.join(', '));

    if(self.quickViewVariants && self.quickViewedItem && self.quickViewVariants.length
    && checkoutButton) {

      let checkoutForm = checkoutButton.closest('form[action*="/cart/add"]');
      if(checkoutForm) {
        let quantity = 1;
        let variantInput = checkoutForm.querySelector('input[name="id"]');
        let quantitySelector = checkoutForm.getAttribute('id');

        if(quantitySelector) {
          let quantityInput =
document.querySelector('input[name="quantity"]')[form="'+quantitySelector+'"]);
          if(quantityInput) {
            quantity = +quantityInput.value;
          }
        }

        if(varientInput) {
          let variant_id = parseInt(varientInput.value);

          if(variant_id) {
            const variant = self.quickViewVariants.find(item => item.id === +variant_id);
            if(variant && self.quickViewedItem) {
              self.quickViewedItem['variant_id'] = variant_id;
              self.quickViewedItem['variant_title'] = variant.title;
              self.quickViewedItem['final_price'] = parseFloat(variant.price) * 100;
              self.quickViewedItem['quantity'] = quantity;

              self.ecommerceDataLayer('add_to_cart', {items: [self.quickViewedItem]});
              self.ecommerceDataLayer('begin_checkout', {items:
[self.quickViewedItem]});
            }
          }
        }
      }
    }
  });
}

```

```

    }
    {% endunless %}
  }
}

// select_item events
selectItemData() {

  const self = this;
  const items = this.itemsList;

  {% if template contains 'collection' %}
  document.addEventListener('pointerdown', function(event) {

    const productLink = event.target.closest('a[href*="/products/"]');

    if(productLink) {
      const linkUrl = productLink.getAttribute('href');

      const matchProduct = (item) => {
        var itemSlug = (item.url.split('/products/')[1]).split('#')[0].split('?')[0].trim();
        var linkUrlItemSlug = (linkUrl.split('/products/')[1]).split('#')[0].split('?')[0].trim();

        return itemSlug === linkUrlItemSlug;
      }

      const item = items.find(matchProduct);
      const index = items.findIndex(matchProduct);

      if(item) {
        self.ecommerceDataLayer('select_item', {items: [{...item, index: index}]});
      }
    }
  });
  {% endif %}

  // select item on variant change
  document.addEventListener('variant:change', function(event) {
    const product_id = event.detail.product.id;
    const variant_id = event.detail.variant.id;
    const vendor = event.detail.product.vendor;
    const variant_title = event.detail.variant.public_title;
    const product_title = event.detail.product.title;
    const final_price = event.detail.variant.price;
  });
}

```

```

const product_type = event.detail.product.type;

const item = {
  product_id: product_id,
  product_title: product_title,
  variant_id: variant_id,
  variant_title: variant_title,
  vendor: vendor,
  final_price: final_price,
  product_type: product_type,
  quantity: 1
}

self.ecommerceDataLayer('select_item', {items: [item]});
});
}

// all ecommerce events
ecommerceDataLayer(event, data) {
  const self = this;
  dataLayer.push({ 'ecommerce': null });
  const dataLayerData = {
    "event": this.eventPrefix + event,
    'ecommerce': {
      'currency': this.cart.currency,
      'items': data.items.map((item, index) => {
        const dataLayerItem = {
          'index': index,
          'item_id': this.formattedItemId ?
`shopify_${this.countryCode}_${item.product_id}_${item.variant_id}` : item.product_id.toString(),
          'product_id': item.product_id.toString(),
          'variant_id': item.variant_id.toString(),
          'item_name': item.product_title,
          'quantity': item.quantity,
          'price': +((item.final_price / 100).toFixed(2)),
          'discount': item.total_discount ? +((item.total_discount / 100).toFixed(2)) : 0
        }

        if(item.product_type) {
          dataLayerItem['item_category'] = item.product_type;
        }

        if(item.vendor) {
          dataLayerItem['item_brand'] = item.vendor;
        }
      })
    }
  };
  dataLayer.push(dataLayerData);
}

```

```

    }

    if(item.variant_title && item.variant_title !== 'Default Title') {
      dataLayerItem['item_variant'] = item.variant_title;
    }

    if(item.sku) {
      dataLayerItem['sku'] = item.sku;
    }

    if(item.item_list_name) {
      dataLayerItem['item_list_name'] = item.item_list_name;
    }

    if(item.item_list_id) {
      dataLayerItem['item_list_id'] = item.item_list_id.toString()
    }

    return dataLayerItem;
  })
}
}

if(data.total_price !== undefined) {
  dataLayerData['ecommerce']['value'] = +((data.total_price / 100).toFixed(2));
} else {
  dataLayerData['ecommerce']['value'] =
+(dataLayerData['ecommerce']['items'].reduce((total, item) => total + (item.price * item.quantity),
0)).toFixed(2);
}

if(data.item_list_id) {
  dataLayerData['ecommerce']['item_list_id'] = data.item_list_id;
}

if(data.item_list_name) {
  dataLayerData['ecommerce']['item_list_name'] = data.item_list_name;
}

if(data.search_term) {
  dataLayerData['search_term'] = data.search_term;
}

```

```

        if(self.dataSchema.dynamicRemarketing &&
self.dataSchema.dynamicRemarketing.show) {
            dataLayer.push({ 'dynamicRemarketing': null });
            dataLayerData['dynamicRemarketing'] = {
                value: dataLayerData.ecommerce.value,
                items: dataLayerData.ecommerce.items.map(item => ({id: item.item_id,
google_business_vertical: self.dataSchema.dynamicRemarketing.business_vertical}))
            }
        }

        if(!self.dataSchema.ecommerce || !self.dataSchema.ecommerce.show) {
            delete dataLayerData['ecommerce'];
        }

        dataLayer.push(dataLayerData);
        self.eventConsole(self.eventPrefix + event, dataLayerData);
    }

// contact form submit & newsletters signup
formData() {
    const self = this;
    document.addEventListener('submit', function(event) {

        let targetForm = event.target.closest('form[action^="/contact"]');

        if(targetForm) {
            const formData = {
                form_location: window.location.href,
                form_id: targetForm.getAttribute('id'),
                form_classes: targetForm.getAttribute('class')
            };

            let formType = targetForm.querySelector("input[name='form_type']");
            let inputs = targetForm.querySelectorAll("input:not([type=hidden]):not([type=submit]),
textarea, select");

            inputs.forEach(function(input) {
                var inputName = input.name;
                var inputValue = input.value;

                if (inputName && inputValue) {
                    var matches = inputName.match(/\[(.*)\]/);

```

```

        if (matches && matches.length > 1) {
            var fieldName = matches[1];
            formData[fieldName] = input.value;
        }
    }
});

if(formType && formType.value === 'customer') {
    dataLayer.push({ event: self.eventPrefix + 'newsletter_signup', ...formData});
    self.eventConsole(self.eventPrefix + 'newsletter_signup', { event: self.eventPrefix +
'newsletter_signup', ...formData});

    } else if(formType && formType.value === 'contact') {
        dataLayer.push({ event: self.eventPrefix + 'contact_form_submit', ...formData});
        self.eventConsole(self.eventPrefix + 'contact_form_submit', { event: self.eventPrefix +
'contact_form_submit', ...formData});
    }
}
});

}

// phone_number_click event
phoneClickData() {
    const self = this;
    document.addEventListener('click', function(event) {
        let target = event.target.closest('a[href^="tel:"]');
        if(target) {
            let phone_number = target.getAttribute('href').replace('tel:', '');
            let eventData = {
                event: self.eventPrefix + 'phone_number_click',
                page_location: window.location.href,
                link_classes: target.getAttribute('class'),
                link_id: target.getAttribute('id'),
                phone_number
            }

            dataLayer.push(eventData);
            this.eventConsole(self.eventPrefix + 'phone_number_click', eventData);
        }
    });
}

// email_click event

```

```

emailClickData() {
  const self = this;
  document.addEventListener('click', function(event) {
    let target = event.target.closest('a[href^="mailto:"]');
    if(target) {
      let email_address = target.getAttribute('href').replace('mailto:', '');
      let eventData = {
        event: self.eventPrefix + 'email_click',
        page_location: window.location.href,
        link_classes: target.getAttribute('class'),
        link_id: target.getAttribute('id'),
        email_address
      }

      dataLayer.push(eventData);
      this.eventConsole(self.eventPrefix + 'email_click', eventData);
    }
  });
}

```

//login register

```

loginRegisterData() {

  const self = this;
  let isTrackedLogin = false;
  let isTrackedRegister = false;

  if(window.location.href.includes('/account/login')) {
    document.addEventListener('submit', function(e) {
      const loginForm = e.target.closest('[action="/account/login"]');
      if(loginForm && !isTrackedLogin) {
        const eventData = {
          event: self.eventPrefix + 'login'
        }
        isTrackedLogin = true;
        dataLayer.push(eventData);
        self.eventConsole(self.eventPrefix + 'login', eventData);
      }
    });
  }
}

```

```

if(window.location.href.includes('/account/register')) {
  document.addEventListener('submit', function(e) {
    const registerForm = e.target.closest('[action="/account"]');

```

```
if(registerForm && !isTrackedRegister) {
  const eventData = {
    event: self.eventPrefix + 'sign_up'
  }

  isTrackedRegister = true;
  dataLayer.push(eventData);
  self.eventConsole(self.eventPrefix + 'sign_up', eventData);
}
});
}
}
}
// end Ultimate_Shopify_DataLayer

document.addEventListener('DOMContentLoaded', function() {
  try{
    new Ultimate_Shopify_DataLayer();
  }catch(error) {
    console.log(error);
  }
});

})();
</script>
```